NTNU

Norwegian University of
Science and Technology

Introduction to digital core analysis: 3D reconstruction, numerical flow simulations and pore network modeling

Radmila Mandzhieva

Petroleum Engineering
Submission date: June 2017
Supervisor: Ole Torsæter, IGP

Norwegian University of Science and Technology
Department of Geoscience and Petroleum

To my beloved family…

“Science is beautiful when it makes simple explanations of phenomena or connections between different observations.”

- Stephen Hawking

“All models are wrong, but some are useful.”

- George Box

Abstract

Digital core analysis can replace the use of conventional core samples by decreasing the time required for conducting experiments. Moreover, it gets a better understanding of complex reservoir structures due to its ability to capture pore geometries and fluid behavior at pore scale level.

In the current thesis, the whole workflow of digital core analysis – from image study to overlook of pore network models – had been performed. It started from analysis of color 2D thin section images. It was noticed that they might contain significant amount of information about pore microstructure. In addition, several different approaches of image binarization were considered.

Then some 3D micro-CT models (incl. four sandstone and one carbonate samples from the open database of the Imperial College London) had been analyzed and reconstructed. To calculate the absolute permeability, a direct pore-scale modeling approach using the lattice Boltzmann method (LBM) was implemented in PALABOS software. It is worth noting that several parameters influenced on the obtained results, for example, the image resolution and the volume of analyzed microstructure. It was found that while using low quality images the permeability by LBM might be significantly overestimated. In addition, to obtain reliable permeability (i.e. close to experimentally measured), the volume should be greater or equal to representative elementary volume (REV). Furthermore, it was possible to compare LBM permeability for reconstructed cases and original models. Results showed the best consistency for medium permeable sandstone samples. However, it had been noticed that reconstruction was not able to produce exactly the same porous media, since several other properties of reconstructed models, e.g. formation factor and specific surface area, were differed from original inputs.

When multiphase flow occurs in porous structures, it is difficult to estimate relative permeabilities directly, since in this case REV should be bigger and LBM calculations are very computationally expensive. That is the reason why the pore network modeling approach was briefly discussed, and several models generated by maximal ball extraction algorithm had been reviewed.

Acknowledgements

Time has been passed so fast, and now it is my turn to write “Acknowledgements” section in the master thesis.

I would like to express my deepest appreciation to my supervisor, Prof. Ole Torsæter, for his valuable comments and suggestions how to improve this work. He always shows me a right direction to move forward by his own example and enthusiasm in doing research. I also want to gratitude my co-supervisor, Dr. Hamid Hosseinzade Khanamiri, for his comprehensive feedbacks and active involvement to this project. In addition, special thanks should be said to Prof. Mai Britt Mørk for preparing thin sections of the Wessex Basin’s samples.

This thesis would not be done without using open source programs and input models. Therefore, I would like to express my sincere gratitude to FlowKit Ltd. for PALABOS and to OpenPNM for the pore network modeling package. Furthermore, I am thankful to the PERM Research Group and Prof. Martin Blunt in the Imperial College London for such a great open database of micro-CT models. My deepest appreciation should also be said to Dr. Samuel Cooper (Imperial College London) for his MATLAB app to calculate formation factors. Moreover, I want to thank Prof. Hongyan Yuan from the University of Rhode Island for a part of MATLAB code, as well as Dr. Arash Rabbani and Prof. Chenfeng Li for answering my questions about watershed algorithm and stochastic reconstruction. I appreciate intentions of these researchers to share their ideas and work by creating a common database of knowledge for everyone.

Furthermore, I am extremely grateful to my family for all the love and support that they give me every day. You are always in my heart, and I dedicate this work to you.

I must also thank the Norwegian government for providing me an opportunity to study at NTNU and to meet new friends from all over the world.

Finally, a big thanks to my groupmates and neighbours here, in Trondheim, for our tea meetings and common study during these two years. I especially thank my friend, Verónica Torres, who checked this document to detect possible linguistic inaccuracies.

Table of Contents

List of Figures

List of Tables

Abbreviations and Acronyms

1D	1-dimentional
2D	2-dimentional
3D	3-dimentional
µm	Micron
ANU	Australian National University, a research university located in Canberra
BGK operator	Bhatnagar-Gross-Krook operator
BMP	Bitmap Picture
BSE	Backscattered Electrons
CT	Computer tomography
DAT-file	Data file used as an input to PALABOS
deg.	Degrees
FFT	Fast Fourier Transform
GRF	Gaussian Random Field
IDE	Integrated Development Environment, software application that consists of a source code, editor, build automation tools and a debugger
incl.	Including
l.u.	Lattice Units
LBM	Lattice Boltzmann Method
MATLAB	Matrix Laboratory, programming language developed by MathWorks
MRT model	Multiple-Relaxation-Time model
NTNU	Norwegian University of Science and Technology, the biggest university in Norway
PALABOS	PArallel LAttice BOltzmann Solver, open source software developed by FlowKit Ltd.
PNM	Pore Network Modeling
PPM file	Portable Pixmap Image file
PSD	Pore Size Distribution
PSD function	Power Spectral Density function
REV	Representative Elementary Volume
RGB	Red Green Blue; an RGB image has these three channels
SE	Secondary Electrons
SEM	Scanning Electron Microscopy
Spyder	Scientific PYthon Development EnviRonment
SSA	Specific Surface Area
TIFF	Tagged Image File Format
VTK	Visualization Tool Kit

Chapter 1.
Introduction

Porosity, pore size distribution and average pore diameter, as well as pore connectivity and formation factor are important parameters of every porous structure. Correctly estimated, they play a significant role in determining transport properties of reservoir rocks, since to have accurate predictions of absolute and relative permeabilities, realistic input of the pore space is highly required.

The actual pore structure can be obtained either by direct imaging of core samples (e.g. by computer tomography) or by 3D reconstruction of thin sections. In some cases, the direct CT method may not be reliable, e.g. in carbonate reservoir studies due to their submicron structures and the similarity in X-ray absorption for calcite and dolomite (Sakellariou, et al., 2003). That is why, there are several approaches to reconstruct 3D porous media from 2D thin section images, and perhaps the most well studied technique deals with a Gaussian random field concept, which is discussed in more details in Chapter 3.

The practical importance of using pore-scale modeling is related to estimation of such macroscopic parameters as capillary pressure, absolute and relative permeabilities which are important data-in of almost every reservoir model. With a current development of computational methods in imaging, digital core analysis can become a good alternative of expensive laboratory experiments and existing phenomenological approach, or at least, help to check their quality control. Furthermore, it is possible to use the same digital model several times to simulate different displacement scenarios.

Therefore, the motivation behind the present thesis aims to extract necessary information from micro-CT and thin section images to build a reliable pore-scale model that can be used further for prediction of important reservoir properties (absolute and relative permeabilities are the most important ones). In addition, it can be interesting to estimate how good the results obtained by proposed numerical solutions match with available experimental measurements.

Figure 1.1 displays the major steps that have been done under the current research.

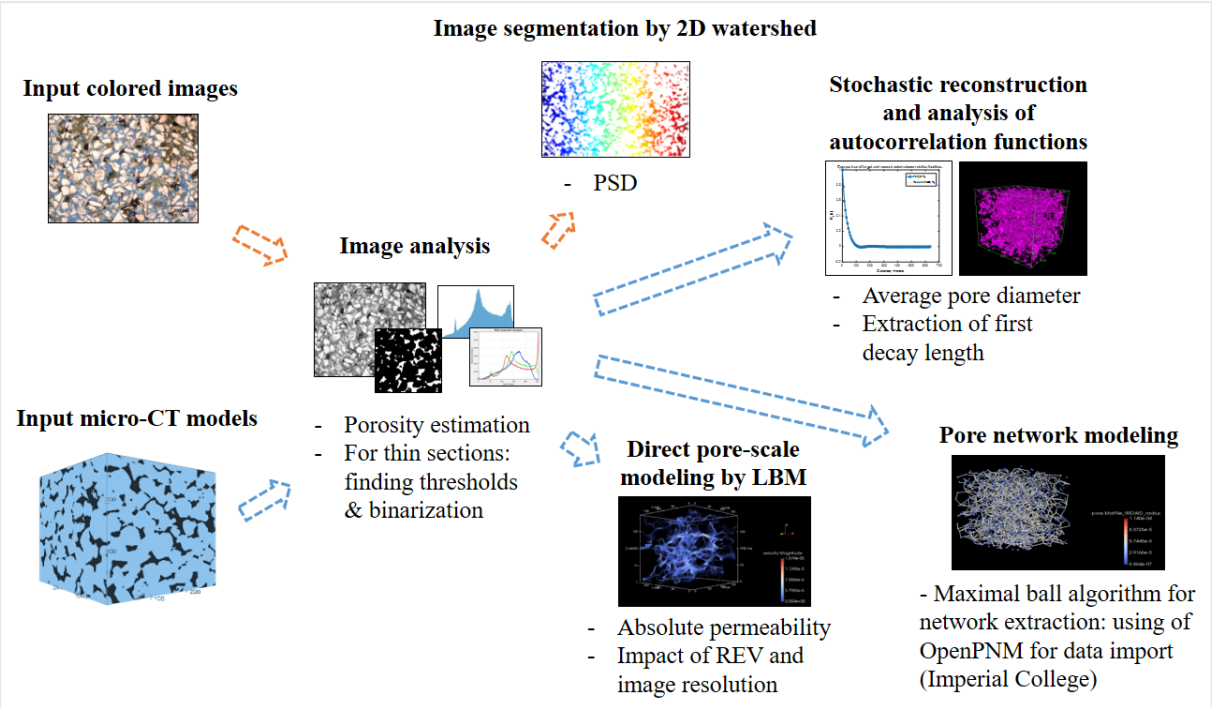


Figure 1.1. The workflow of the current research

Conceptually, the thesis is divided by several sections, and the outline starts out by presenting the major theoretical aspects in Chapters 2-5. Chapter 2 contains basics of image analysis (including brief introduction about imaging by means of thin section & SEM microscopy and micro-CT), as well as ways how to binarize input color images. The general information about reconstruction procedure with focus on the stochastic Gaussian random field method is discussed in Chapter 3. Chapter 4 presents the most important equations and theory behind the lattice Boltzmann method that is used for absolute permeability estimation. Finally, Chapter 5 summarizes theory about pore network models and the most commonly used network extraction procedures. The methodology and obtained results are presented in Chapter 6. Here, each subchapter is dedicated to one of the chapters with theory described above. Finally, the most important findings and possible suggestions for future work are stated in Chapter 7, which also contains discussion about all possible inaccuracies and bottlenecks of this research study, as well as compares obtained results with those from other literature.

Chapter 2.
Image Analysis

Image analysis involves the extraction of measurements from an image (Alshibli & Reed, 2010). Thin section and SEM images are 2D digital representatives of reservoir rocks. They are easily available, relatively cheap (e.g. thin sections obtained by optical microscope) and may contain significant amount of information about pore microstructure. Therefore, image analysis of thin section, SEM and micro-CT samples can provide a reliable information about geometric complexity of reservoir pore systems.

2.1. General information about imaging of porous media

Perhaps the most widespread method for visualization of the pore space is geological thin section analysis by optical microscopy. Thin section examination is a standard technique of analysis under a petrographic microscope for almost all rock types, since it can provide a valuable information about mineralogical composition, textural and structural features of rocks and soils. The best resolution that can be achieved by petrographic microscopy is limited by the wavelength of visible light, or so-called diffraction barrier, which is around 0.23 μm (Bultreys, et al., 2016) & (Nichols, 2009).

In thin sections, some grains may appear as dark brown or black due to their opacity. Opaque minerals do not allow any light to go through them even when thin slices (normally 30 μm thickness) had been cut. The most typical opaque minerals are oxides and sulphides, particularly iron oxides (hematite) and iron sulphides (pyrite). In these cases, it is important to highlight the pore space. Colored dye (e.g. blue epoxy vacuum impregnation) is often used to assist in the determination of porosity and to stabilize the material (Nichols, 2009).

Figure 2.1 indicates the typical examples of photomicrograph obtained by a petrographic microscope.

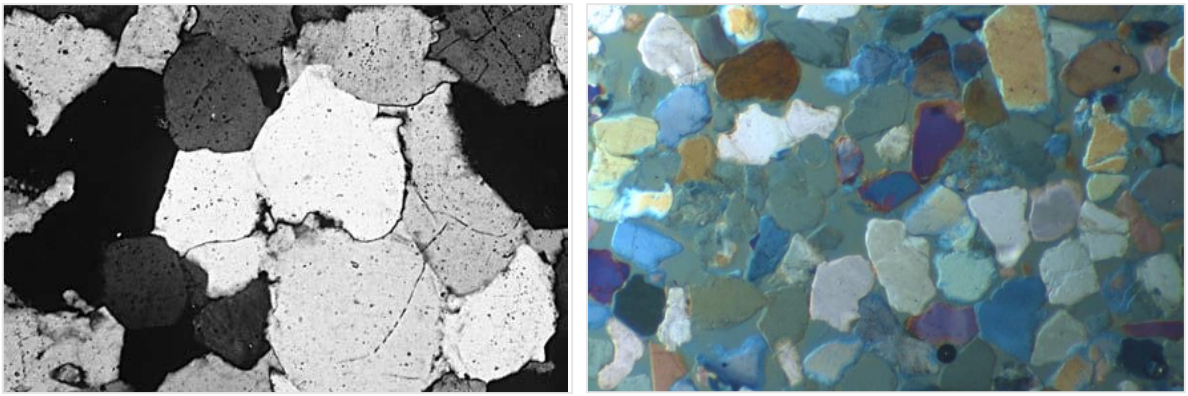


Figure 2.1. A photomicrograph of a sandstone (left) and a blue epoxied thin section of Reigate Silver Sand under cross-polarized light (right)
(taken from Nichols, 2009 and Alshibli & Reed, 2010 respectively)

If a higher resolution is required, scanning electron microscopy (SEM) can be used, since it provides resolution down to several nanometers. In general, SEM is quite similar to electron micro-probe procedure, but is primarily developed for imaging rather than for chemical analysis. Images are produced by scanning the sample with a high-energy electron beam (Figure 2.2, left). These electrons interact with atoms of the sample, producing signals that contain information about sample's surface topography and composition (Bultreys, et al., 2016).

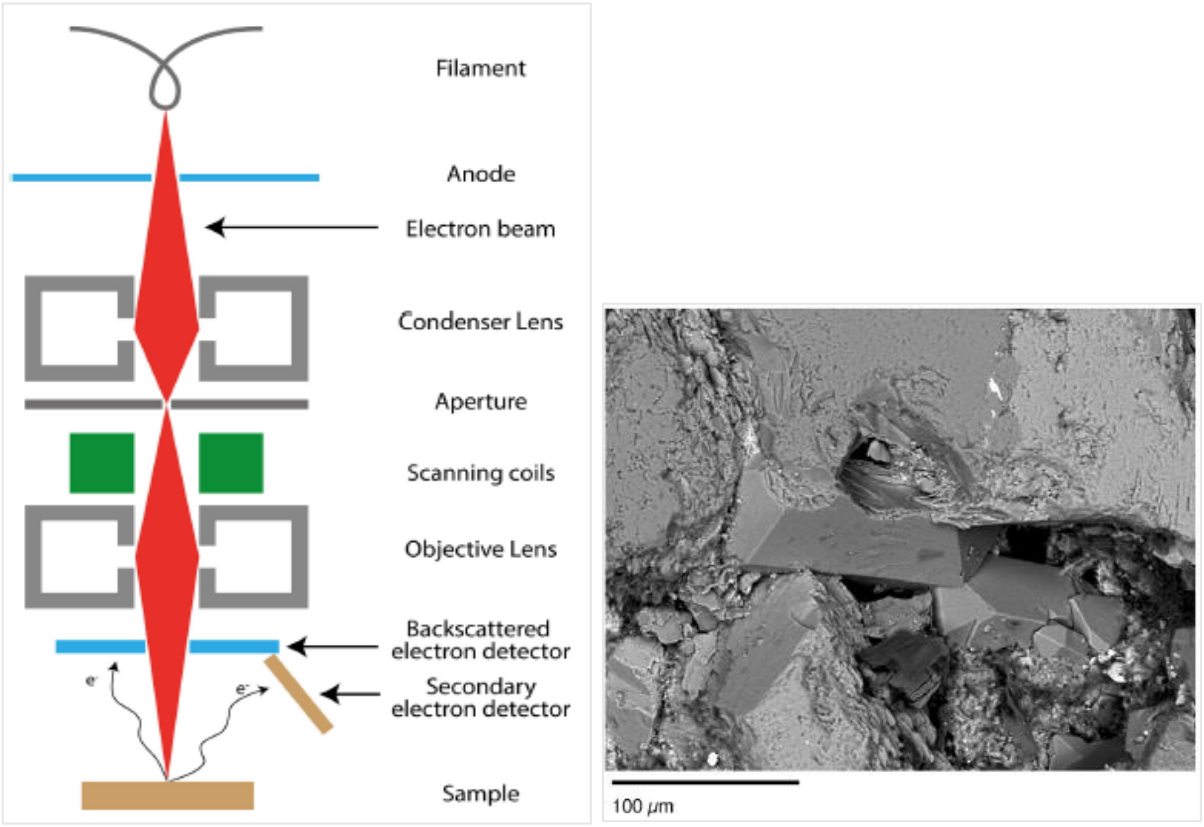


Figure 2.2. Schematic representation of a SEM (left); BSE image of Vosges sandstone (right)
(taken from Bultreys, et al., 2016)

One serious drawback of SEM imaging on geological materials is related to the necessity of samples' prior treatment, since most of geological samples are electrical insulators and require a conductive coating to avoid charging under electron bombardment.

There are several types of signal produced by a SEM, such as secondary electrons (SE), backscattered electrons (BSE) and characteristic X-rays. Probably the most common SEM approach is to detect secondary electrons that are emitted by atoms excited by the electron beam due to their low energy (as a result, very high resolution can be achieved). Backscattered electrons are electrons generated by the electron beam, but reflected from the sample by electron scattering. Although spatial resolution in SE images is higher than in BSE images due to smaller interaction volume, the latter are still considered as an important tool for analysis of composition and porosity of geological materials. The number of detected electrons depends on specimen topography. Therefore, by scanning and collecting electrons by a special detector, it can be possible to obtain an image displaying the topography of the created surface, as in Figure 2.2, right (Bultreys, et al., 2016).

Finally, X-ray computed tomography provides 3D structural information of porous media in a wide range – between submicron to millimeter scale. The micro-CT method is based on the attenuation of X-rays when they travel through materials with different absorbing capacity, expressed by the Beer-Lambert law:

$$I = I_0 e^{-\int \mu(s) ds} \tag{2.1}$$

which states that transmitted X-ray intensity I depends on the initial intensity I_0 and absorption coefficient $\mu(s)$ along the path of rays s.

This method is a non-destructive technique to examine the internal structure of objects by creating 3D images that map the variation of the X-ray attenuation coefficient values (so-called CT numbers). There are two main types of micro-CT devices that are commonly used in imaging of geological materials – desktop micro-CT scanner and synchrotron X-ray setup. The main differences between them are related to the resolution limit and rotational angle for a sample (360 deg. for desktop and 180 deg. for synchrotron, due to having two parallel beams in the latter case) (Dong, 2007) & (Bultreys, et al., 2016).

A typical X-ray CT setup includes stationary X-ray source and detector, and rotating sample, as it is shown in Figure 2.3. In addition, computers are used during micro-CT scanning procedure to control the scanning parameters and to monitor the reconstruction. The sample is

placed between source and detector, where the resulting radiograph is recording. The intensity of X-rays reaching detector, among other factors, depends on the sample's thickness, composition and density. At the time when the rotational angle reaches 360 deg. (or 180 deg. for synchrotron), several hundreds of radiographs are registered and further transformed into a 3D volume by applying appropriate reconstruction algorithm. This data volume represents a 3D distribution of the linear attenuation coefficient values which can be viewed as gray scales. After that, it is necessary to apply filtering to smooth, reduce noise and improve the contrast between pores and grains of the obtained image (Bultreys, et al., 2016).

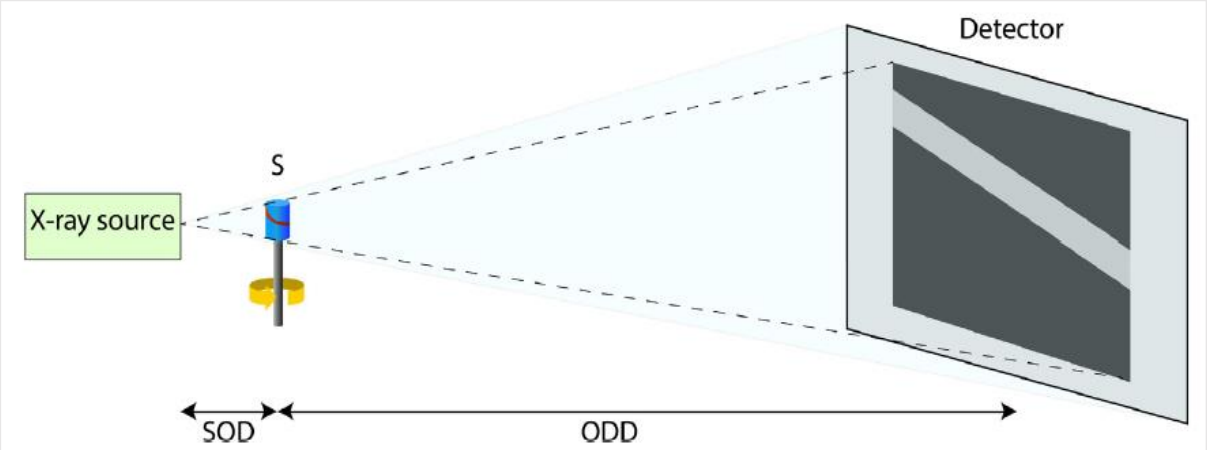


Figure 2.3. The schematic representation of basic components of a micro-CT scanner (taken from Bultreys, et al., 2016).
Here SOD denotes the distance between X-ray source and stage sample, and ODD – between stage sample and detector

Although most of the desktop scanners can typically provide a resolution between 2 and 5 microns (Sakellariou, et al., 2003), the best achievable image resolution is around 700 nm, and it has been obtained by using synchrotron micro-CT (Bultreys, et al., 2016).

2.2. Image binarization and porosity estimation

One of the most significant properties that can be determined during image analysis is porosity. Usually, to find porosity from thin sections, optical point count method can be used. Although this method provides accurate results, it can be difficult to automate porosity estimation in a case of a big amount of thin sections. In this case, the better approach is to define porosity from binary images (Richa, et al., 2006).

To represent binarized images the indicator binary function can be introduced:

$$Z(\mathbf{r}) = \begin{cases} 1 \text{ if } \mathbf{r} \text{ in pore space} \\ 0 \text{ otherwise} \end{cases} \tag{2.2}$$

For statistically homogeneous media, the simplest morphological measure is the volume fraction φ_1 of phase 1 (i.e. pore space), which is the one-point correlation function and can be written as follows:

$$\varphi_1 = \langle Z(\mathbf{r}) \rangle \tag{2.3}$$

It means that the volume fraction, or porosity, can be interpreted as the probability to find a point in phase 1. In a digitized medium, it is possible to find φ_1 by direct counting the number of pixels in phase 1, and then divide them over the total amount of pixels (Yeong & Torquato, 1998).

There are several approaches to convert colored thin section images to binary format, e.g. using thresholds or classifying grains and pore space by supervised or unsupervised learning techniques (Richa, et al., 2006). In this thesis, threshold method is implemented, since it provides a trustworthy and quick algorithm to obtain porosity.

2.3. Important image processing techniques and morphological operations

Very often, real image systems are imperfect and may contain some noise or artifacts that can make the image analysis a challenging task. For example, as it will be discussed later, input color thin sections contain some tiny dark spots that during binarization can be considered as an undesired noise. Furthermore, raw reconstructed images can also be noised due to the stochastic nature of the reconstruction procedure.

Therefore, in this subchapter the most important and widely used image processing techniques are described, including median filtering, erosion-dilation operations and opening procedure.

2.3.1. Median filtering

This filtering is often used to remove salt and pepper, as well as other types of noise. As an averaging filter, median filter sets the value of the output pixel to the average of the pixel values in the neighborhood around the corresponding input pixel. Due to the fact that median filtering

is less sensitive to extreme values and does not reduce the image sharpness, it typically uses more often than mean (average) filtering (MathWorks_Documentation, 2017).

Figure 2.4 shows an example of 3x3 median filter mask. In this case, nine pixel values are taken from the input image with further numerical sorting. Then, the median of this 1D list can be found, and this value is assigned to the appropriate pixel in the output image.

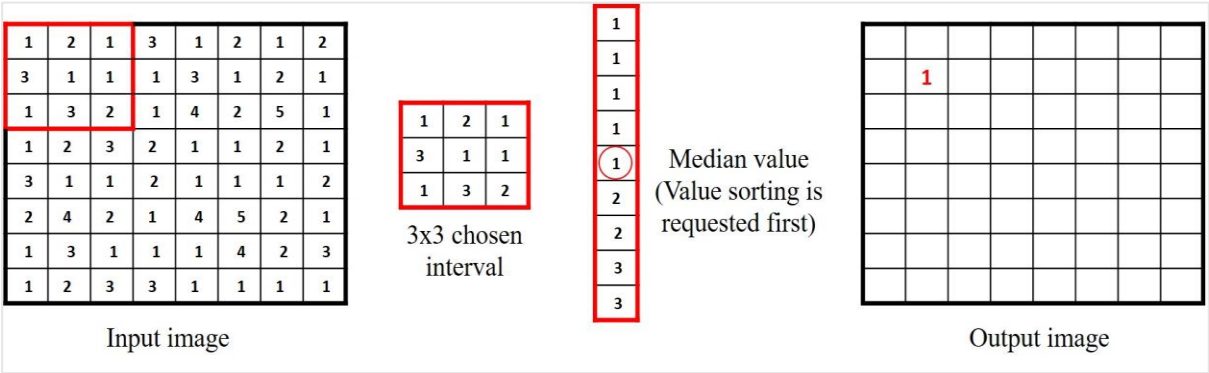


Figure 2.4. 3x3 median filter mask

2.3.2. Erosion-dilation morphological operations

Morphology offers a broad set of image processing operations based on shapes. In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors (MathWorks_Documentation, 2017).

The most important morphological operations are dilation and erosion. In general, dilation is defined by placing an imaginary sphere to the non-reference phase and determining space available for it as if the reference phase was impenetrable to this sphere. The space unavailable to the sphere is considered as a dilated reference phase. Erosion is an inversed procedure such that now reference phase is allowed to be penetrated by a virtual sphere (Guo, et al., 2014).

Figure 2.5 displays these two processes. In digital imaging, dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries.

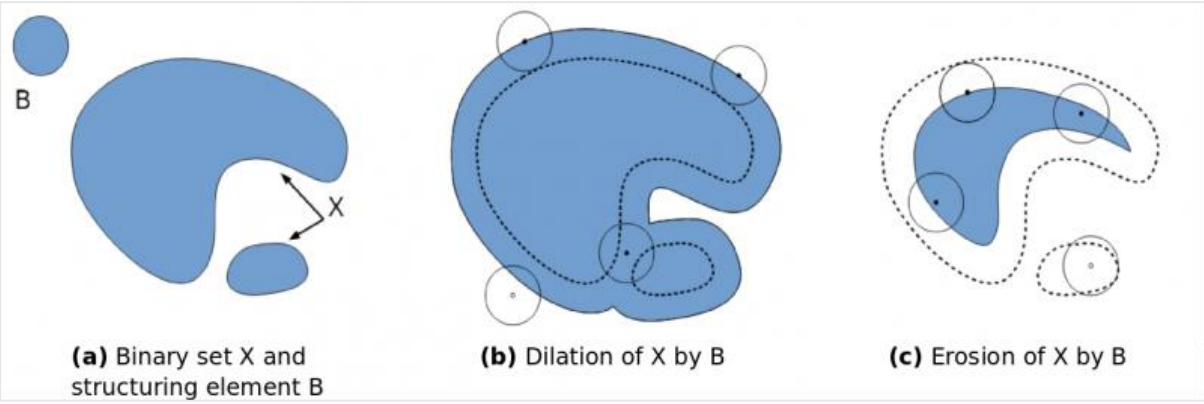


Figure 2.5. Principle of morphological dilation and erosion
(taken from ImageJ website – *www.imagej.net*)

The number of added or removed pixels depends on the size and shape of the structuring element used to process the image. In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbors in the input image (MathWorks_Documentation, 2017).

Dilation and erosion operations are commonly used in pair, and this coupled method is called the opening procedure. It allows to improve significantly the quality of the reconstructed images and can be especially helpful for a topologically complex microstructure when it is challenging to capture the important structural features using a limited number of statistical correlations. In this work, it was used after finishing the reconstruction procedure to improve the connectivity within obtained reconstructed images.

Chapter 3.
Stochastic Reconstruction of Pore Structure

As it has been referred earlier, in some cases when the submicron structure of porous media occurs, microtomography may not be reliable due to resolution limits. Furthermore, in some cases there is no technical possibility to use the direct μ-CT imaging, as Figure 3.1 displays.

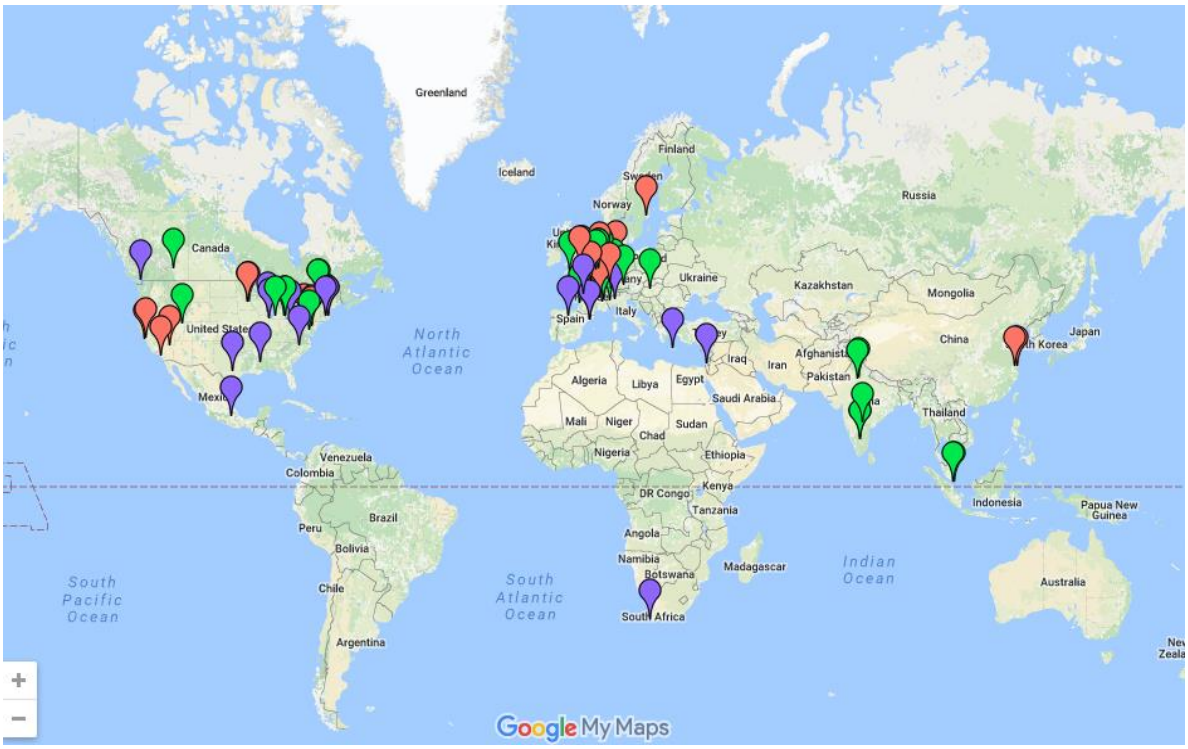


Figure 3.1. Map showing a global distribution of micro-CT scanners
(taken from *www.microctworld.net*).
Here different marker colors are related to different sources, incl. universities and research institutes marked in purple

Although this map is not the most updated one, since it does not show presence of CT-scanners in Australia (new ANU laboratory), in Norway (FEI company, Trondheim) or in Russia (Schlumberger research center in Moscow), it clearly indicates that there are a lot of countries where X-ray computed micro-tomography analysis is simply unavailable.

That is why, the reconstruction of 3D porous media is still of a great interest among many researchers and will be discussed in more details below.

3.1. General information and review of existing methods

There is a great amount of techniques for reconstructing 3D porous media from 2D thin section images (the latter are in binary format). E.g. several statistical methods have been developed – typically, they include measurements of statistical properties using 2D images. Then, random 3D models can be generated.

As follows from Equation 2.3, porosity is a one-point probability function for a void phase, and it reflects the probability of chosen point to be inside the considered phase. Another important characteristic is a two-point correlation function which defines the probability of finding two points separated by some distance within the same phase. It is typically used to assess the sample's heterogeneity and representative volume size (e.g. through the characteristic length as it will be shown later), since it contains an important information about the typical feature size of image objects and depends on both rock particles and porels between them (ImageJ_webpage, 2015).

Moreover, the form of a two-point correlation function provides an additional information about a type of particle systems, as Figure 3.2 indicates.

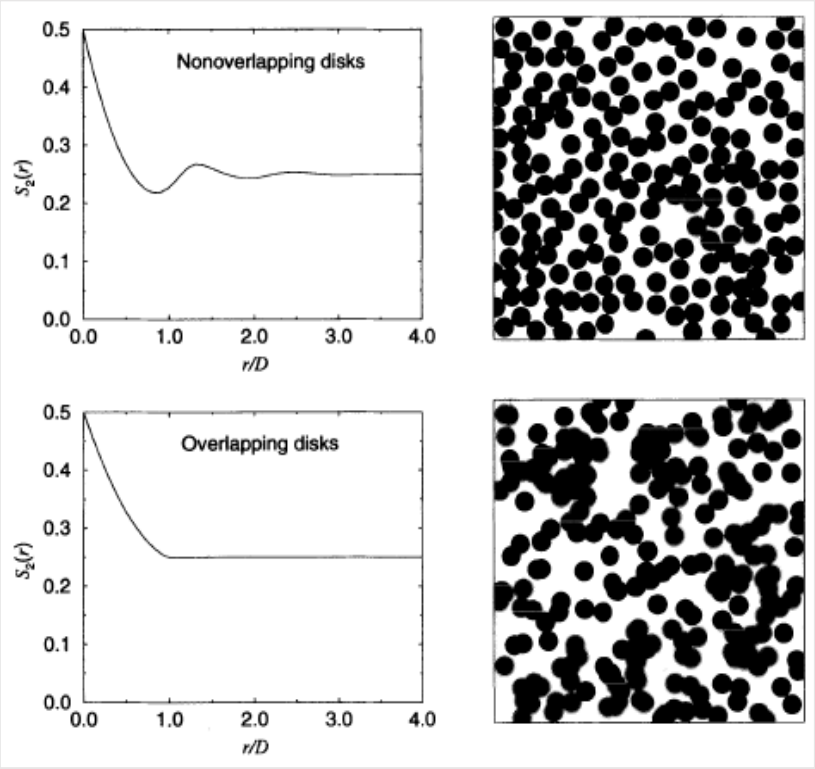


Figure 3.2. Two-point probability function of void phase for two different systems (taken from Torquato, 2002)

As follows from Figure 3.2, in a case of nonoverlapping disks, the two-point correlation function shows oscillations for small r (short-range order) with periodicity approximately equal to the particle diameter D. In a case of overlapping disks, however, it displays no short-range order oscillation, but rather demonstrates a monotonic decay to the asymptotic value of φ^2 at $r = D$.

Nevertheless, to reproduce a high degree interconnectivity, two-point statistical method may not be sufficient. In this case, so called multi-point statistical approach developed by Okabe and Blunt can be implemented (Okabe & Blunt, 2005). Very often, to reconstruct pore scale structures, binarized thin section images are used as training images that contain only void or solid phase. The training image then is scanned using a special template, and each occurrence of any possible pattern of void and solid phases is reported. Generally, the key mechanism of the method deals with the finding of local conditional probability distribution functions. The template includes a center location and several others which can be found as:

$$\mathbf{u}_\alpha = \mathbf{u} + \mathbf{h}_\alpha, \alpha = 1, ..., n \tag{3.1}$$

where $\mathbf{h}_\alpha$ are vectors describing the template, $\mathbf{u}$ is a central location and $\mathbf{u}_\alpha$ are other available locations within the template.

Figure 3.3 shows an example of training 2D thin section image and a 9x9 template used to capture multiple-point statistics. In order to have a better reconstruction, however, various template sizes n and geometries are recommended to try.

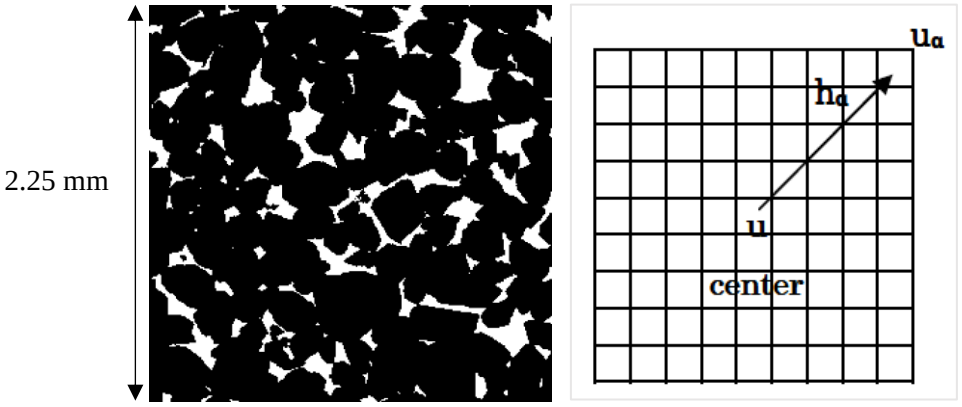


Figure 3.3. One of training images from 3D micro-CT model (left); a 9x9 template used to analyze multi-point statistics
(taken from Okabe & Blunt, 2005)

Since 2D images were used to measure the multi-point statistics, to generate a whole 3D cube, it is necessary to propagate statistics obtained on the *XY* plane to the *YZ* and *XZ* planes assuming

the sample's isotropy in orthogonal directions (Okabe & Blunt, 2005). If strong anisotropy of sample takes place, this assumption can be a serious limitation of the proposed method.

Another possible option to avoid the lack of matching parameters is to implement a simulated annealing method, when additional correlated functions are used as constrains (e.g. chord distribution function, void-phase lineal path function, etc.). It starts from an initial random binary configuration satisfying the volume fraction. It is necessary then to transform this high-energy state into a state with the minimum of 'energy'. Here, 'energy' measures the difference in correlation function, two-point cluster function, lineal path function, etc. between the reference and the simulated image:

$$E = \sum_k \sum_i [f_{ref}^{(k)}(r_i) - f_{sim}^{(k)}(r_i)]^2 \quad \text{when} \quad E \to \min \tag{3.2}$$

In Equation 3.2 i indicates counting on the number of using statistical functions as constrains, while k is used for counting number of iterations.

At each iteration step, the interchange of two randomly chosen pixels occurs. If energy decreases after this pixel rearrangement, a new configuration is accepted as a suitable one. However, if energy increases after the exchange, this configuration can be considered as an intermediate one, and the exchange is accepted with a certain probability, which depends on the energy difference between current and previous iterations and the value of annealing temperature:

$$p_a^{(k)} = \begin{cases} 1, \text{ if } \Delta E \le 0 \\ \exp\left(-\dfrac{\Delta E}{T^{(k)}}\right), \text{ if } \Delta E > 0 \end{cases} \tag{3.3}$$

Figure 3.4 demonstrates how this method works. Here, the reconstruction of glass sphere aggregate is shown (pores are indicated in black). Two constrain functions are used – the pore-pore autocorrelation function and the solid-phase chord distribution function.

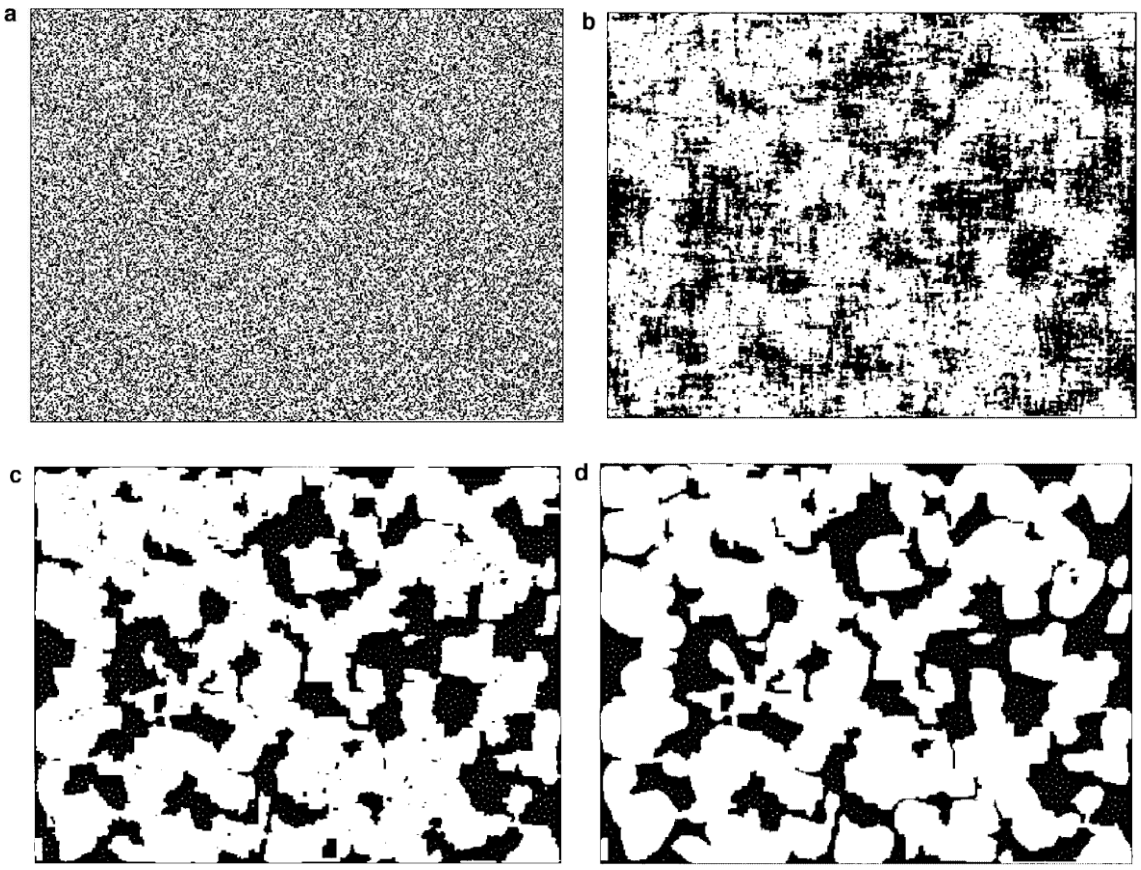


Figure 3.4. Simulated annealing reconstruction method
(taken from Talukdar, et al., 2002).
Here: (a) Initial random structure (E=36.31); (b) Intermediate structure (E=18.00); (c) Intermediate structure (E=1.00); (d) Final structure ($E=10^{-4}$)

Serious drawback of simulated annealing method is related to its computational efficiency. Due to the random interchange of pixels, it can require a lot of time to get the final low-energy state. As many researchers emphasized, the main computational cost in the simulated annealing technique occurs due to the repeated computations of the reconstructed statistical functions after each iteration. Furthermore, in a case of a multiple sampling, the required computational time should increase proportionally to the number of used samples. Therefore, to overcome mentioned method's shortcomings, several modifications have been suggested (e.g. hybrid tabu search-simulated annealing method, etc.).

Finally, process-based method can be considered as more advanced reconstruction technique, since it considers how different geological processes affect the pore structure and the permeability of sedimentary rocks. The approach suggested by Øren and Bakke is to build sandstone models that are analogs of real sandstones by stochastic modeling of the main sandstone forming processes, like sedimentation, compaction, and diagenesis. For example, to model compaction the shift of vertical coordinates of every grain center to the center of

considered plane should take place with a choice of a required compaction factor λ_z and a random variance that imitates grain rearrangement. To mimic compaction of the Berea sandstone, Øren and Bakke used $\lambda_z = 0.05$ and randomly distributed ε_z in the interval $[-0.02; 0.02]$. At last, to simulate diagenesis processes the radius of the originally deposited grain should be modified in order to mimic quartz cement growth or dissolution. Also, it is possible to control a direction of cement growth (in the direction of pore bodies or pore throats) (Øren & Bakke, 2003).

The input properties for the process-based modeling are taken from backscattered electron (BSE) images of 2D thin sections represented as a gray-scale values array (Figure 3.5).

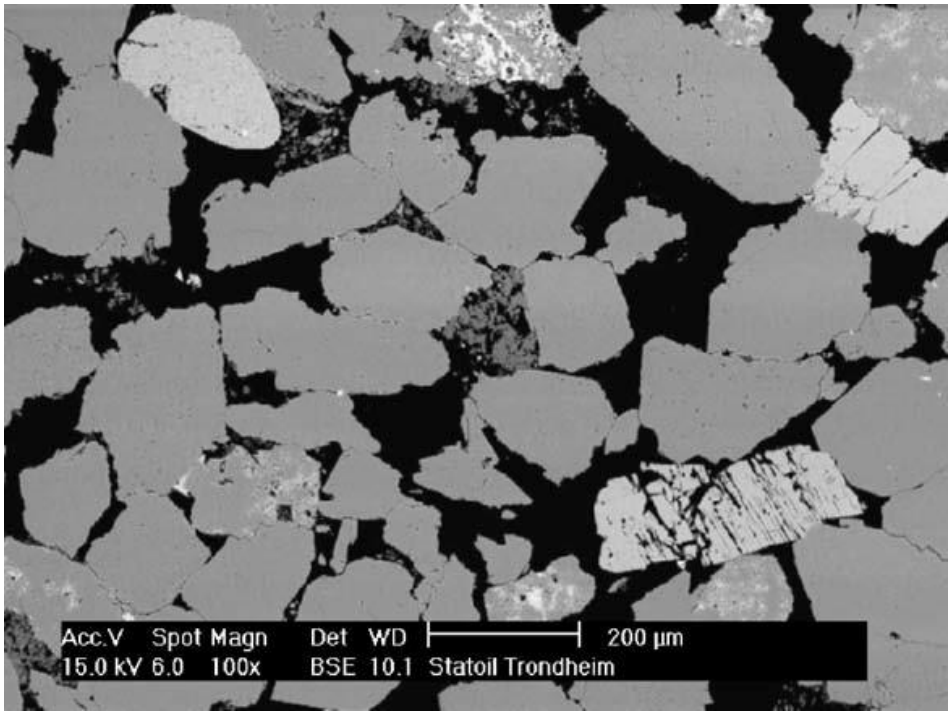


Figure 3.5. BSE thin section image of Berea sandstone
(taken from Øren & Bakke, 2003)

By applying a thresholding procedure, it is possible to distinguish different phases within the given image, e.g. in Figure 3.5, pore space is presented in black, feldspar – in light gray, quartz – in medium gray, and clays are dark gray.

However, in this thesis, the two-point statistical method based on the Gaussian random field reconstruction has been used due to its simplicity and easiness to code it. As it was discussed in Chapter 2, the opening procedure can be applied after the reconstruction procedure to

improve its quality and connectivity. More details about how to implement this method in MATLAB will be presented in Chapter 6.

Several options will suggest to check in which extent the geometrical properties of reconstructed media are matching with the original cases. As many researchers point out, one of the most important criterion for estimation of stochastic reconstruction is its ability to reproduce the connectivity of the original pore structures (Manwart, et al., 2000). Another important factor is how good reconstructed samples can capture the macroscopic parameters of the original materials, e.g. permeability.

Therefore, the visual inspection of real and reconstructed cases is implemented, first, following by comparison of transport properties of these porous media (absolute permeability and formation factors), as well as cluster analysis.

3.2. Gaussian random field reconstruction method

Due to the random nature of phase distribution, in a probabilistic context, it is possible to consider the indicator function in Equation 2.2 as a binary random field, $\mathrm{y}(\mathbf{r},\omega)$ where ω characterizes a basic random event (Feng, et al., 2014). The difference between random process and random field is in dimensionality: random process can be considered as a 1D-case of the random field, while the latter is related to 2D or 3D space.

It is a common practice to describe random fields as statistically homogeneous, or stationary up to the second order. Stationarity means that field's mean and variance are invariant when shifted in space. Isotropy of the random field means that in this case the correlation function is a function of the distance only, and can be extracted from 2D images, i.e.:

$$\mathrm{g(r)} = \langle y(\mathbf{r}_1) y(\mathbf{r}_2) \rangle \tag{3.4}$$

As it was mentioned earlier, in porosity analysis, a two-point correlation function can also be used:

$$S_2(r) = \mathrm{g(r)} \cdot (\varphi - \varphi^2) + \varphi^2 \tag{3.5}$$

It is clear that $S_2(r) = 1$ when $r = 0$, and $S_2 = \varphi^2$ when $g(r) = 0$ (absence of correlation).

Another typical assumption of $y(\mathbf{r})$ is its ergodicity, i.e. if a system is ergodic, it is possible to observe a few elements of it for a long period, or many elements during a short time, and the results in both cases should be the same.

A random field is called a Gaussian random field (GRF) if it involves Gaussian probability density functions of the variables. When $y(\mathbf{r})$ is a GRF, its joint probability density is given by:

$$P_n(\mathbf{y}) = \frac{1}{\sqrt{(2\pi)^n \det \mathbf{G}}} \exp\left(-\frac{1}{2}\mathbf{y}^T \mathbf{G}^{-1}\mathbf{y}\right) \tag{3.6}$$

where $\mathbf{y} = [y(\mathbf{r}_1), ..., y(\mathbf{r}_n)]^T$ and the elements of $\mathbf{G}$ are given by field-field correlation function $g_{ij} = g(r_{ij}) = \langle y(\mathbf{r}_i) y(\mathbf{r}_j) \rangle$ (the last equation is identical to Equation 3.4).

Note: It is feasible to assume without loss of generality that $y(\mathbf{r})$ has zero mean and unit standard deviation (as it has been done in Equation 3.6).

There are two main methods to generate isotropic GRF. The first one uses Fourier summation and creates a random field in a cube with a side length T:

$$y(\mathbf{r}) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} \sum_{k=-N}^{N} c_{ijk} e^{i\mathbf{k}_{ijk} \cdot \mathbf{r}} \tag{3.7}$$

where $\mathbf{k}_{ijk} = \frac{2\pi}{T} \cdot (i\mathbf{i}_e + j\mathbf{j}_e + k\mathbf{k}_e)$. In this case, the statistics of the field are set by the random variables $c_{ijk} = a_{ijk} + ib_{ijk}$.

Alternatively, the random field can be considered as a superposition of waves with wave vectors' magnitude k_i, and phase ϕ_i in a range $[0; 2\pi]$:

$$y(\mathbf{r}) = \sqrt{\frac{2}{N}} \sum_{i=1}^{N} \cos(k_i \hat{\mathbf{k}}_i \cdot \mathbf{r} + \phi_i) \tag{3.8}$$

If the Gaussian field can be presented as a sum of planes or waves, then the GRF is completely described by its power spectral density (PSD) function $\rho(k)$, or according to the Wiener-Khinchin theorem, by its autocorrelation function which can be defined for the isotropic radial function as follows (Roberts, 1997):

$$g(r) = \langle y(\mathbf{r}_1) y(\mathbf{r}_2) \rangle = \int_0^\infty 4\pi k^2 \rho(k) \frac{\sin kr}{kr} dk \tag{3.9}$$

The general form of analytical correlation function that is used in this work:

$$g(r) = \frac{e^{-\frac{r}{\xi}} - \frac{r_c}{\xi} e^{-\frac{r}{r_c}}}{1 - \frac{r_c}{\xi}} \frac{\sin \frac{2\pi r}{d}}{\frac{2\pi r}{d}} \tag{3.10}$$

where ξ is a correlation length, d is a domain scale and r_c – a cut-off scale (Roberts, 1997).

The function in Equation 3.10 has the following Fourier transform (can be obtained from Equation 3.9 by inserting Formula 3.10 and inverting):

$$\rho(k) = \frac{\pi^{-2}(\xi - r_c)^{-1} \xi^4 d^4}{[d^2 + \xi^2 (kd - 2\pi)^2][d^2 + \xi^2 (kd + 2\pi)^2]} - \frac{\pi^{-2}(\xi - r_c)^{-1} r_c^4 d^4}{[d^2 + r_c^2 (kd - 2\pi)^2][d^2 + r_c^2 (kd + 2\pi)^2]} \tag{3.11}$$

here k is a frequency (Roberts, 1997).

Some authors emphasize the direct connection between the average pore radius (diameter) and the correlation length ξ, which can be described by the following expression (Bowen & Tanner, 2006):

$$\langle D \rangle = \frac{\xi}{(1 - \varphi)} \tag{3.12}$$

where $\langle D \rangle$ is the average pore diameter, ξ – the correlation length and φ is porosity.

It means, if the correlation length is known, one may calculate the pore radius and compare with the value obtained by other methods, e.g. by watershed segmentation to check the results consistency. Furthermore, there is a connection of cut-off parameter with pore connectivity, i.e. high cut-off value indicates a case when only few pores are connected, and vice versa – low r_c means better connection between pores within the sample (Mooney & Korošak, 2009). Finally, domain scale value d can characterize the minimum size of lag that can be chosen to build the representative partial correlation function in a case of having too big image samples.

Stationary Gaussian fields are fully defined by two first statistical moments – one-point correlation function, or porosity from Equation 2.3, and two-point autocorrelation function of Formula 3.4. This fact has a contradictory effect, since using only two constrains greatly simplifies the reconstructed procedure. On the other hand, in several cases with complex

structured and continuous patterns, using only the void fraction (porosity) and void-phase autocorrelation function of the target medium as reconstruction criteria can be not enough and more statistical information is required.

To reconstruct microstructure, the one-cut GRF is used in the current study. The level-cut parameter α can be obtained by solving the following equation:

$$\varphi_1 = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\alpha} \mathrm{e}^{-\frac{t^2}{2}}\, dt \equiv \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} \mathrm{e}^{-\frac{t^2}{2}}\, dt \tag{3.13}$$

Remember that the complementary error function can be defined in the following manner,

$$erfc(x) = 1 - erf(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2} dt \tag{3.14}$$

it is possible then to rearrange Equation 3.13 and to express α :

$$\varphi_1 = \frac{1}{2} \cdot \left(1 - erf\left(\frac{\alpha}{\sqrt{2}}\right)\right) \Rightarrow \frac{\alpha}{\sqrt{2}} = erfinv(1-2\varphi_1) \Rightarrow \alpha = \sqrt{2}\, erfinv(1-2\varphi_1) \tag{3.15}$$

where *erfinv* is the inverse error function so that $erfinv(erf(x)) = x$, and φ_1 is a volume fraction of pore space, or porosity, as in Equation 2.3.

Considering one-cut level parameter, it is possible to rewrite Equation 2.2 in the following manner (Yuan, et al., 2010):

$$Z(\mathbf{r}) = \begin{cases} 1, \ \mathrm{y}(\mathbf{r}) \geq \alpha \\ 0, \ \mathrm{y}(\mathbf{r}) < \alpha \end{cases} \tag{3.16}$$

Three parameters (ξ, d, r_c) can be determined by a best-fit procedure, which minimizes the least-square errors between target and reconstructed functions:

$$E = \sum [p_{\mathrm{fit}}(r) - p_{\mathrm{expt}}(r)]^2 \tag{3.17}$$

where p_{expt} is experimentally measured two-point correlation function (or 'target correlation function'). In this work, it is found in ImageJ open-source platform. The correlation function of reconstructed images (p_{fit}) is assumed to be of the general form given by Equation 3.10.

Once parameters have been obtained, the reconstruction can be generated. Probably, the most efficient method is to use a fast Fourier transform, since it can be implemented in MATLAB (Yuan, 2007). The discrete realizations of the random field $\mathrm{y}(\mathbf{r})$ in a cubic domain $\mathrm{V} = \mathrm{s}_1^3$,

where s_1 is a side length of input images, can be generated by 3D inverse FFT, taking into account Formula 3.7:

$$y_{LMN} = \frac{1}{T^3}\sum_{i=0}^{T-1}\sum_{j=0}^{T-1}\sum_{k=0}^{T-1} Y_{m_1 m_2 m_3} \exp\left(\frac{2\pi i}{N}(m_1 \mathrm{L} + m_2 \mathrm{M} + m_3 \mathrm{N})\right) \tag{3.18}$$

where T is the number of sampling points in each direction (assume that it is equal to the side length, since a single sample interval has been chosen), and $Y_{m_1 m_2 m_3}$ are random numbers defined as follows, since typically the Fourier image is considered as a complex function (Yuan, et al., 2010):

$$Y_{m_1 m_2 m_3} = \left(G^{real}_{m_1 m_2 m_3} + i \cdot G^{imag}_{m_1 m_2 m_3}\right) \cdot \sqrt{\frac{V\rho(k)}{2}\left(\frac{\sqrt{M_1^2 + M_2^2 + M_3^2}}{s_1}\right)} \tag{3.19}$$

In Equation 3.19 $\rho(k)$ is a power spectral density function described by Equation 3.11, $G^{imag}_{m_1 m_2 m_3}, G^{real}_{m_1 m_2 m_3}$ are independent Gaussian distributed random variables, and $\mathrm{M}_1, \mathrm{M}_2, \mathrm{M}_3$ can be found using the following scheme (Yuan, et al., 2010):

$$M_j = \begin{cases} m_j, & m_j \le \dfrac{T}{2} \\ T - m_j, & m_j > \dfrac{T}{2} \end{cases}, \; j = 1,2,3 \tag{3.20}$$

It is worth noting that $G^{imag}_{m_1 m_2 m_3}, G^{real}_{m_1 m_2 m_3}$ are characterized by zero mean and unit variance.

Condition of having real y_{LMN} requires considering several possible cases. Furthermore, adding the frequency threshold can help to avoid the use of high frequencies, to improve smoothness of the obtained reconstructed images, as well as to decrease the runtime (Yuan, 2007).

Chapter 4.
Lattice Boltzmann Method for Permeability Estimation

Direct pore scale modeling methods are currently considered as a standard tool to calculate single-phase flow and transport properties, although the obtained real structure of the reconstructed porous media can be very complex. Among the most widely used methods for numerical flow simulations is the lattice Boltzmann method (LBM) that can be applied directly to the real pore structure without simplifying it. This is a great advantage of the proposed method, since there is no need to build approximated pore models to calculate absolute permeability.

4.1. Theoretical background about LBM method

4.1.1. Mesoscopic theory: from Boltzmann equation to the Navier-Stokes equations

Before discussing theory behind the LBM, it is important to mention why this method is used for flow simulations in the porous materials.

Actually, everything starts from the Navier-Stokes equations that govern the motion of fluids and are considered as the fundamental in fluid flow modeling. They can be interpreted as Newton's second law of motion for fluids. In a case of an incompressible Newtonian fluid, it is possible to write this as follows (Comsol_webpage, 2017):

$$\overbrace{\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)}^{\text{inertial forces}} = \underbrace{-\nabla p}_{\text{pressure gradient}} + \overbrace{\mu\nabla^2\mathbf{u}}^{\text{viscosity forces}} + \underbrace{\mathbf{F}}_{\text{external forces}} \tag{4.1}$$

where ρ is the fluid density, $\mathbf{u}$ is the fluid velocity, μ – fluid dynamic viscosity.

Very often, Equations. 4.1 are solved together with the continuity equation:

$$\frac{\partial \rho}{\partial t}+\nabla\cdot(\rho\mathbf{u})=0 \tag{4.2}$$

Since Equations. 4.1 represent the conservation of momentum, while Equation 4.2 introduces mass conservation.

Solving the Navier-Stokes equations for a particular set of boundary conditions, one may obtain the distribution of fluid velocity and its pressure for a considered geometry. It should be noted that due to computational complexity, these equations can be directly solved only for few analytical cases (e.g. flow in circular pipe, flow between two parallel plates, etc.). However, they still need to be solved for geometries that are more complex (Comsol_webpage, 2017).

One way to do that is to find potential alternatives, i.e. to solve them numerically or to use simplified concepts. Although the Boltzmann equation is not suitable for dense fluids (i.e. non-gases), it becomes possible to substitute the microscopic equations for liquids by microscopic equations for gases while keeping the macroscopic level (i.e. the Navier-Stokes equations) properly described (Sukop & Thorne, 2006).

There are many excellent literature resources that describe the lattice Boltzmann method in details, e.g. (Succi, 2001), (Krüger, et al., 2017), (Sukop & Thorne, 2006), etc. Therefore, in the current thesis only few words about its theory are mentioned.

The lattice Boltzmann method is a mesoscopic scheme of solving a discretized Boltzmann equation of fluid particle distributions that move and interact on a regular lattice pattern. Particle interaction is characterized by limited degrees of freedom, since the number of vectors defined for a particle distribution at each node is restricted by only allowing particles to move to a neighboring node. At each time step, these particles propagate to the neighboring lattice according to their velocity and initial position, and then collide with each other, while conserving momentum.

The lattice can be considered as the voxel grid of the 3D image obtained, for example, by means of microtomography, and therefore no complicated meshing is required. Thus, during LBM simulations, it can be very practical to use the same number of cells as the pixel size of micro-CT images, i.e. when one lattice cell is equal to a single pixel.

In this work, D3Q19 scheme is implemented for LBM simulations. It describes motion in 3D with 18 links representing 19 possible velocity distributions (incl. zero velocity), as Figure 4.1 indicates.

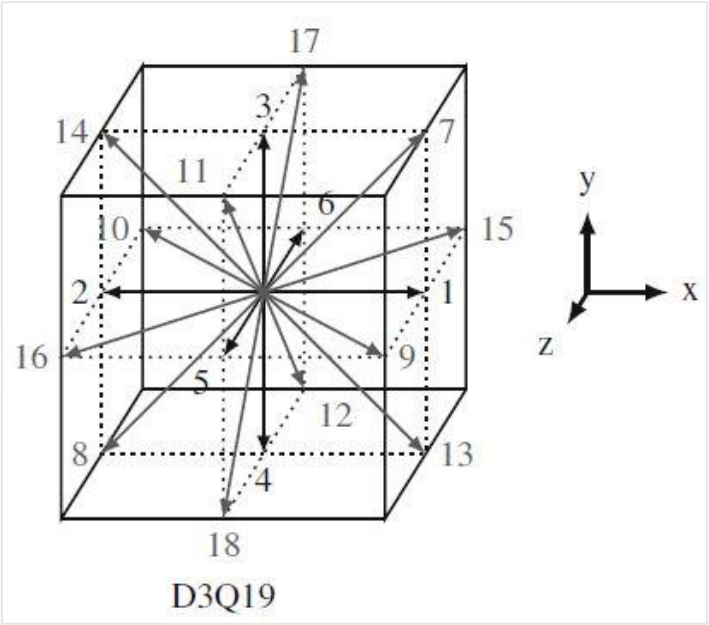


Figure 4.1. Diagram showing the velocity discretization in D3Q19 lattice scheme (taken from Krüger, et al., 2017)

Discretized Boltzmann equation is described by the following equation:

$$f_i(\mathbf{x}+\mathbf{c}_i\delta t, t+\delta t) - f_i(\mathbf{x},t) = \Omega_i(f_i(\mathbf{x},t)) \tag{4.3}$$

where $f_i(\mathbf{x},t)$ is the particle distribution function at location x and time t along the ith direction, $\Omega_i(f_i(\mathbf{x},t))$ is the collision operator, and $\mathbf{c}_i$ is local particle velocity (on the D3Q19 lattice $i = 0,1,...,18$).

Furthermore, $\mathbf{c}_i$ is proportional to a constant lattice velocity $c = \frac{\delta x}{\delta t}$, where δx and δt are the lattice grid spacing and time step respectively (Guiet, et al., 2011).

Once again, the 'mesoscopy' of this method means that by considering interactions between particles on the microscopic scale, it can be possible to reproduce their macroscopic behaviors in a correct manner.

4.1.2. The lattice-BGK method

The simplest case takes place when the collision operator in Equation 4.3 is defined as the Bhatnagar-Gross-Krook operator (Guiet, et al., 2011):

$$\Omega_{i(BGK)} = -\tau^{-1} \cdot \left(f_i(\mathbf{x},t) - f_{i(eq)}(\mathbf{x},t) \right) \tag{4.4}$$

where τ is the relaxation time and $f_{i(eq)}$ – the local equilibrium state.

Relaxation time τ is directly related to the kinematic viscosity as follows:

$$\nu = \frac{\delta t \cdot c^2}{3} \cdot \left(\tau - \frac{1}{2} \right) \tag{4.5}$$

From Equation 4.5 it is clear that in order to provide numerical stability by keeping viscosity positive $\tau > \frac{1}{2}$ should take place.

Although BGK collision scheme is a very simple, it has one serious drawback namely almost linear increase of permeability with increase of relaxation frequencies, which is unphysical. However, this effect disappears when $\tau = 1$ is chosen. Therefore, using a single relaxation time in BGK operator allows to avoid more complicated collisional models, e.g. the multiple relaxation time model (Krüger, et al., 2017).

The equilibrium distribution $f_{i(eq)}$ appearing in Formula 4.4 corresponds to an ideal state to which the particle distribution functions tend to a specific macroscopic state. This is derived from a Maxwellian distribution function and is expressed in terms of macroscopic flow parameters $(\rho, \mathbf{u})$ under a low-Mach assumption to ensure fluid incompressibility (Guiet, et al., 2011):

$$f_{i(eq)}(\rho, \mathbf{u}) = \rho w_i \left(1 + \frac{3}{c^2} \mathbf{c}_i \cdot \mathbf{u} + \frac{9}{2c^4} (\mathbf{c}_i \cdot \mathbf{u})^2 - \frac{3}{2c^2} \mathbf{u}^2 \right) \tag{4.6}$$

here w_i are weighting parameters that depend on the chosen lattice scheme, other parameters have been described earlier.

For D3Q19 lattice scheme they are defined as follows:

$$w_i = \begin{cases} 1/3 & \text{for i=0} \\ 1/18 & \text{for i=1,...,6} \\ 1/36 & \text{for i=7,...,18} \end{cases} \tag{4.7}$$

Finally, at each time step, from the distribution functions at a mesoscopic level, the macroscopic flow characteristics can be approximated in the following manner (Guiet, et al., 2011):

$$\rho = \sum_i f_i(\mathbf{x},t) = \sum_i f_{i(eq)}(\mathbf{x},t)$$
$$\rho\mathbf{u} = \sum_i \frac{f_i(\mathbf{x},t)}{c^2}\mathbf{c}_i = \sum_i \frac{f_{i(eq)}(\mathbf{x},t)}{c^2}\mathbf{c}_i \quad (4.8)$$

The set of Equations. 4.4-4.8 along with specific boundary conditions allows to simulate fluid flow in the lattice Boltzmann framework. Very often, for simplicity $\delta x = \delta t$ is chosen, since it enables to have $c = 1$ (Guiet, et al., 2011).

Therefore, one may conclude that the Navier-Stokes equations can be recovered from lattice Boltzmann scheme in the incompressible limit (Guiet, et al., 2011).

4.2. LBM implementation in PALABOS

It is possible to use the open source software PALABOS (**PA**rallel **LA**ttice **BO**ltzmann **S**olver) to carry out permeability calculations. In the current study, a single-phase fluid flow has been simulated.

4.2.1. LBM application to calculate absolute permeability

In general, permeability can be calculated according to the empirical Darcy's law, which for a single-phase 1D flow is defined as follows:

$$k = \frac{\mu\langle u\rangle}{\partial p / \partial x} \quad (4.9)$$

where $\langle u\rangle$ is the average velocity, μ – dynamic viscosity, $\partial p / \partial x$ – pressure gradient.

Application of the Darcy's law, or continuum modeling, becomes possible after averaging macroscopic parameters obtained from the direct pore-scale simulations.

It is worth noting that the use of Equation 4.9 requires adherence of several important assumptions about fluid flow in porous media. In particular, having a laminar (i.e. slow, low-Reynolds flow), single-phase and 1D horizontal flow.

Flow laminarity means that it is possible to neglect inertial forces compared with viscous ones in Equations. 4.1. Assuming also absence of external forces ($\mathbf{F} = 0$), the following expression is obtained:

$$0 = -\nabla p + \mu \nabla^2 \mathbf{u} \tag{4.10}$$

Furthermore, an additional source term must be added to Equation 4.10. From mechanical point of view, this term is a drag force due to the viscous friction of fluid with solid walls (Soulaine, 2017). Taking this fact into account and using an averaged value for velocity, one may write a new expression as follows:

$$0 = -\nabla p + \mu \nabla^2 \langle \mathbf{u} \rangle - \frac{\mu}{k} \cdot \langle \mathbf{u} \rangle \tag{4.11}$$

here, the permeability k is seen as a drag force coefficient.

In practice, in most cases, the viscous term $\mu \nabla^2 \langle \mathbf{u} \rangle$ is negligible compared with $\frac{\mu}{k} \cdot \langle \mathbf{u} \rangle$. Considering this, one may obtain an easy recognizable Darcy's law:

$$0 = -\nabla p - \frac{\mu}{k} \cdot \langle \mathbf{u} \rangle \Rightarrow \langle \mathbf{u} \rangle = -\frac{k}{\mu} \cdot \nabla p \tag{4.12}$$

here "–" indicates that velocity is directed towards the pressure decrease.

Figure 4.2 summarizes all above-mentioned conversions. First, it is required to shift from Navier-Stokes equations to the lattice Boltzmann approach in order to determine such macroscopic fluid parameters as density and velocity. At this level, the most serious assumption is referred to fluid incompressibility (or having a low-Mach number). In general, it seems to be a valid assumption for liquid meaning that its density should stay constant.

Next step deals with transition from direct pore-scale simulations to continuum modeling which allows using a very simple Darcy's law to find the permeability.

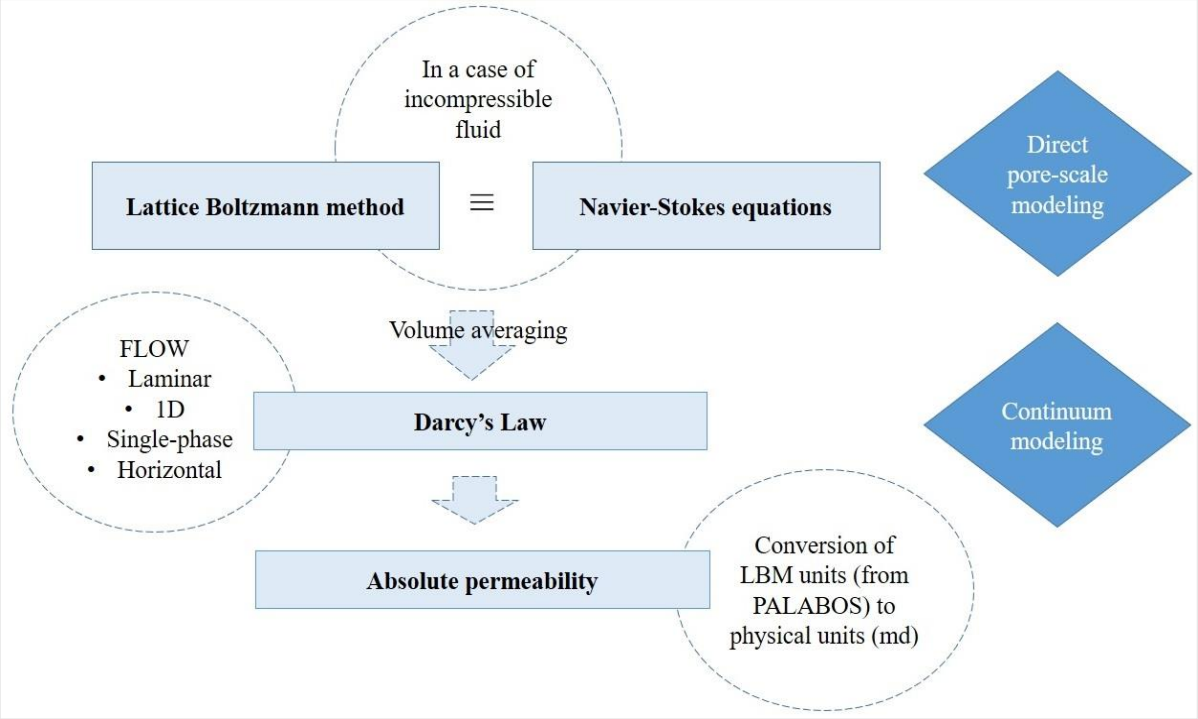


Figure 4.2. Scheme showing how the absolute permeability is calculated

It is important to mention that permeability as an output from PALABOS is given in the dimensionless lattice units. To convert it to physical units the following formula can be used (Sukop, et al., 2013):

$$k_{physical} = k_{PALABOS} \cdot \left(\frac{L_{physical}}{L_{PALABOS}} \right)^2 \tag{4.13}$$

where $k_{PALABOS}$ is permeability from PALABOS in lattice units, $L_{physical}$ is the length in physical units (i.e. image resolution times on pixel size) and $L_{PALABOS}$ is the length in lattice units.

Other practical questions about PALABOS simulations (e.g. how to check flow laminarity) will be discussed in Chapter 6.

4.2.2. Boundary conditions

Within the image samples, two types of boundary conditions have been used in LBM simulations in this work – bounce-back or no-slip boundaries and 'no dynamics' boundaries.

It is a common practice to represent solid boundaries using simple bounce-back condition meaning that when a batch of particles hit a solid wall node at a certain time step, they will be "bounced back" to the node in a pore space from where they came, as Figure 4.3 demonstrates.

Here, solid phase is shown as a dark gray semicircle, while pore space – in a white color. Nodes that are situated deep in the solid phase and do not have any neighbours from void space are displayed in yellow color, those who are in solid phase, but have pore neighbors – in light blue, and finally nodes situated in the pore space – in dark blue. At each time step, particles can be redistributed according to the possible directions showing by arrows between nodes. As it clearly seen from Figure 4.3, once particles reach solid border, they have to jump back to pore space. Thus, for nodes showing in light blue, bounce-back boundaries are used, and for yellow nodes – 'no dynamics' rule is valid.

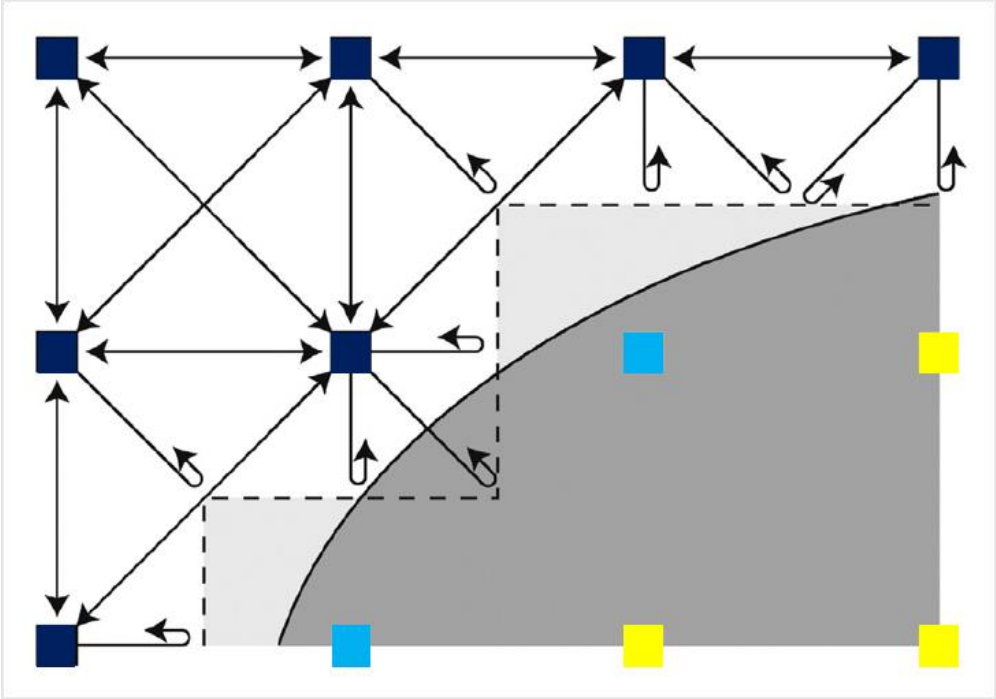


Figure 4.3. Different types of boundary handlings in the LBM
(adapted for PALABOS from Bultreys, et al., 2016)

Furthermore, the inlet and outlet boundaries of image samples are characterized by the Dirichlet boundary conditions.

In some cases, the simulated flow does not stay stable. In this case, it should be simulated using periodic boundary conditions, which are typically applicable for the pairs of connecting faces with similar area. However, real pore microstructure does not have this geometrical feature. Therefore, to mimic cyclic boundary conditions, one may surround the representative pattern by a thin layer of void space. It should be thin enough to have a small impact on the simulation results. Usually, the thickness should be of the same order of magnitude as an average throat radius (Soulaine, 2017). The implementation of this option in PALABOS for several models is considered in more details in Chapter 6.

4.2.3. Issues on grid spacing, REV and image resolution

As it has been discussed previously, the LBM is a powerful tool to compute fluid dynamics, even for complex pore structures. However, it is impossible to predict absolute and relative permeabilities accurately without having good quality input data, i.e. digital rocks models.

There are two main parameters influencing on the quality of digital rocks, viz. image resolution, and the size of digital rock, or representative elementary volume. More formal and general definition of REV states that the representative elementary volume is the smallest volume over which a measurement can be made that will yield a value representative of the whole sample (Keehm & Mukerji, 2004) & (Hill, 1963).

In order to have reasonable comparison among several digital rocks, it is practically convenient to use a common length parameter that can be measured for all of them. Due to its easy calculation and rigorous definition, it is possible to use the characteristic length as such a parameter. It can be described as the first decay length (i.e. when correlation function reaches zero at the first time), and it can be easily extracted from the autocorrelation plot. It is important to note that the choice of the most suitable length scale should be defined by considering both the extra cost of numerical simulations and improvement in accuracy.

In a case of homogeneous and well-sorted pore structures like Berea or Fontainebleau sandstones and for the single fluid flow simulations, it has been suggested to choose length scale according to the following empirical rule (Keehm & Mukerji, 2004):

$$L \geq 10a \tag{4.14}$$

where a is a characteristic first decay length.

Due to some computational limits related to the selected single relaxation time BGK-scheme, the input micro-CT models have to be further cropped in order to satisfy these computational constrains. Based on Formula 4.14, it is possible to check whether the chosen size of cropped models is representative enough to estimate absolute permeability.

Furthermore, in a case of numerical simulations, the question of grid spacing is closely linked with REV concept, since it was showed that permeability results depend on the number of grid points (Keehm & Mukerji, 2004), as indicated by Figure 4.4 (left).

When the grid spacing is big (i.e. when coarse grids are used), the permeability is overestimated significantly. This happens mainly because of poor representation of complex pore geometry by coarse grids. However, almost in all LBM studies, the numerical grid coincides with voxel grid, and it can be difficult to separate effect of improved image resolution from the improving numerical resolution. When grid spacing is low enough ($d < a/8$), the pore structure is well represented by the digital rocks (Keehm & Mukerji, 2004).

The same effect is seen in a case of low image resolution, since images with low resolution lose details in structural complexity and show solid-void surfaces in more staircase-like manner, as it is illustrated in Figure 4.4 (right).

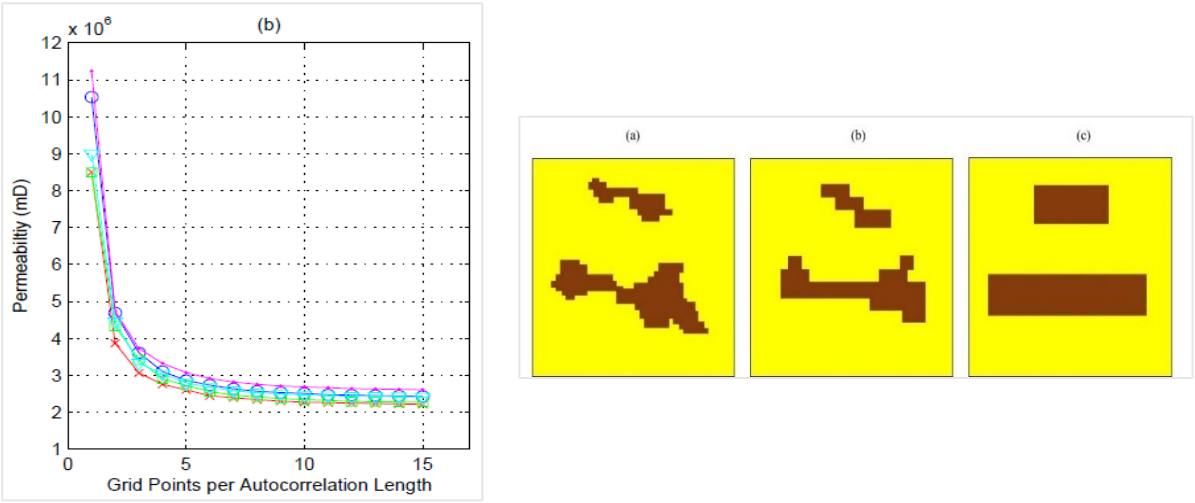


Figure 4.4. Grid spacing effect on permeability and its overestimation by coarse grid (taken from Keehm & Mukerji, 2004, pores are shown in dark brown)

This loss in structure's definition usually provides less values of the specific surface area (i.e. pore surface area per unit volume). It means that for samples with the same porosity, smaller specific surface gives higher permeability according to the general form of the Kozeny-Carman equation (Berg, 2014):

$$k = c_0 \tau^2 \cdot \frac{\varphi^3}{S_0^2} \tag{4.15}$$

where c_0 is Kozeny's constant, τ is tortuosity, φ is porosity, and S_0 – specific surface area.

It is possible to modify Equation 4.15, considering the connection between tortuosity and formation factor (Torsæter & Abtahi, 2003):

$$F = \frac{\tau}{\varphi} \tag{4.16}$$

where τ is the tortuosity of the rock, and φ – its porosity.

Inserting Formula 4.16 into Equation 4.15, one can obtain the equation that relates permeability, formation factor and specific surface area:

$$k = \frac{c_0 F^2 \varphi^5}{S_0^2} \tag{4.17}$$

Chapter 5.
Pore Network Modeling

In spite of the fact that LBM simulations are characterized by relatively simple implementation and suitability for parallelization, there are several important limitations of their use. They are related to their computational efficiency and numerical stability for multiphase flows with large density & viscosity ratios, e.g. for water-gas systems. Furthermore, it is not so easy to connect model's interaction forces to the modeled physical processes in the LBM for multi-phase cases (Bultreys, et al., 2016). Finally, researchers from the Stanford University proved that in a case of two-phase flow simulations, suggested length (Equation 4.14) can be not enough for accurate relative permeabilities' prediction, and hence 2-3 times bigger length size is required (Keehm & Mukerji, 2004). However, bigger volumes cannot be used in PALABOS with simple BGK operator, and more complicated models have to be applied. Therefore, pore network modeling approach has been considered as the most successful modeling tool for practical applications of pore scale simulations of two- and three-phase flow.

5.1. Review of pore network extraction methods

Pore network models describe complex pore structures by a network of simplified geometrical elements – pore bodies and pore throats (the latter are narrow constrictions that connect pores together). Attempts to extract pore networks from 3D images have been tried for more than two decades, although the first serious research using network concept goes back to 1950s, when Fatt created regular lattice of tubes with random radii to simulate fluid flow through the porous medium. After that, pore size distributions started to be used as constrains for regular lattice patterns to match coordination numbers of the considered networks. However, in most cases, regular lattice networks are not able to catch the complex topology of the complex porous structures properly. Therefore, image based network extraction methods became to be widely used starting from 1990s, since they could reproduce the real pore space. When transition to network approach from the real pore microstructure occurs, some arbitrary considerations should be made as to where a pore ends and a constriction starts. It means that PNM is non-

unique representation of pore space, and for each network extraction method different assumptions have to be made, which may complicate significantly a comparison of networks obtained by different extraction algorithms (Bultreys, et al., 2016).

Typically, network extraction includes segmentation of continuous pore space into discrete network elements and defining the main geometrical properties of each element, e.g. hydraulic radius, the volumes and lengths of pores and throats, etc. (Bultreys, et al., 2016).

There are several different approaches to classify PNM extraction, however, the most common one is to distinguish two classes: topology-central and morphology-central methods (Bultreys, et al., 2016). Medial axis, maximal balls and watershed algorithms are discussed in more details below, as the typical representatives from both groups, as well as the most widely used ones.

Medial axis extraction techniques (very often, skeletonization based algorithms are included as a part of them) are related to topology-central methods. They use thinning effect, which includes deleting the pore space from rock surfaces until a line with denoting centers of the pore space – or medial axis – can be found. Thinning starts in the largest pores and occurs while this erosion is not changing the topology of the pore structure. Figure 5.1 shows an example of medial axis network extraction of the pore space for the Fontainebleau sandstone sample. Here, pores will be located at branches of the skeleton, while throats will connect pores.

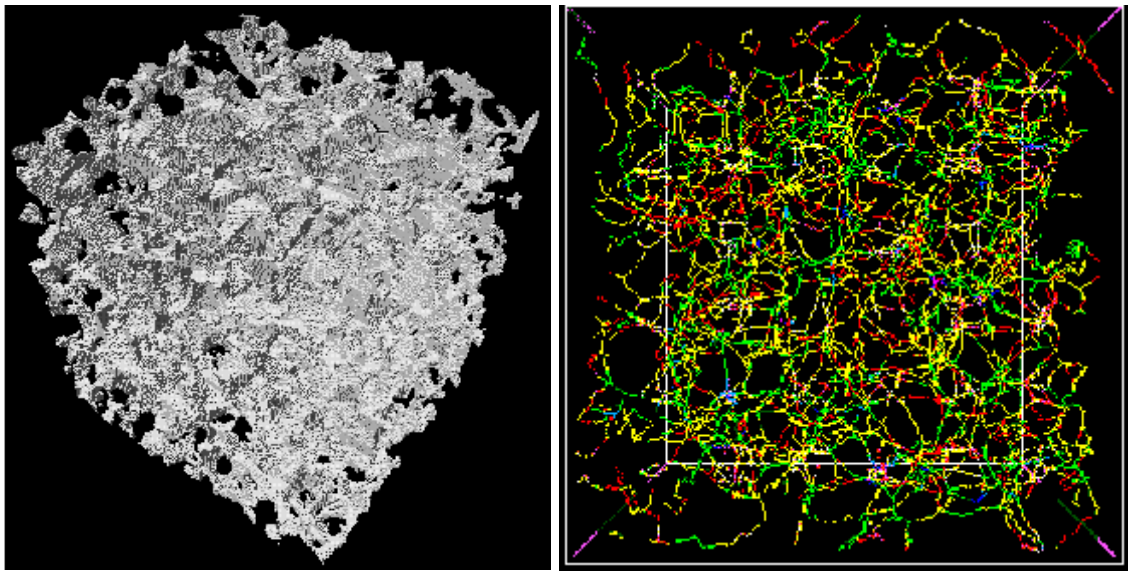


Figure 5.1. 256^3 voxel image of Fontainebleau sandstone with 12% porosity and 6 μm/pixel resolution (left); medial axis of its pore space (right)
(taken from *www.ams.sunysb.edu*)

Methods based on the concept of the medial axis capture pore interconnection adequately, but may have some problems with pore identification, i.e. partitioning of the pore bodies.

Furthermore, they are very sensitive to small defects in the input pore space surface that may lead to significant pore over-segmentation, as faked brunches of the medial axis would be identified. Pre- and post-processing of the input images can help to avoid this problem (Bultreys, et al., 2016) & (Dong, et al., 2007).

The concept of maximal ball algorithm has been suggested by Silin, Jin and Patzek (Silin, et al., 2003). It begins with extraction of maximal balls which are the largest inscribed spheres centered on each void voxel that just touch the grain or the boundary. After that spheres located completely inside other spheres are removed, allowing to decrease the complexity of resulting set of maximal balls. It is possible to cluster them into families, where the common ancestor of each cluster denotes a pore. A throat is determined as a child of parents with different common ancestors, as Figure 5.2, left shows. Here, the white arrows display the pore-throat chains, and they are used to define the pore space's topology (Dong, 2007). The right figure shows the typical network representation extracted by maximal balls method.

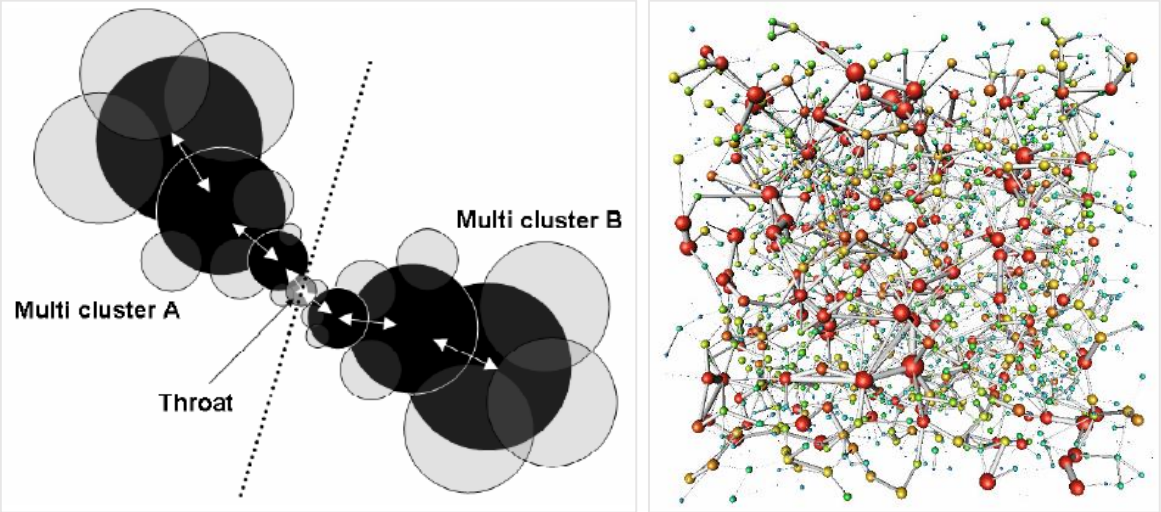


Figure 5.2. Clustering of overlapping maximal balls into family trees (left); the extracted network for S1 sample using the maximal ball algorithm (right).
Taken from Dong, 2007

The main advantage of the maximal ball method deals with explicit manner of pore and throat distinction. In general, the scheme works good for pore identification, but the construction of throats may become a challenge, since there can be several ways to connect pores by overlapping smaller spheres (Dong, et al., 2007).

Finally, the pore space can be segregated on discrete network elements by using of a watershed algorithm on the distance map calculated from the segmented pore space image. The throats

can be found as surfaces separating pores from each other (Rabbani, et al., 2014). More details about watershed algorithm are given in the following subchapter.

5.2. Watershed algorithm

It has been decided to use a watershed algorithm to extract important network characteristics, since it is applicable in 2D, and it can be more understandable in this case. While using the binary images, it is required to modify them and to find boundaries between pores and grains using special segmentation technique.

5.2.1. Watershed in 2D by image segmentation and pore size distribution

As it has been mentioned before, pore size distribution and average pore diameter have significant impact in pore network characterization. Pore size distribution describes the range of sizes for pores in a considered sample or image. Very often, pore sizes are described by single value of average pore diameter (or radius). As it is shown later, average pore diameter is directly connected to a correlation length parameter which was introduced in Chapter 3 (Equation 3.12).

The watershed segmentation method couples two well-known image processing algorithms of distance (`bwdist` command in MATLAB) and watershed transform (`watershed` command in MATLAB), due to the fact that watershed segmentation can produce reasonable results only when it is applied to images with different minima and catchment basins. However, binary images contain only two possible outcomes, zero or one. Therefore, before applying watershed segmentation to binary images, the distance transform should be used to preprocess them.

To understand how the algorithm works, one may consider two connected objects with local deepest points situated in their centers, as Figure 5.3, left indicates. These deepenings are called catchment basins. If water starts to come up in these basins now, the first contact line before it will be connected, is called a watershed ridge line. By cutting the structure on this line, the two touching objects can be separated and labeled as pore bodies, while the line between them is considered as a pore throat, as it can be seen from Figure 5.3, right (Rabbani, et al., 2014).

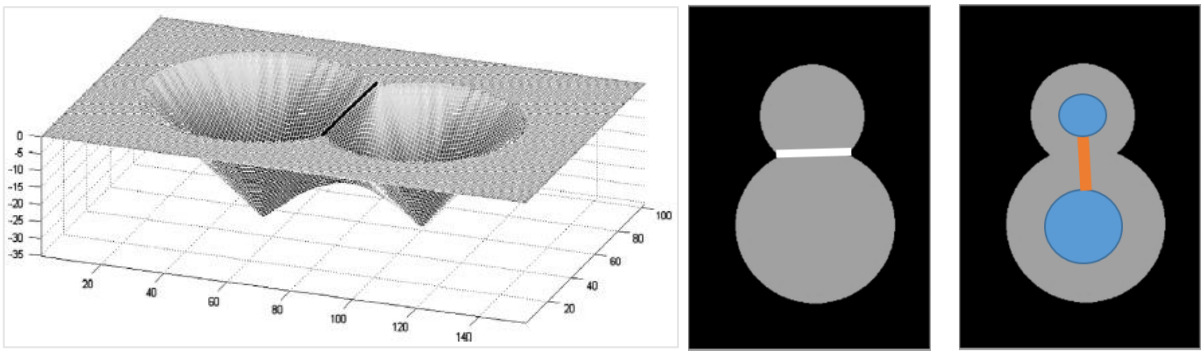


Figure 5.3. Watershed based algorithm: catchment basins (left) and two detected pore bodies, separated by ridgeline or as a ball-and-stick model (right)
(taken from Rabbani, et al., 2014)

As it was mentioned previously, distance transform plays a significant role in detection and distinction of connected pores. For each pixel (x_i, y_i) inside the object, the distance from the nearest non-zero pixel need to be calculated. In order to find a distance map, the distance between this particular pixel and an arbitrary non-zero pixel should be minimized (Rabbani, et al., 2014).

This Euclidian distance is:

$$D = \sqrt{(x_{n-z} - x_i)^2 + (y_{n-z} - y_i)^2} \tag{5.1}$$

where (x_{n-z}, y_{n-z}) are coordinates of the nearest arbitrary non-zero pixel.

One thing that should be kept in mind is that watershed segmentation can be sensitive to image noise, and sometimes it has a tendency to "over-segment" input images. The reason behind this, is that each local minimum (even a small one), can easily become a catchment basin. To avoid over-segmentation, median filter should be applied on the distance map. For thin section images, the median filter with the same 7x7 neighboring interval has been used.

Chapter 6.
Methodology and Results

The following chapter presents the used methodology and obtained results. Each subsection below is referred to one of the chapters (Chapter 2-Chapter 5) that described theoretical aspects.

6.1. Image analysis

6.1.1. Information about input images and models

In this work, a set of two microscopic images and six micro-CT models (incl. five sandstone and one carbonate) are analyzed. The images are color thin sections of reservoir samples collected from the outcrops in the Southern England (the Wessex Basin). Five micro-CT models are taken from the open source database of the Imperial College London. Samples had been chosen such wise to represent a broad variety in permeability values. General information related to the porosity, permeability and other properties of images and samples is presented in Table 6.1.

Table 6.1. General information about input images and models

Images					
Image #	**Information**	**Resolution, µm/pixel**	**Porosity, %**	**Absolute permeability, mD**	
1	Sherwood Sandstone Group: Triassic age, fine to medium grained	2.40	36.2	100-140	
2	Bridport Sand: Lower Jurassic age, very fine grained	1.94	28.1	15-25	
Micro-CT models					
Model #	**Information**	**Resolution, µm/pixel**	**Porosity, %**	**Absolute permeability, mD**	
1	Berea Sandstone	5.420	14.0	-	
2	Berea Sandstone: high permeable	5.345	19.0	1286*	1111**
3	Sandstone S1: high permeable	8.643	14.1	1678*	1486**
4	Sandstone S3: medium permeable	9.100	16.9	224*	281**
5	Sandstone S4: medium permeable	8.960	17.1	259*	169**
6	Carbonate C2	5.345	16.8	72.3*	158**

Note: * – experimentally measured permeability (averaged value); ** – permeability obtained from network modeling

Figure 6.1 shows microscopic images #1-2.

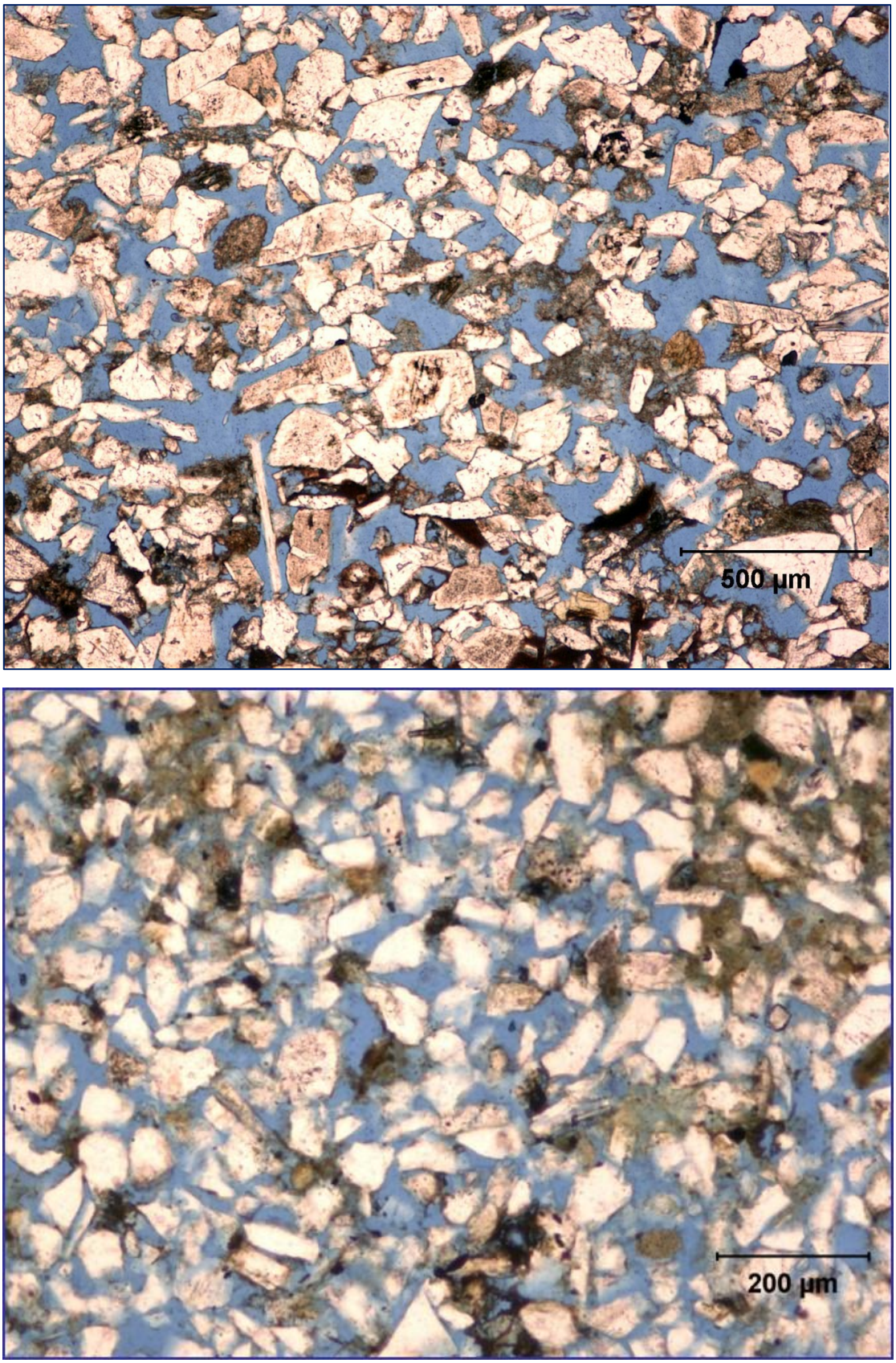


Figure 6.1. Input thin section color images.
The upmost is the image of the Sherwood sample, the left in the bottom row is the thin section of the Bridport sample, and the right one is the binary micro-CT image of Berea sample

Pore space in colored thin section images from the Wessex Basin is indicated in light blue color, since they had been prepared by a blue epoxy impregnation. Sand grains are shown by white or light beige colors. In general, grains for the Wessex Basin's samples are characterized by angular shape. The Sherwood sample is described by greater variety in grain size and shapes, however, the Bridport sample shows denser packing of grains. Furthermore, it is possible to distinguish some dark inclusions in colored images #1 and 2. Image of the Sherwood sample (the upmost one in Figure 6.1) may contain some dark brown inclusions related to hematite mineral that also gives the outcrop its characteristic red color. Grains that are dark brown can be fragments of organic material or calcite cement, since the grains of calcite against glass have dark edges due to density and high refractive index (Nichols, 2009). It is important to mention that the presence of dark inclusions in the Wessex Basin's color thin sections can challenge their binarization process. Therefore, to avoid the probable misestimation of porosity, a proper thresholding should be applied.

Note: Hereinafter, for the binary thin section images, pores will be indicated in black and grains – in white, however, for micro-CT models pores will be displayed in white and solid matrix will have a black color. This choice is simply due to the initial light color of solids in the thin sections.

The input X-ray microtomography models studied under the current research are presented in Figure 6.2-Figure 6.7, which show 3D cubes of their microstructure.

As it was mentioned in Chapter 4, due to the limit of computational power, these models should be further modified, i.e. during permeability estimation and reconstruction procedure only their fragments are used. The final size of analyzed samples will be chosen according to the REV concept.

High permeable samples

Berea Sandstone (sample #2): Initial size 400x400x400 voxels

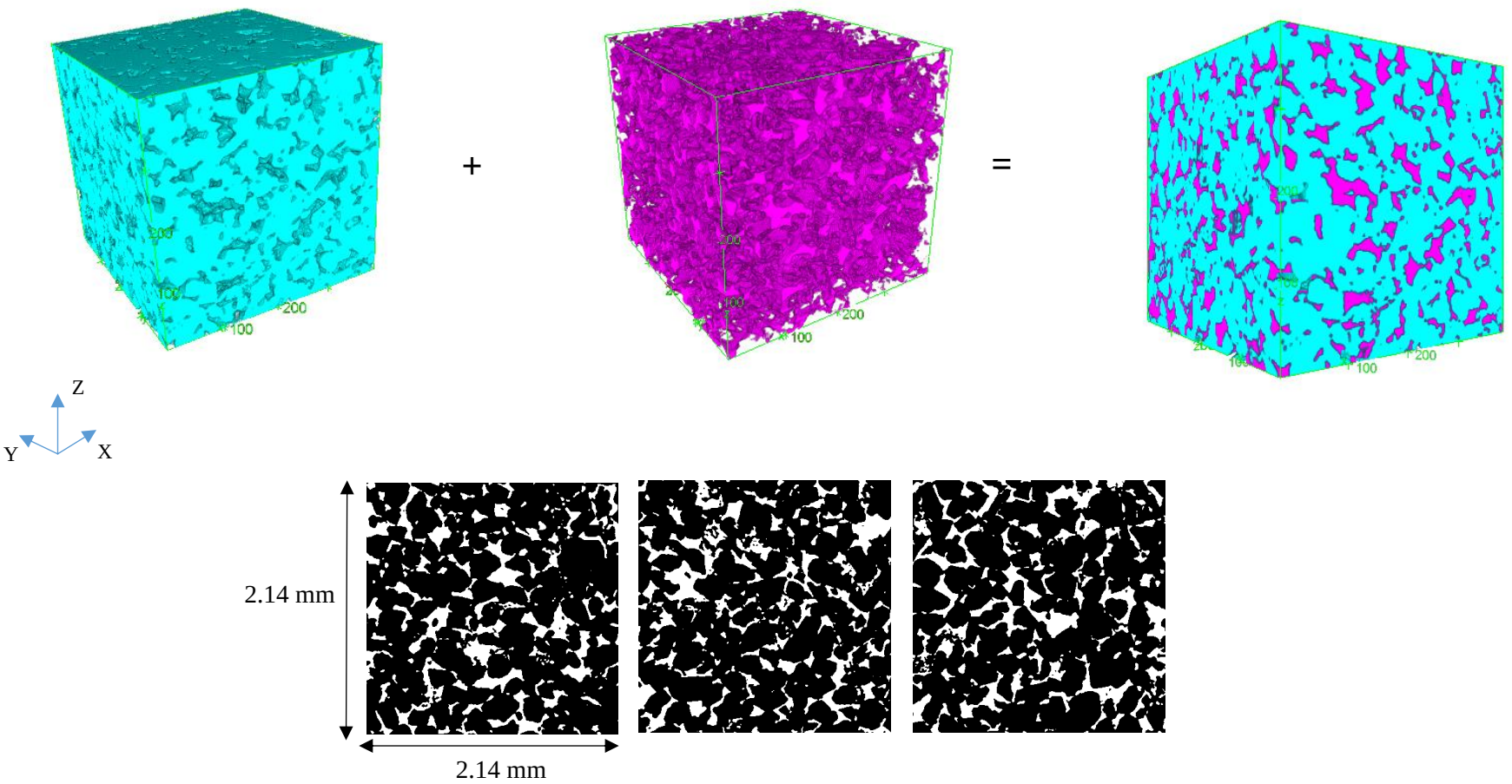


Figure 6.2. Top: 400^3 voxel 3D micro-CT representation of Berea sandstone structure (pores are shown in magenta, solid matrix – in cyan); bottom: several 2D cuts from the model (pores are white, rock matrix is black)

Sandstone S1: Initial size 300x300x300 voxels

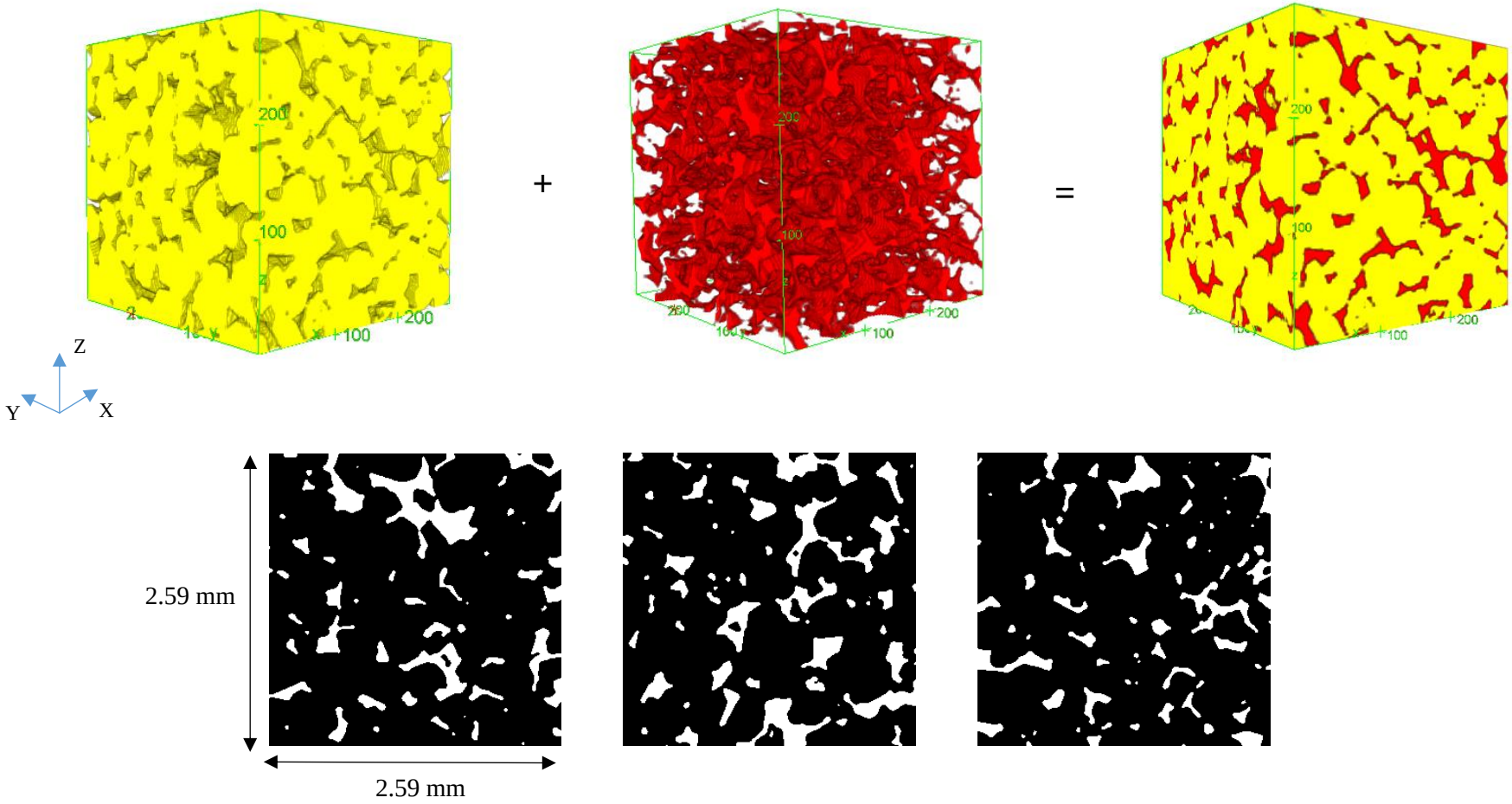


Figure 6.3. Top: 300^3 voxel 3D micro-CT representation of S1 sandstone structure (pores are shown in red, solid matrix – in yellow); bottom: several 2D cuts from the model (pores are white, rock matrix is black)

Medium permeable samples

Sandstone S3: Initial size 300x300x300 voxels

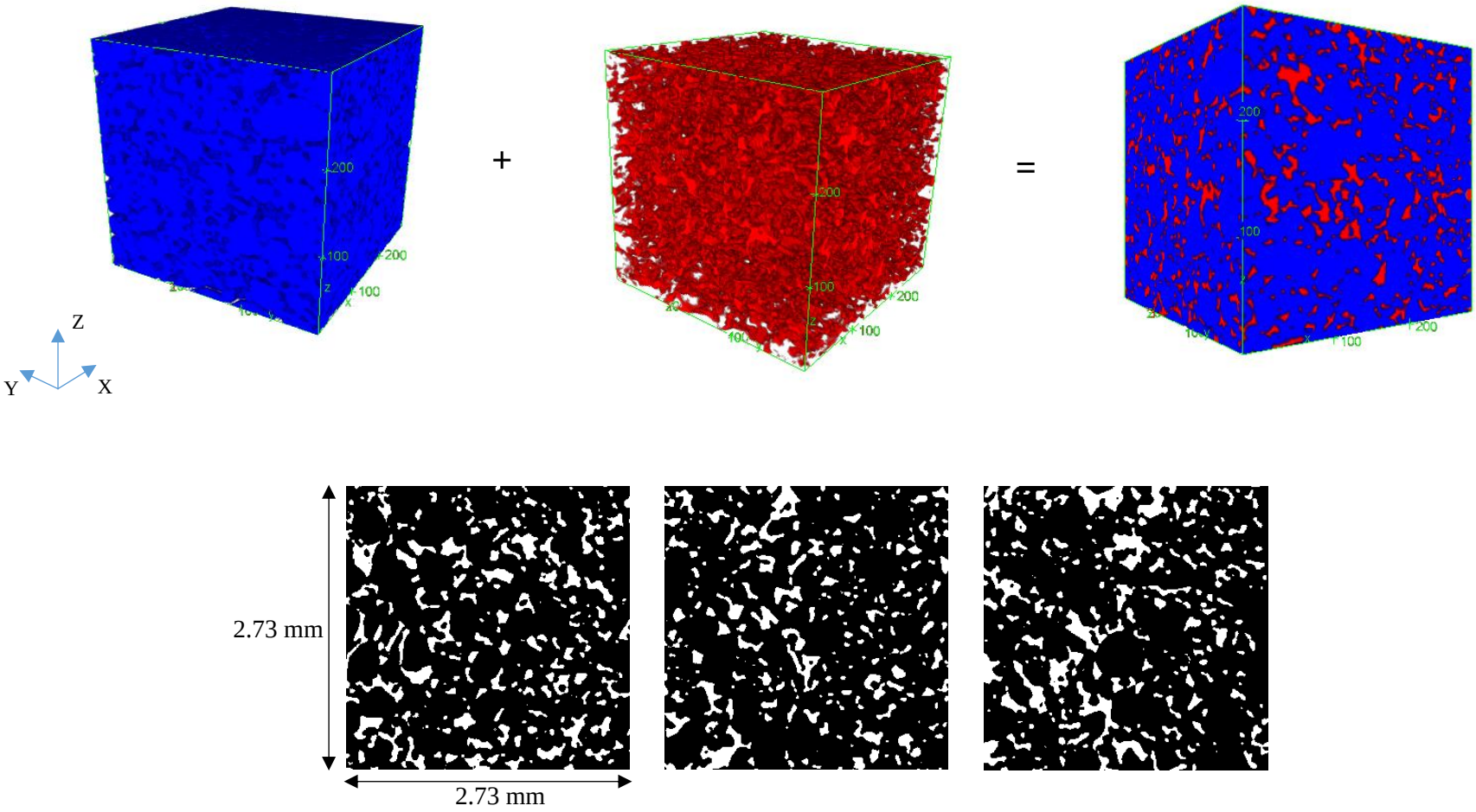


Figure 6.4. Top: 300^3 voxel 3D micro-CT representation of S3 sandstone structure (pores are shown in red, solid matrix – in blue); bottom: several 2D cuts from the model (pores are white, rock matrix is black)

Sandstone S4: Initial size 300x300x300 voxels

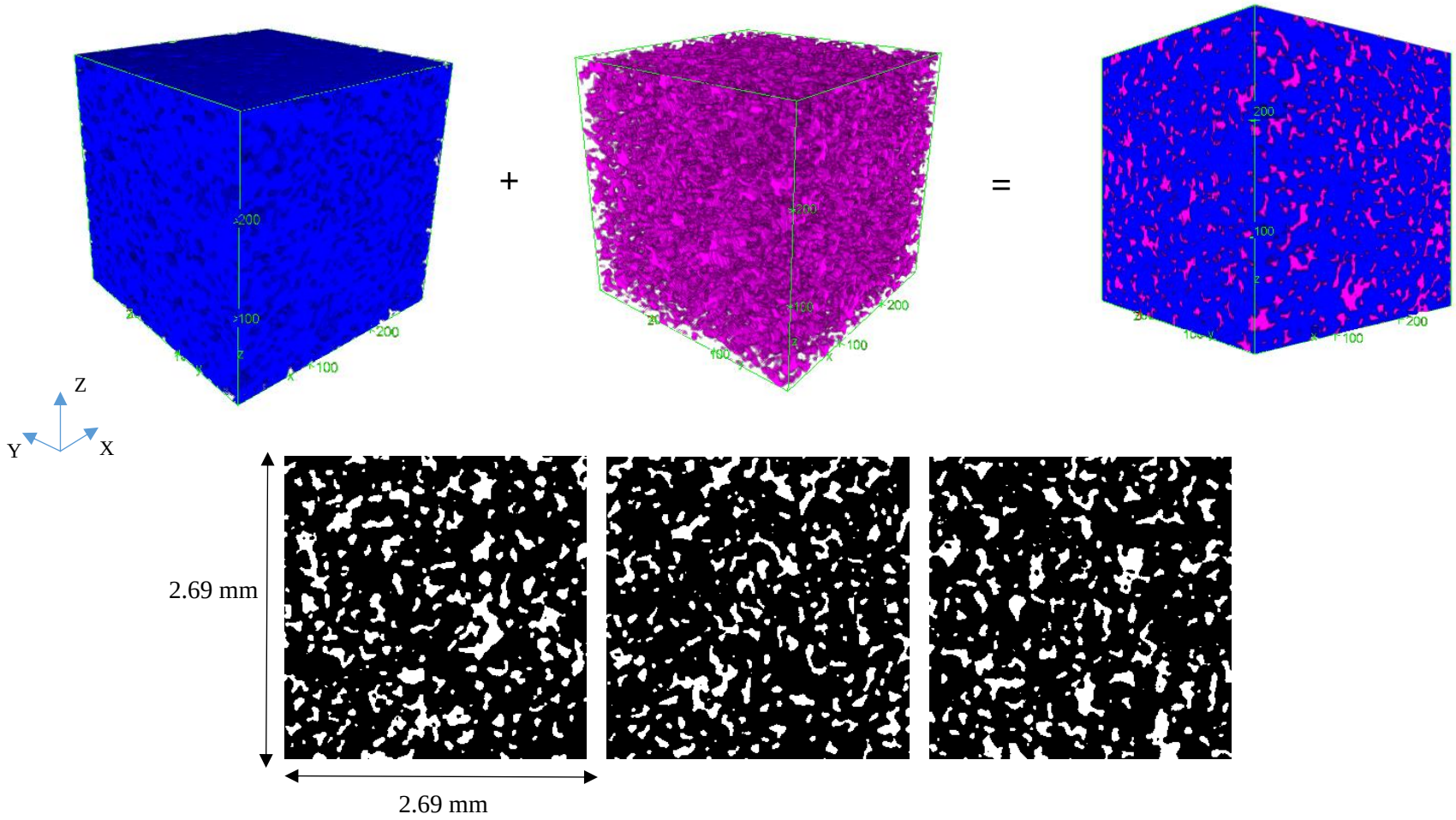


Figure 6.5. Top: 300^3 voxel 3D micro-CT representation of S4 sandstone structure (pores are shown in magenta, solid matrix – in blue); bottom: several 2D cuts from the model (pores are white, rock matrix is black)

Berea Sandstone (sample #1): Initial size 713x713x262 voxels with further cropping of cubic model (262x262x262 voxels)

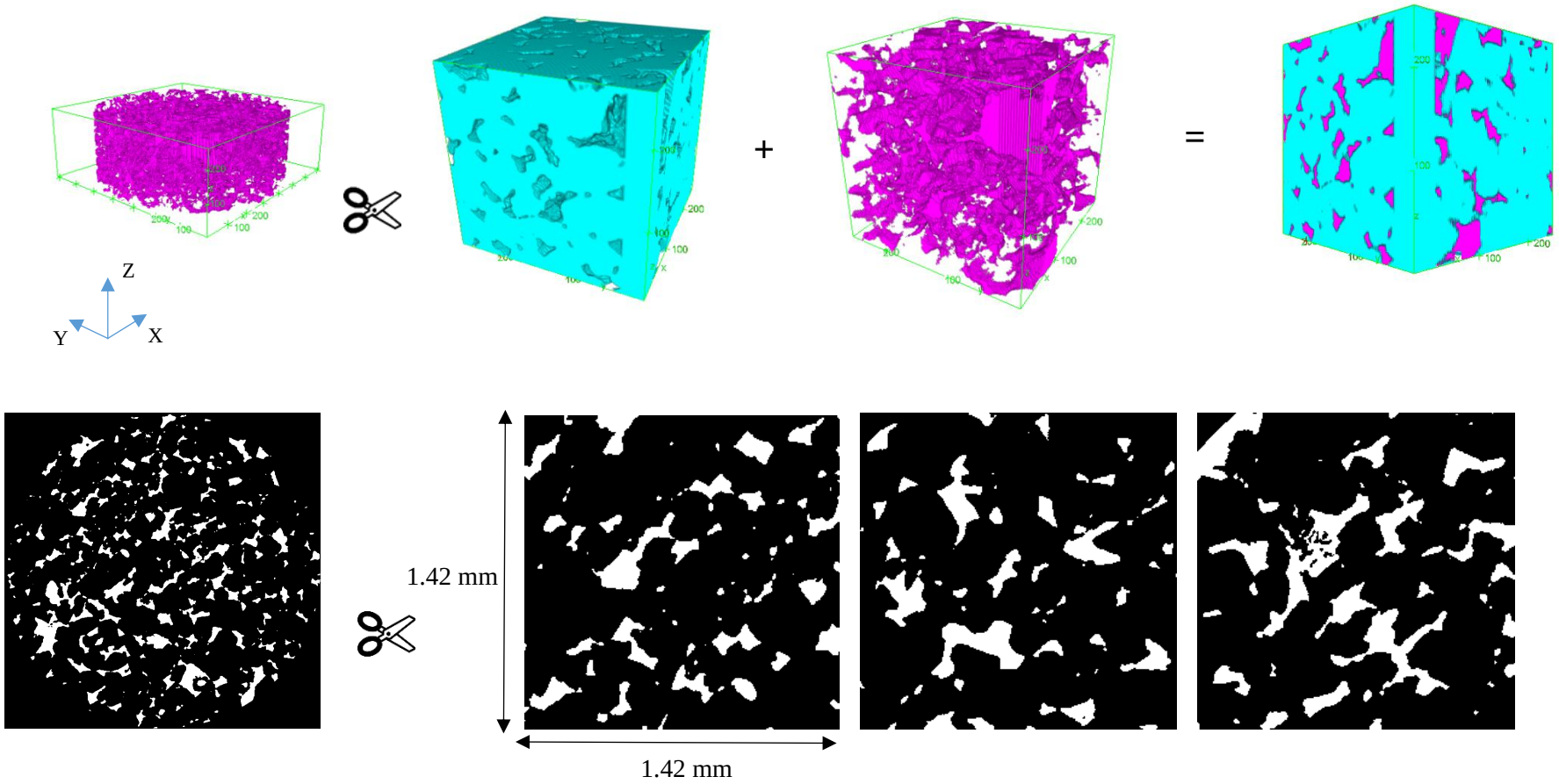


Figure 6.6. Top: Initial and cropped 3D micro-CT representation of Berea sandstone structure (pores are shown in magenta, solid matrix – in cyan); bottom: several 2D cuts from the model (pores are white, rock matrix is black)

Carbonate sample C2: Initial size 400x400x400 voxels

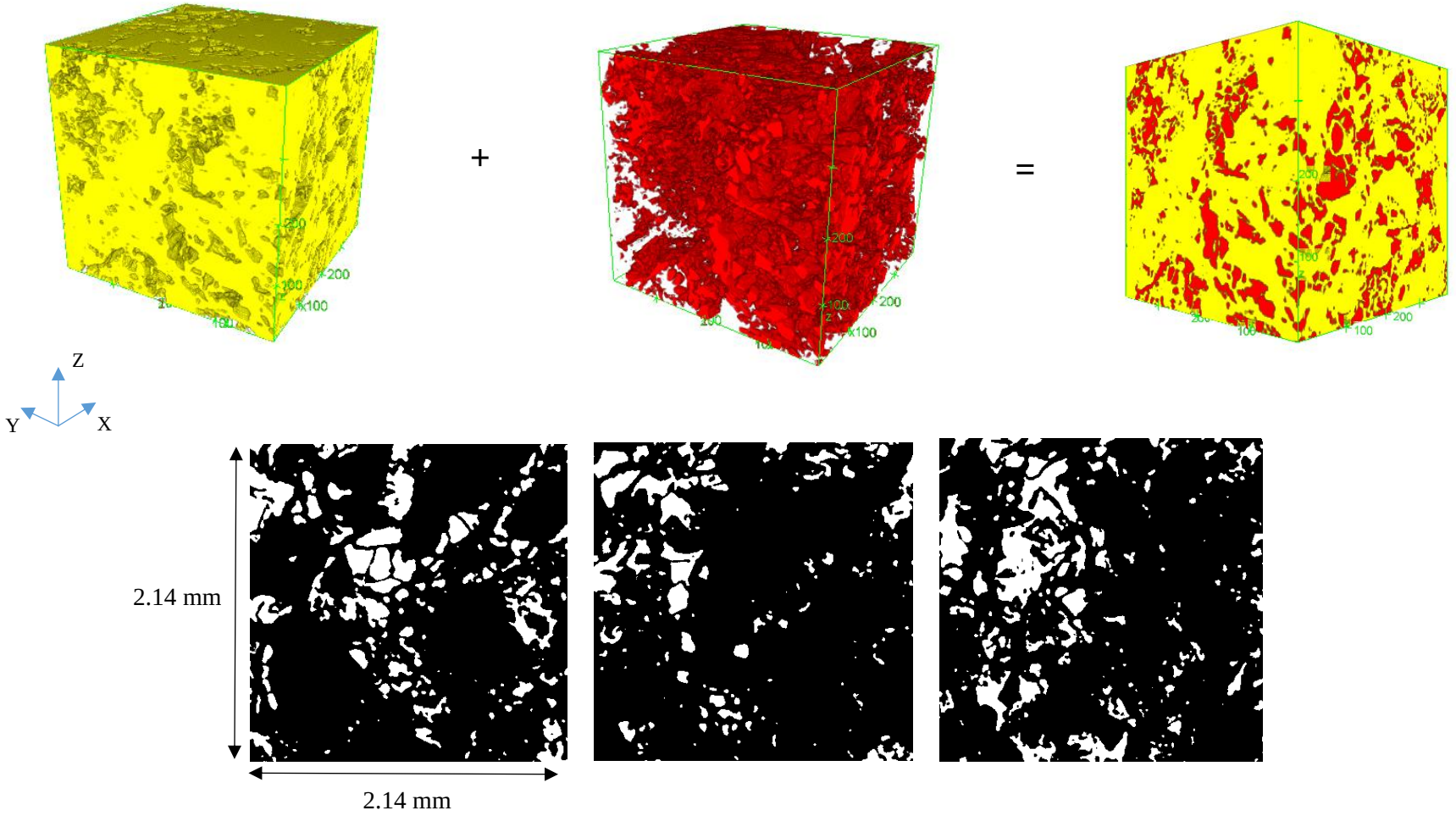


Figure 6.7. Top: 400^3 voxel 3D micro-CT representation of C2 carbonate structure (pores are shown in red, solid matrix – in yellow); bottom: several 2D cuts from the model (pores are white, rock matrix is black)

6.1.2. Thresholding and binarization (for color thin section images only)

The procedure of image binarization for images #1 and 2 has been implemented in MATLAB. As it was mentioned earlier, in the scope of the present work, only threshold method is considered due to its reliability and simplicity.

There are two possible options to binarize thin section images. First method is related to a conversion of input thin sections into gray-colored images using MATLAB command `rgb2gray`. The second technique deals with applying so called multiple dimension thresholds, i.e. finding its own threshold for each component of the input color RGB image. This approach is described in details in the paper "Image analysis and pattern recognition for porosity estimation from thin sections" (Richa, et al., 2006).

Gray-colored images

Having gray images, one can build a gray image data histogram that reflects the distribution of pixels based on the chosen color pattern. Figure 6.8 shows such a gray-colored image and its histogram for one of the analyzed samples (image of Bridport sample, 509x677 pixels size).

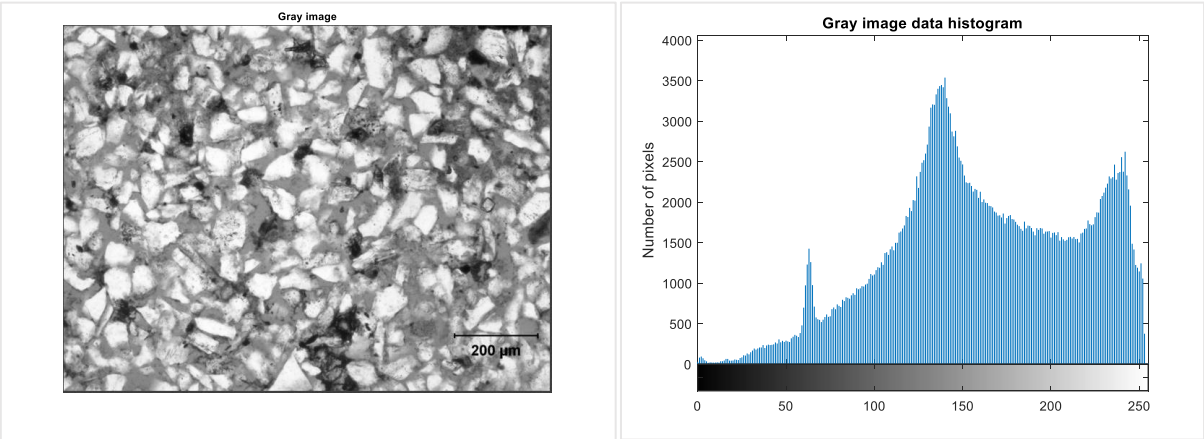


Figure 6.8. Gray-colored image and corresponding histogram for the Bridport sample

As follows from Figure 6.8, the histogram displays several peaks. The first peak is referred to the darkest gray values and can be related to the cemented material. The second peak, most likely, reflects pore space, and the last one shows the grains.

It has been noticed that while applying a single threshold on a gray image, MATLAB considers dark 'spots' as a part of pore space. Since the value of threshold is selected such that the porosity of image should be close to the measured sample porosity, the image quality has been

decreased, and all solid particles become indistinguishable. Figure 6.9 displays an example of a failed binarization by applying a single threshold for the image of the Bridport sample.

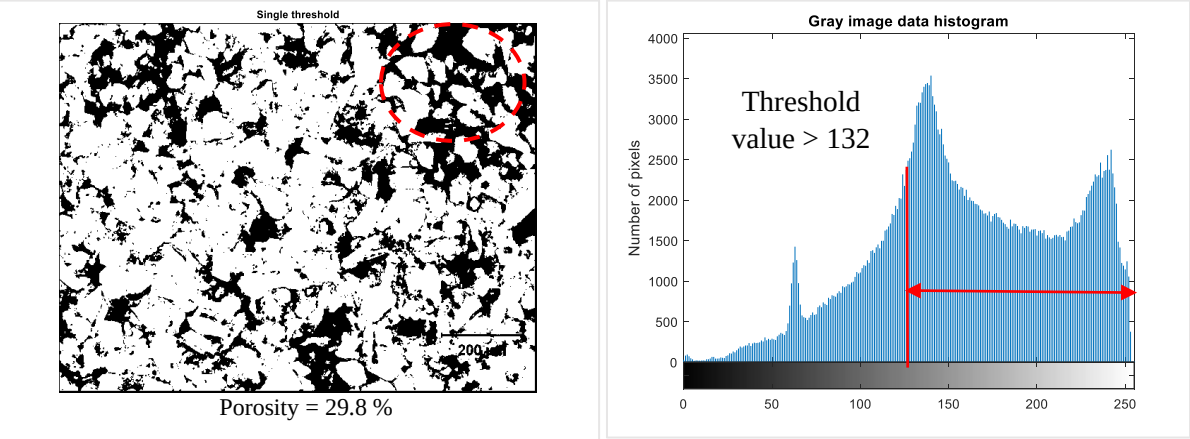


Figure 6.9. Example of a failed binarization using single threshold.
Black pixels marked with red circle are counted as a part of pore space, however, they are related to cement or other mineral intrusions

That is why, it is suggested to use two thresholds for gray images – minimum and maximum ones. In MATLAB, it can be implemented by the following command: `gray_image (gray_image < min | gray_image > max) = 0`, i.e. everything that is out of min & max borders becomes zero. After that, it is possible to binarize image using `im2bw` command. However, this approach typically causes the inversion of the image, i.e. for thin sections pore space displays in white and grains – in black. To return the chosen color pattern (pores are in black, grains are in white), `imcomplement` MATLAB command should be applied. Furthermore, to improve the quality of the obtained binary images and to decrease the image noise, the median filter can be applied. The neighborhood pixel interval of 7-by-7 pixels is chosen. It means that each output pixel contains the median value in the 7x7 neighborhood around corresponding pixel in the input image, as it was discussed in Chapter 3.

Final versions of the Wessex Basin's binary images obtained by min-max thresholding of the gray images are shown in Figure 6.10.

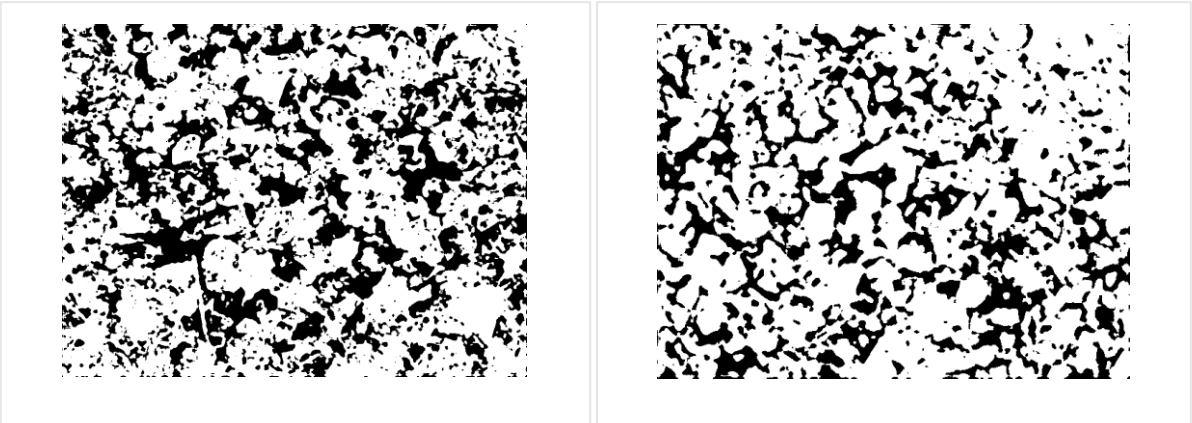


Figure 6.10. Post-processed binary images obtained by double thresholding of gray images. Left is the image of Sherwood sandstone, right – the image of Bridport sand

RGB images

It is important to note that RGB is not an image format, but a color separation scheme, when colored images can be represented as a combination of **R**ed, **G**reen and **B**lue color components unlike JPEG image format, which uses YUV scheme. Several file formats support images in RGB-form, e.g. PNG, TIFF, BMP. In this work, TIFF image format is used. It can be possible to extract each individual component (red, green and blue respectively) in MATLAB using `rgbimage` command. Figure 6.11 presents histograms of color components for the Wessex Basin's images.

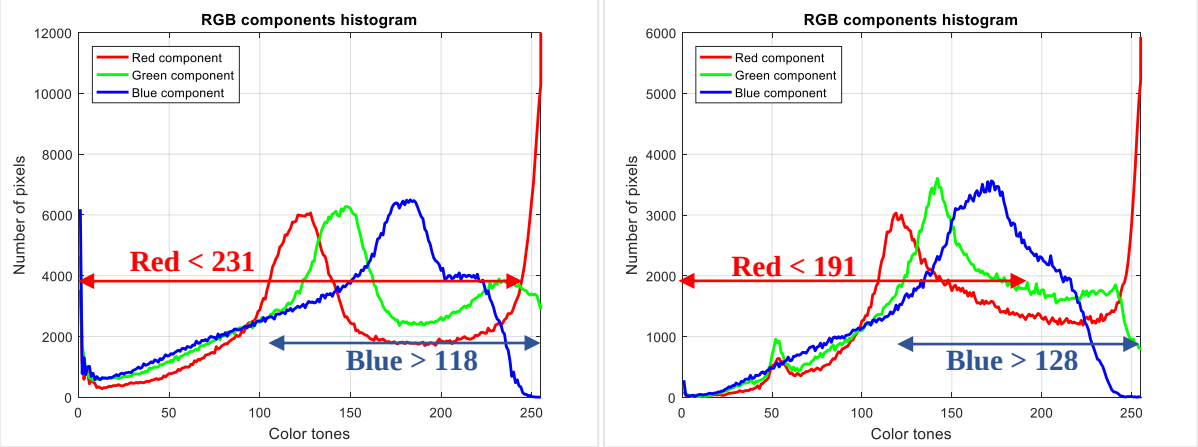


Figure 6.11. Histograms of color components for the image of Sherwood sample (left) and for the image of Bridport sample (right) with chosen values of thresholds

As it is stated in the article "Image analysis and pattern recognition for porosity estimation from thin sections", for RGB color-space reasonable binary images are obtained if only red and blue components have been constrained, since thresholding of green constituent element often

distort the output (Richa, et al., 2006). That is why, in this work, the same approach is considered.

Figure 6.12 shows obtained binary images for this method of thresholding. The detailed MATLAB code describing both methods is presented in Appendix B.

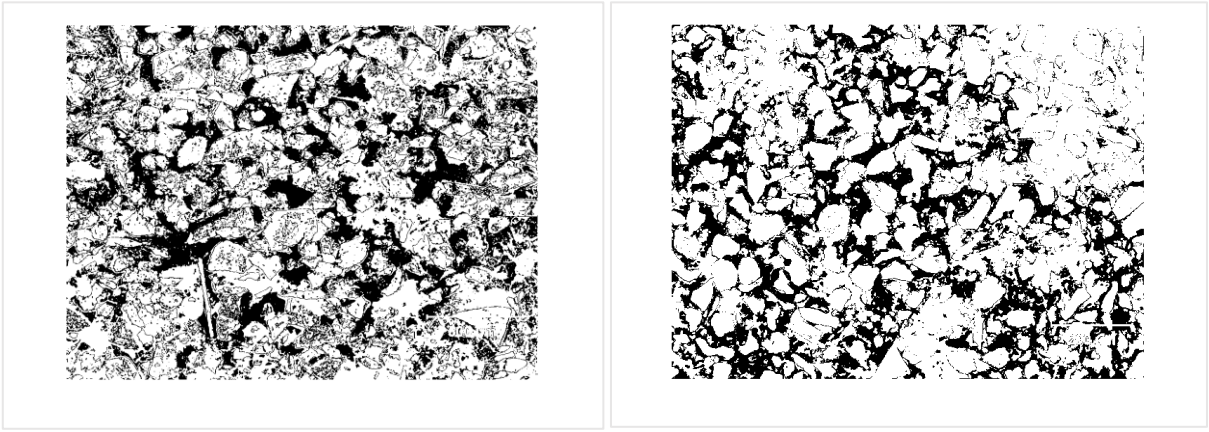


Figure 6.12. Binary unprocessed images obtained by multiple dimension thresholding. The image of Sherwood sample is on the left side, the image of Bridport sample – on the right

As follows from the comparison of Figure 6.10 and Figure 6.12, the obtained binary images resemble to each other in quite a good manner. The porosity estimation is done for both cases, however, henceforth, only binary images obtained by double thresholding will be used, since they have been already post-processed and do not contain noise.

6.1.3. Porosity estimation

In MATLAB, first, a total number of pixels in binary image should be counted by `numel` command, which returns the number of elements in an array. After that, an amount of white pixels can be determined as: `whitepix = sum (B1(:))`, where `B1` is a binary image. The number of black pixels is calculated as the difference between total number and white pixels. Finally, porosity is defined as a ratio of black pixels to the total number of pixels.

Note: For micro-CT images, porosity value is defined as an average for all images in the model. It is possible to read all BMP-files of image sequence in the chosen folder by `find_fil = fullfile(fold,'*.bmp')` and then find porosity as described above (`poro{y} = black_pix{y}/pix{y}`) keeping in mind that here `y` indicates the number of 2D cut images in the folder.

Table 6.2 shows a comparison of porosity obtained by means of image analysis vs. given values, where the relative error is calculated as the ratio of absolute difference between porosity values divided by the value from Table 6.1.

Table 6.2. Comparison of porosity values (image analysis vs. experimental measurements)

Images					
Image #	**Porosity φ, %**			**Relative error Δ, %**	
	Measured	**Double thresholding (gray images)**	**Multiple dimension thresholding**	**Double thresholding**	**Multiple dimension thresholding**
1	36.2	35.6	36.1	1.66	0.28
2	28.1	28.8	28.7	2.49	2.14
Micro-CT models					
Model #	**Porosity φ, %**			**Relative error Δ, %**	
	Measured	**From image analysis**			
1	-	14.30		-	
2	19.6	19.65		0.230	
3	14.1	14.13		0.215	
4	16.9	16.86		0.254	
5	17.1	17.13		0.150	
6	16.8	16.83		0.183	

As follows from Table 6.2, due to an appropriate choice of thresholds (both for gray and color images) values of porosity obtained by image analysis are quite close to the measured ones – the errors are less than 5%. Furthermore, in both cases, binary images capture the pore distribution in a very good manner without counting cemented material as a pore space. Moreover, as it can be seen from results for the micro-CT models, porosity values obtained through a direct pixel counting are close to the indicated values with less than 1% relative errors.

6.2. Stochastic reconstruction and analysis of autocorrelation functions

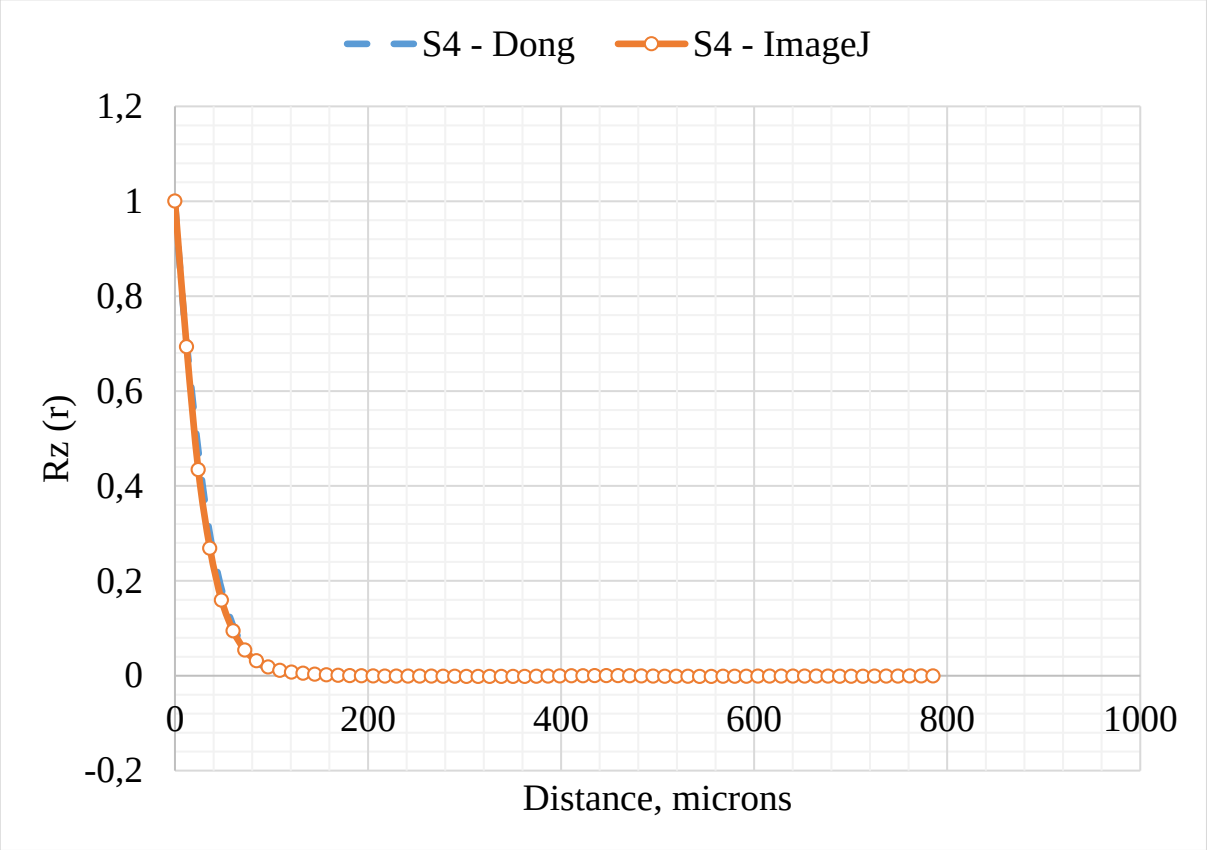
Imagine that there is only one binary image from each model available, and it is necessary to build the whole 3D model of pore space by reproducing statistical information measured on this 2D thin section binary cut. As it has been discussed in Chapter 3, the stochastic reconstruction in this work will be based on matching porosity and two-point correlation function between the given image and those created during the reconstruction.

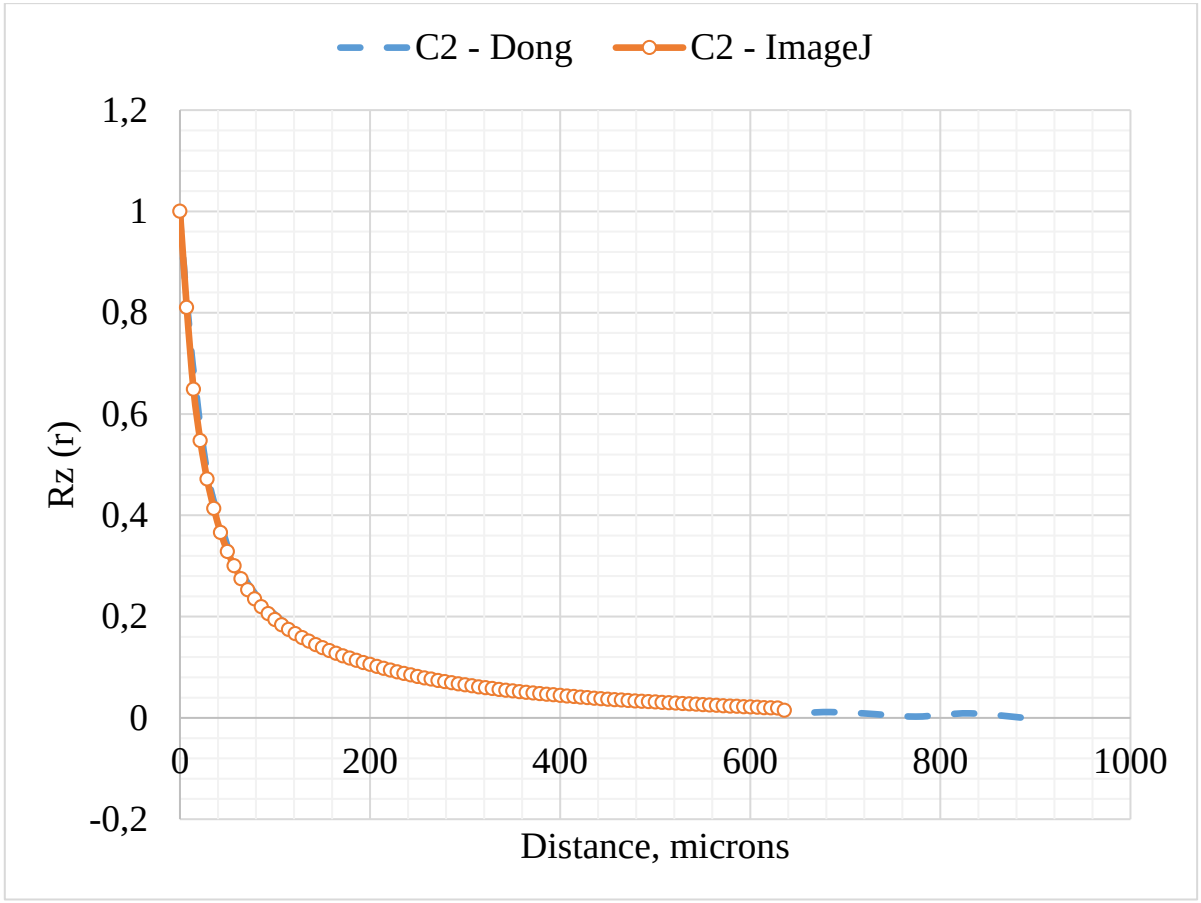
Since there are several micro-CT models available, it can be an advantage to use them for validating the results of numerical reconstruction of the given samples. However, only visual inspection can be not enough, and that is why, it is decided to use permeability and other transport properties estimation as a quantitative parameter to analyze the quality of the reconstruction procedure.

Below, the algorithm of the reconstruction procedure implemented in MATLAB is briefly discussed.

First, it is required to read an input image. In MATLAB, it is possible to do this with function `imread`. The next step is to find the statistical parameters of the target (i.e. input) image, since these correlation functions will be the matching criteria for the whole reconstruction process. To find the radially averaged autocorrelation function (it is called radially averaged autocorrelation function, because instead of using a probe line, it draws 'a probe circle' assuming that there is an isotropy in all directions), the macro from the open-source ImageJ library is used (Appendix C). The macro is based on the Fourier transform, and it considers the finite size of the image (i.e. there is no need to extend the image size to avoid the edging effects). Then it is possible to export values to Excel file, from where they will be read by the MATLAB code.

Figure 6.13 shows a general view of autocorrelation functions obtained for three samples in ImageJ. For two of them – S4 and C2 – it is possible to compare ImageJ results with those obtained by Dr. Hu Dong in his PhD thesis (Dong, 2007). As Figure 6.13 indicates, they look almost identical. The autocorrelation graphs for remained models are presented in Appendix A.





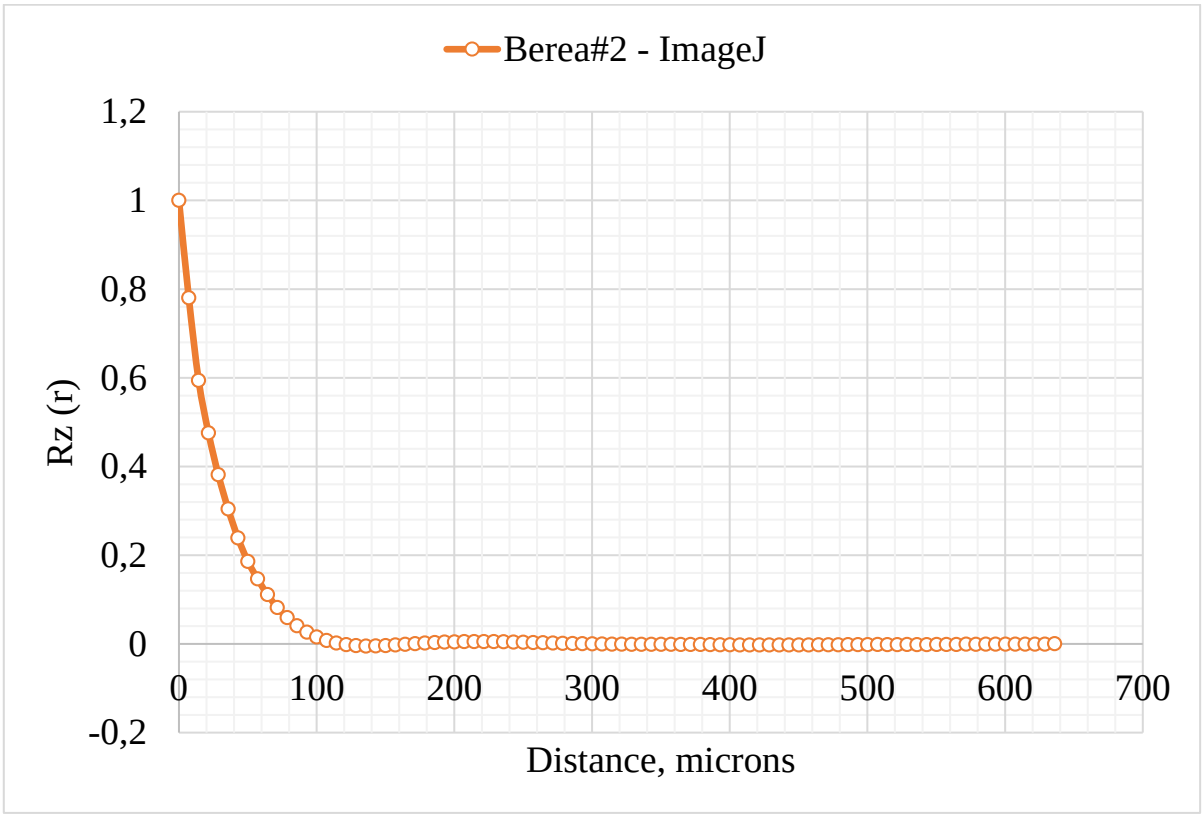


Figure 6.13. Autocorrelation functions for three analyzed samples.
Those from ImageJ are shown in orange color, from Dong's thesis – in blue

Referring to the discussion in Chapter 3 and Chapter 4, the plots of autocorrelation functions may contain important data that can be used in REV estimation. Table 6.3 summarizes information related to the characteristic length obtained from their analysis for all studied samples.

Table 6.3. Information about characteristic length for all samples

Sample	First decay length, microns		Length scale, microns	
	ImageJ	**Dong**	**ImageJ**	**Dong**
Berea #1	No decay	-	No decay	-
Berea #2	116	-	1160	-
S1	220	220	2200	2200
S3	200	170	2000	1700
S4	190	232	1900	2320
C2	881*	881*	8810	8810

Note: * – the value of first decay length for C2 sample has been taken from (Dong, 2007), since autocorrelation from ImageJ did not reach zero.

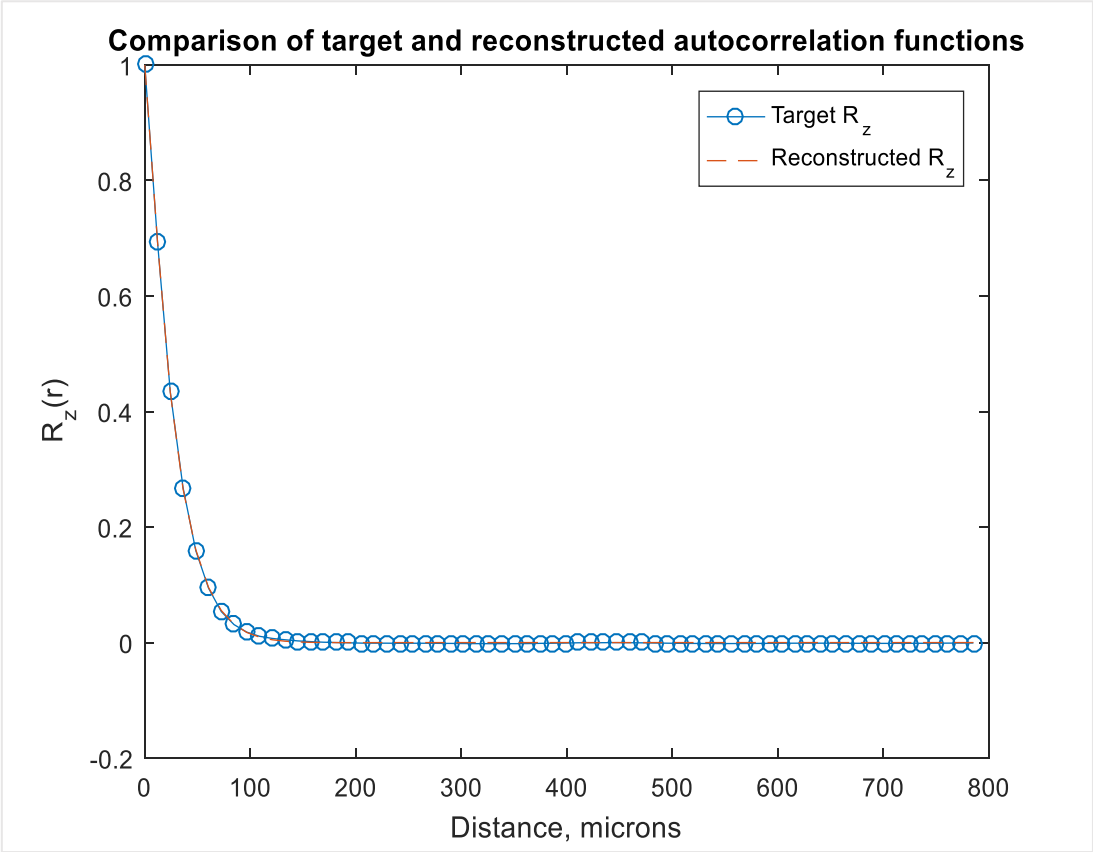
As follows from Table 6.3, for Berea #1 there is no decay in the autocorrelation function graph; therefore, this sample is not going to be considered in further permeability calculations, since without knowing the characteristic length it is impossible to choose a reasonable REV size.

The part of the MATLAB code was taken from the master thesis of Prof. Hongyan Yuan. It is included in Appendix D in a red frame and is written based on Equations. 3.7, 3.9-3.11, 3.13-3.20 from Chapter 3. The reconstruction begins with generation of a continuous 3D uncorrelated Gaussian field. Then it passes through the linear one-cut filter that is constructed from the target φ_1 and p_{expt}. Furthermore, condition of having real y_{LMN} requires to consider eight possible realizations. Finally, adding the high frequency threshold can help to avoid using of high frequencies, to improve smoothness of the obtained reconstructed images, as well as to decrease the runtime.

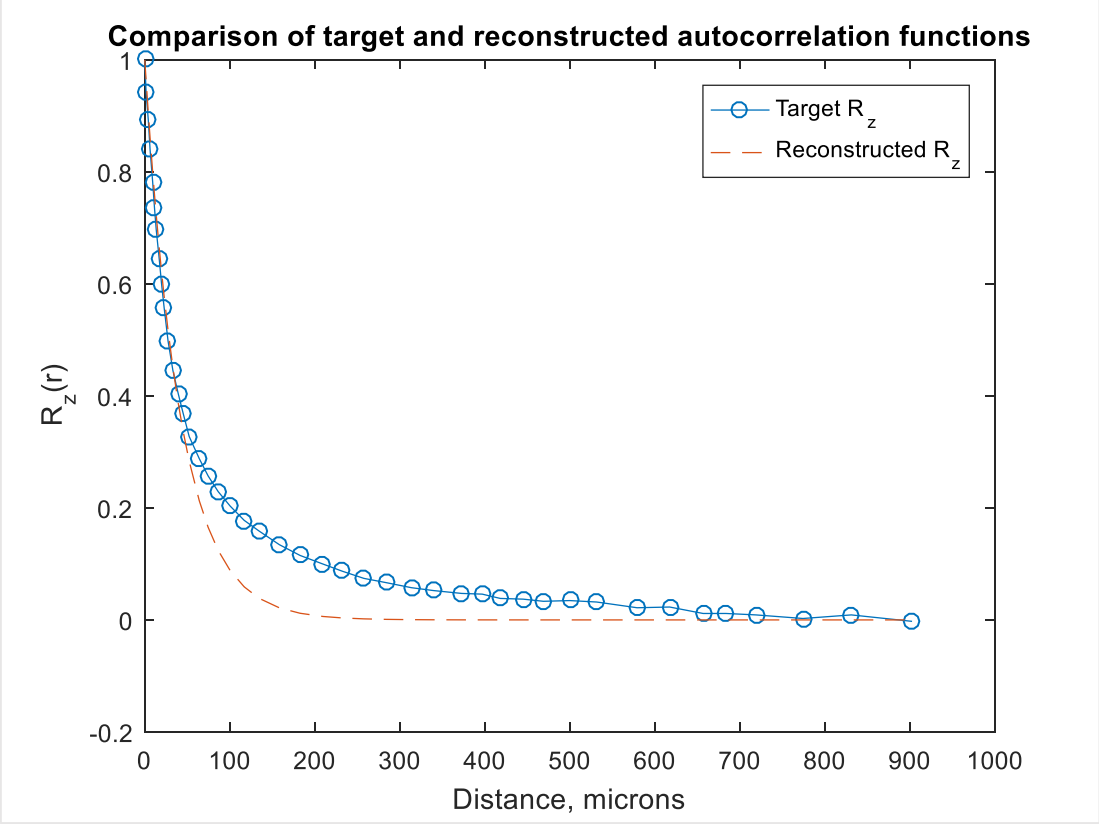
As an output, reconstructed cross sectional images and a plot that indicates a fit between target and reconstructed statistical functions can be obtained.

Comparison of target p_{expt} and reconstructed p_{fit} autocorrelation functions for Berea #2, S4 and C2 samples is shown in Figure 6.14. The comparison for remained samples is presented in Appendix A.

S4 sample



C2 sample



Berea #2 sample

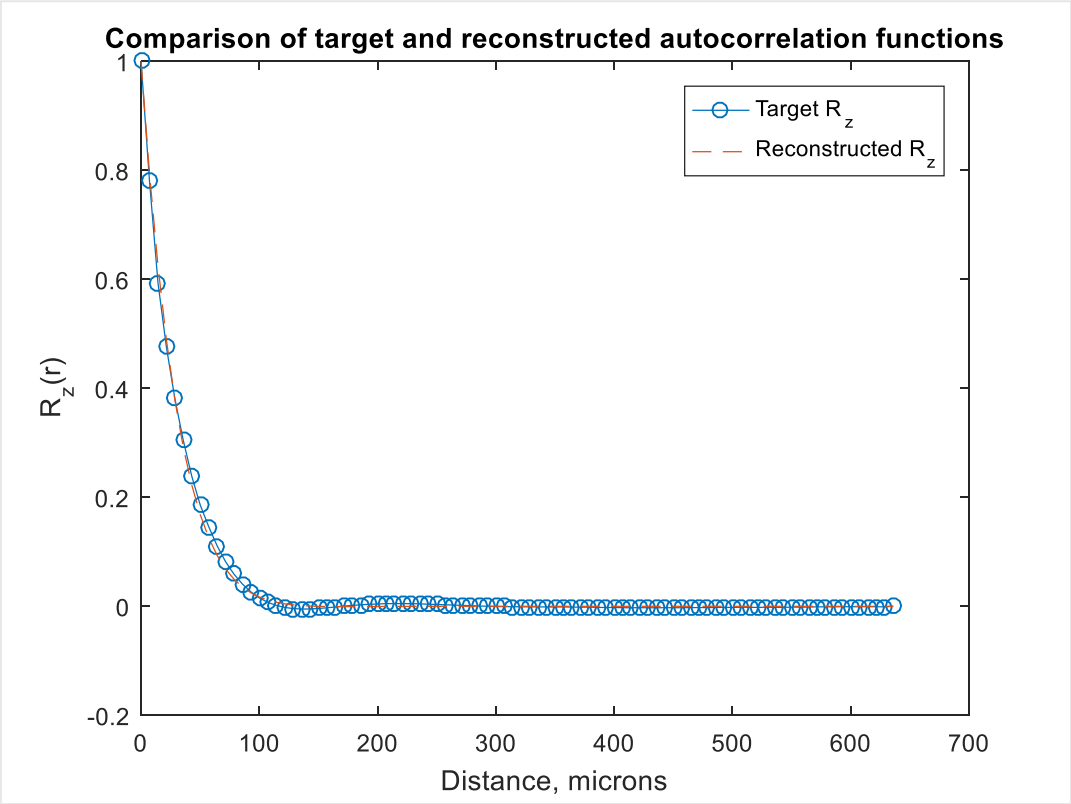


Figure 6.14. Comparison of target and reconstructed autocorrelation functions for the chosen micro-CT models

As it can be seen from Figure 6.14, the difference between target p_{expt} and reconstructed p_{fit} functions for Berea #2 and S4 sandstone samples is small, suggesting a good quality of the reconstruction procedure. However, for C2 it is possible to observe a significant mismatch between functions. This can be an indicator of either non-successful reconstruction or failure of the selected analytical function to capture complex carbonate structure properly. In addition, for the reconstructed case, correlation function reaches zero much earlier than for a real case meaning that if correlation length is used for estimation of REV, the latter will be significantly lower than it should be in the reality.

Dr. Hiroshi Okabe obtained similar mismatch between target and reconstructed cases in his PhD thesis (Figure 6.15), although he used another approach of multi-point statistics to obtain reconstructed images. As the researcher mentioned, this discrepancy might be due to strong heterogeneity of carbonate samples. The autocorrelation function can be reproduced only if the lag distance is small, and improvements are required to capture longer correlation lengths (Okabe, 2004).

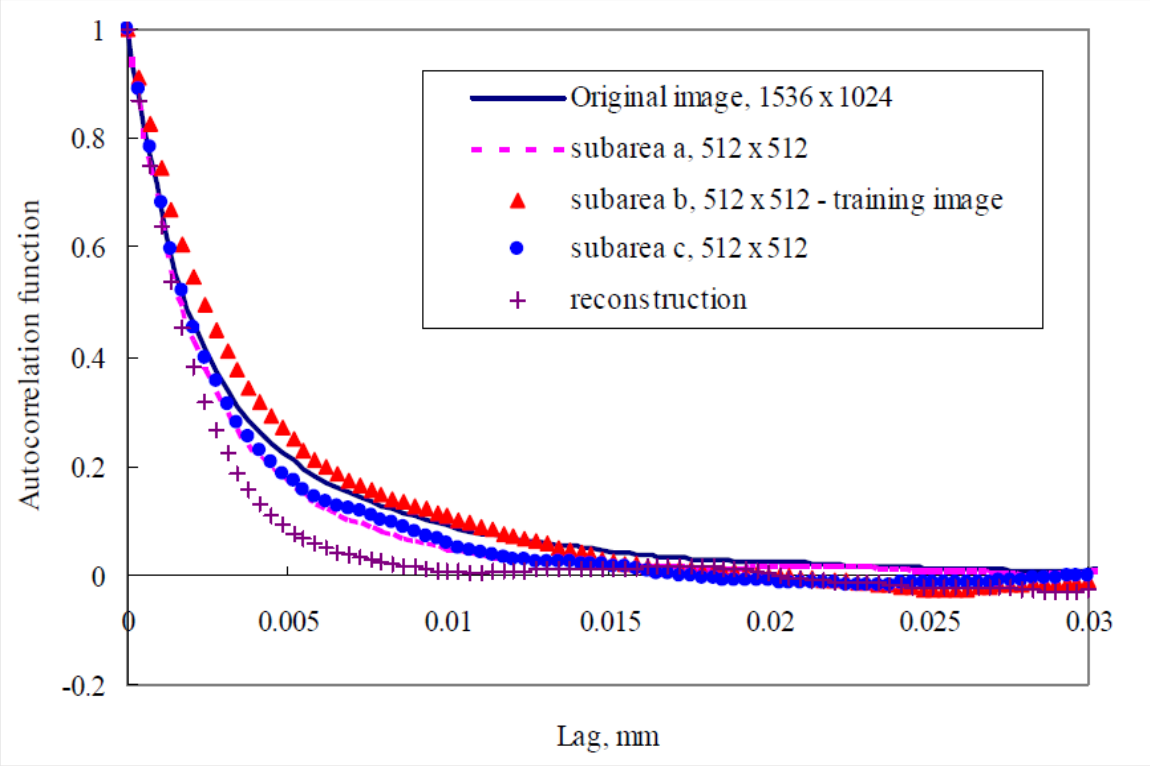


Figure 6.15. Void-void correlation functions of a carbonate rock: the reconstructed structure is compared with those from 2D thin sections incl. the original (target) image (taken from Okabe, 2004)

Finally, the parameters ξ, d, r_c of the analytical correlation function have been found based on the best-fit condition given by Equation 3.17. Table 6.4 contains values of these parameters for the models.

Table 6.4. Values of correlation length, cutoff scale and domain scale for all samples

Sample	Parameters, microns		
	ξ	r_c	d
Berea #2	31.45	3.79e-04	2.99e+02
S1	60.76	2.78e-03	4.74e+02
S3	27.40	2.14	7.17e+03
S4	26.96	2.35	3.54e+02
C2	7.57e-05	41.55	2.53e+03

As it has been discussed in Chapter 3, these parameters can help to better understand the microstructure analysis. For instance, based on the cut-off length values, Berea #2 and S1 samples should have the greatest pore interconnection, whereas C2 stack is characterized by the lowest one. Since permeability is directly related to the pore connectivity, most likely, Berea #2 and S1 samples should have the highest permeability among others, S3 and S4 – medium permeability, and C2 – the lowest one.

Since the input models are quite big, it is decided to reconstruct only their fragments. The size is chosen according to the concept of REV, and it is discussed in more details in the next subchapters.

It is possible to compare 2D reconstructed cuts with the original image, and their porosity values. Figure 6.16 displays the comparison between target image and some of cropped 2D cuts (the latter ones are post-processed, i.e. after median filtering and opening dilation-erosion procedure; pores are shown in white).

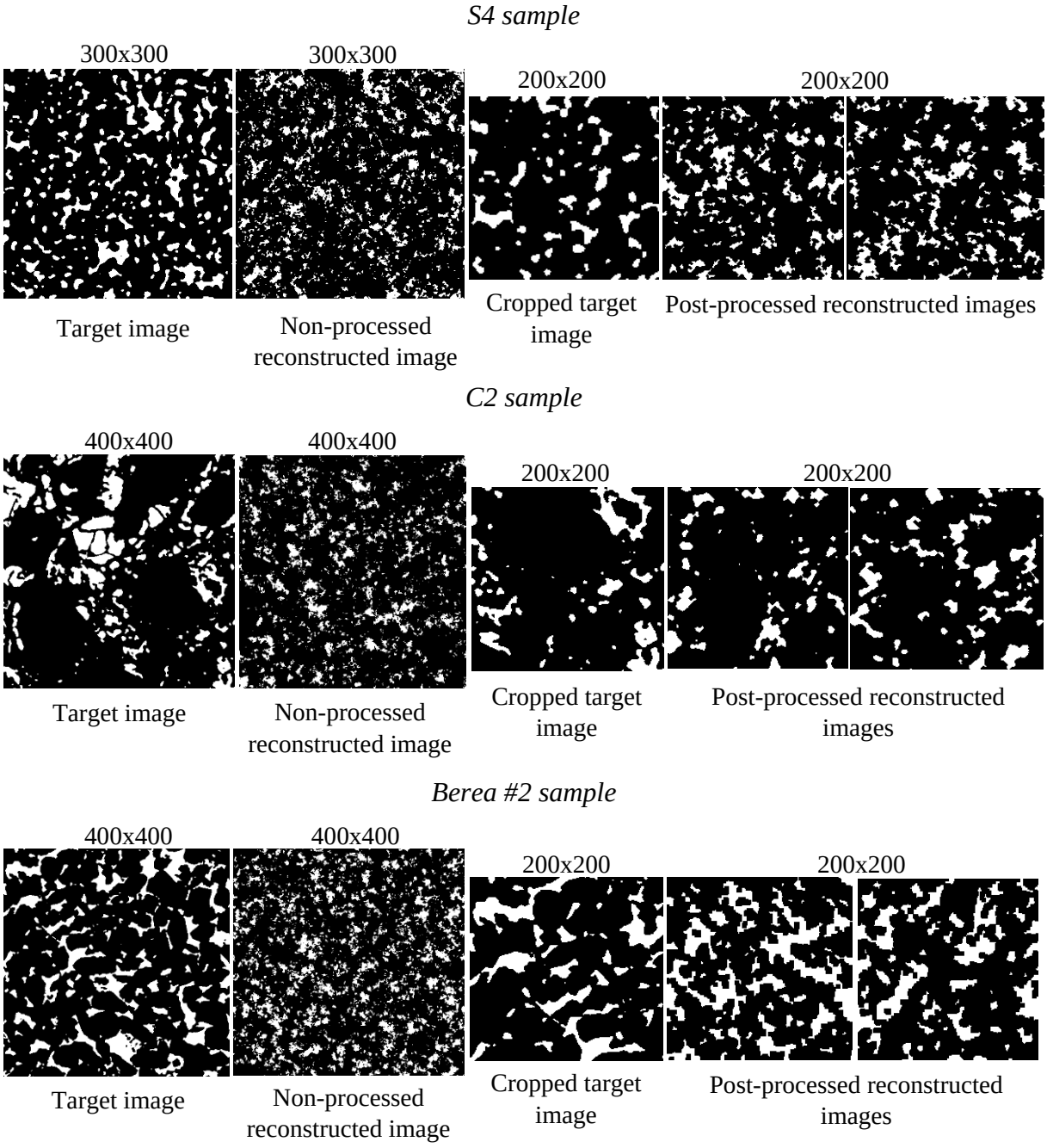


Figure 6.16. Comparison of original micro-CT images with randomly chosen 2D cuts of the reconstructed media

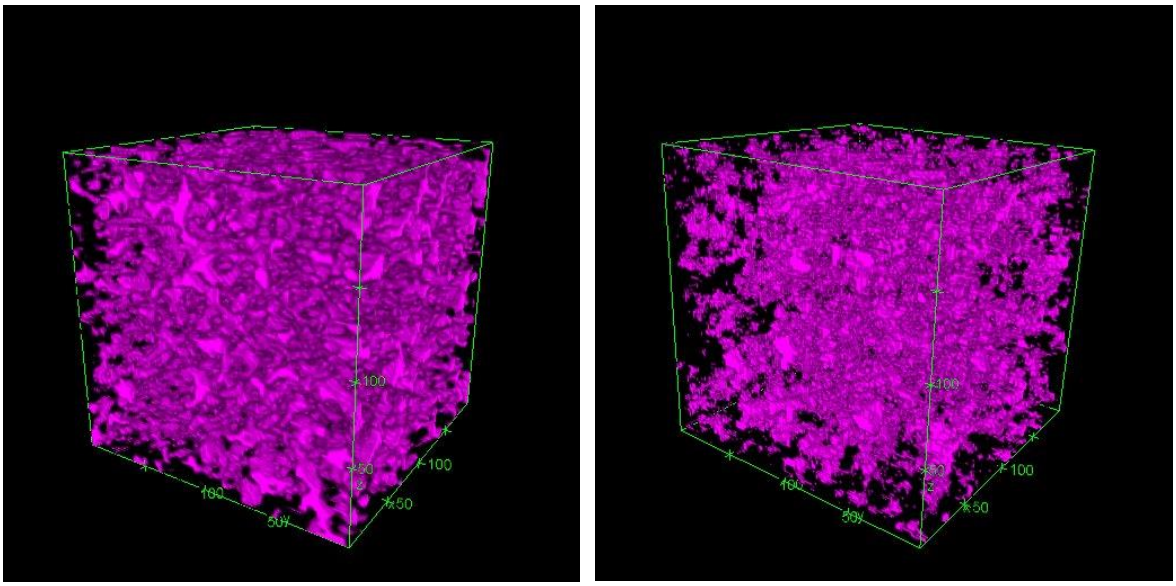
Table 6.5 shows difference between porosities for target and reconstructed images. The reconstruction results for remained samples are placed in Appendix A.

Table 6.5. Comparison of porosity for the target and reconstructed images

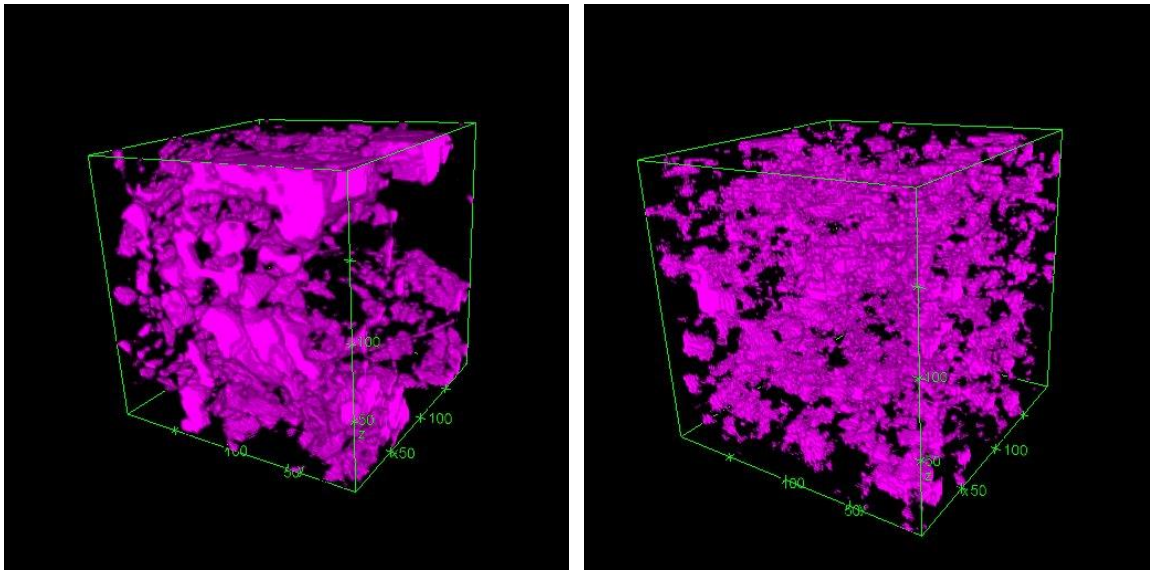
S4		**C2**		**Berea #2**	
Image #	**Porosity, φ [%]**	**Image #**	**Porosity, φ [%]**	**Image #**	**Porosity, φ [%]**
Target image	17.13	Target image	16.87	Target image	19.05
Micro-CT cropped image sequence	17.13	Micro-CT cropped image sequence	9.54	Micro-CT cropped image sequence	19.65
Reconstructed sequence	17.00	Reconstructed sequence	12.93	Reconstructed sequence	23.74

In order to check whether stochastic reconstruction provides a reliable representation of target porous media, visual inspection has been done for initial and reconstructed models respectively. Indeed, visual comparison of 3D cubes shows that the reconstructed microstructure does not provide an exact representation of the true samples' microstructure, since it contains a bigger number of smaller pores, as Figure 6.17 indicates.

S4 sample



C2 sample



Berea #2 sample

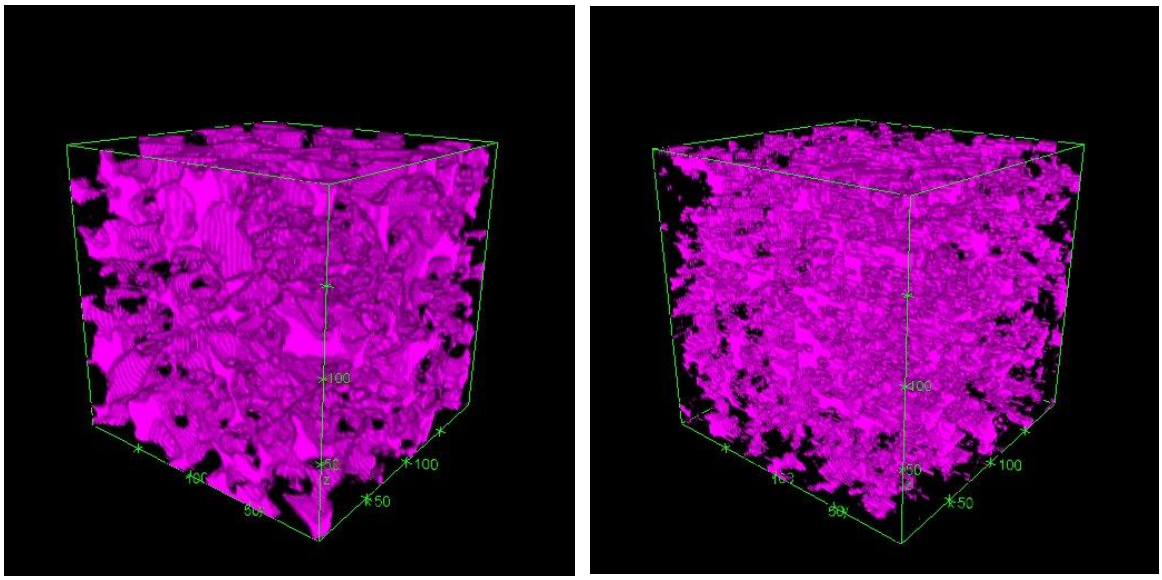


Figure 6.17. The pore space of original (left) vs. reconstructed (right) for three analyzing cases; pores are shown in magenta color

In general, from Figure 6.16 and Figure 6.17 it is possible to see that reconstructed cases have identical pore distribution within the post-processed 2D cuts and 3D cubes for sandstone samples, although for high permeable Berea, pores seem to be smaller in the case of stochastic reconstruction. Non-processed images, however, look almost identical for all cases, so that one may agree on how important the post-processing procedure is. It is also possible to notice that the C2 reconstructed sample is characterized by more homogeneous pore distribution compared with the original model.

In some cases, a stochastic reconstruction using Gaussian random functions yields much lower connectivity than it is for the actual sample, and this could be a serious bottleneck of the proposed method. Although, as Øren and Bakke mentioned, the average properties of different media may agree in quite a good manner even if the visual inspection shows the opposite (Øren & Bakke, 2003). Therefore, to have more quantitative comparison between original and reconstructed models, it is suggested to use estimation of transport properties (e.g. absolute permeability, formation factor) as a reference parameter.

6.3. Permeability estimation by lattice Boltzmann method using PALABOS

6.3.1. PALABOS input: creating a DAT-file and code description

There are three main steps in permeability calculations using PALABOS (Degruyter, 2011):

- Read the geometry that is given by a stack of binary images.

On this step, the creation of an input DAT-file from these images is required in order to ensure that the code is able to capture a particular geometry. The procedure of DAT-file generation is written in MATLAB and is presented in Appendix E. It consists of two elements – the function itself that was taken from PALABOS website, and the script, which calls the function. The DAT-file contains a unique combination of zeros, ones and twos, where zero means a pore, one – a material voxel that is contacted with pore space, and two indicates an interior material voxel. Furthermore, for illustration purposes, the script visualizes the conversion of each 2D-cut image, and displays each number – 0, 1 or 2 – in a specific color. Figure 6.18 shows several examples of such a visualization. Here, zeros are shown in dark blue, ones – in light blue, and twos as a yellow color background.

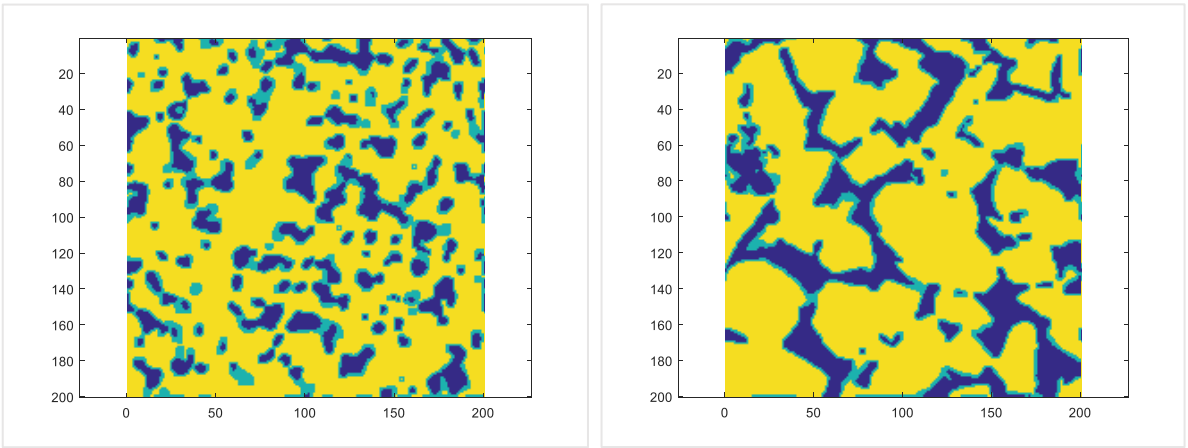


Figure 6.18. Visualization of DAT-files. S4 sample is on left, Berea #2 – on right

Once the DAT-file is created, the simulation can be started.

- Next step is related to the simulation itself. Before running a simulation, it is required to compile the code and to create the `permeability.exe` file in *bin* folder. If IDE Code::Blocks is used (www.codeblocks.org), it is important to check that all PALABOS directories, incl. sources, headers and others have been indicated on the left panel, as Figure 6.19 shows.

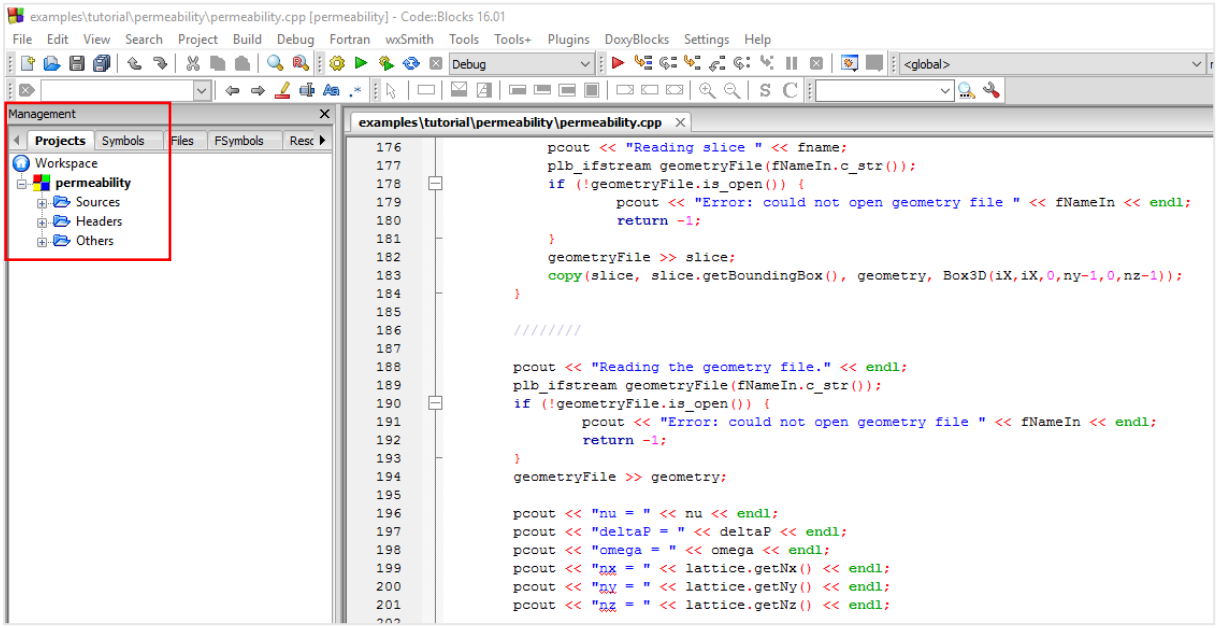


Figure 6.19. The main window of IDE Code::Blocks

In this study, a single-phase fluid flow is simulated through the given geometry of the porous structure by imposing a constant pressure at the inlet and outlet. After the initial step when a fluid velocity is zero, the flow is continued due to applied pressure gradient between inlet and outlet of the image. When the standard deviation of the average energy falls below the given threshold value (10^{-4}) or when there are no changes in *StdDev/Average* parameter, a flow time-independency is taking place, and the steady state flow is assumed to be reached.

The code is written in C++ and is presented in Appendix F. It is possible to find a general outline of the code on the PALABOS website, and therefore it will be briefly described below.

Lines 1-4: There are several include files (e.g. “palabos3D.h”) that give access to all declarations in the PALABOS project. It is important either to copy them to the project’s folder, or to specify the path where they are located.

Lines 6-7: These two declarations guarantee an access to the names in PALABOS code (`plb`), and to C++ standard library (`std`).

Lines 9-10: The simulation will be executed with double precision floating point numbers and using D3Q19 lattice scheme.

Lines 16-33: This part of the code defines the initial conditions. It is used in `porousMediaSetup` function. Initially, the fluid has a zero velocity, and a linear pressure gradient is applying in the x-direction.

Lines 35-68: This code fragment is responsible for setting boundary conditions. For all voxels read as ones in DAT-file, the bounce-back boundary conditions are defined, while for twos no dynamics boundaries are using, as it was discussed in Chapter 4. For inlet and outlet, Dirichlet boundary conditions with fixed pressure difference are setting by default.

Lines 70-133: Several output files are created. In particular, several GIF-files can be generated to capture the velocity distribution within different slices (lines 70-93), as well as a final VTK-file, which can be visualized by ParaView program afterwards (lines 96-105). The permeability is calculated based on the Darcy's law applying to obtained velocity data (lines 107-133).

Line 135-231: The main part of the code. According to lines 141-148, six input parameters are required – input DAT-file, a directory where outputs will be saved, number of cells in x-, y- and z-directions and pressure difference between inlet and outlet. If some inputs are missing, the code will give a warning notice. After each 500 iterations, PPM-files are written, and after finishing simulations, a VTK-file showing velocity distribution is generated.

After creating exe-file, the simulation can be run through the Command Prompt. As it was mentioned before, it is necessary to specify six input parameters to start calculations. Typical view of the Command Prompt with input parameters is presented in Figure 6.20. Here, if required, it is possible to change the directories by commands *chdir /d* (change disk), or by *cd* to change folders within the same disk.

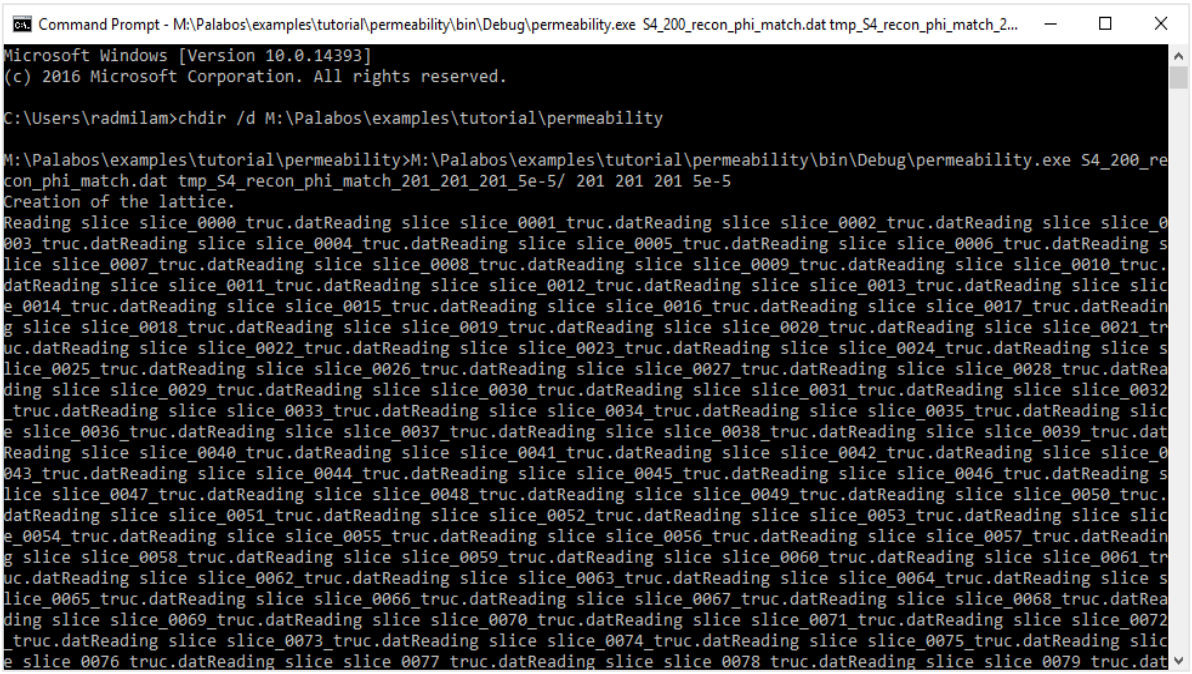


Figure 6.20. Typical view of the Command Prompt while running PALABOS simulations

- Finally, it is required to calculate permeability based on the Darcy's law as Equation 4.9 indicates. Considering that in PALABOS all physical quantities are given in lattice units for convenience, to obtain permeability in physical units, special conversion has to be applied according to the Formula 4.13.

Furthermore, as it has been mentioned in Chapter 4, applying the Darcy's law requires several assumptions to be valid (in particular, a flow laminarity is the most important one). To ensure that flow is laminar, it is possible to check whether the permeability stays constant when the applied pressure difference varies. In addition, the relaxation parameter (τ) has been chosen to be equal to single l.u. (or $\nu = 1/6$ equivalently) to minimize the effect of viscosity on permeability. This choice allows to obtain the maximum accuracy by means of the simplified BGK collisional model and to avoid the use of more complex cases, like the multiple-relaxation-time (MRT) model.

As it was discussed in Chapter 4, in some cases when the flow is not stable, it is required to have periodic boundary conditions in order to apply the lattice Boltzmann method. However, PALABOS does not have this type of boundary conditions. Therefore, it is necessary to mimic them by adding a thin layer of void space as a frame for the used models. It has been defined that one lattice cell is equal to one voxel, so the size of this frame should not exceed 1-2 lattice

cells, since it should be the same order of magnitude as a throat radius which is normally no more than 5-10 microns.

Figure 6.21 compares 2 cases when there is no thin layer framing the input images of S4 sample and when a thin layer of void space is applied. For the case of having a void frame, the standard deviation over average energy (*StdDev/Average*) parameter is decreased with number of iterations, and stable and time-independent flow can be reached after 5000 iterations. However, for the case of no periodic boundaries, the solution is unstable, and the parameter of *StdDev/Average* is increasing all the time, meaning that steady-state flow, most likely, is not achievable even after 30000 iterations.

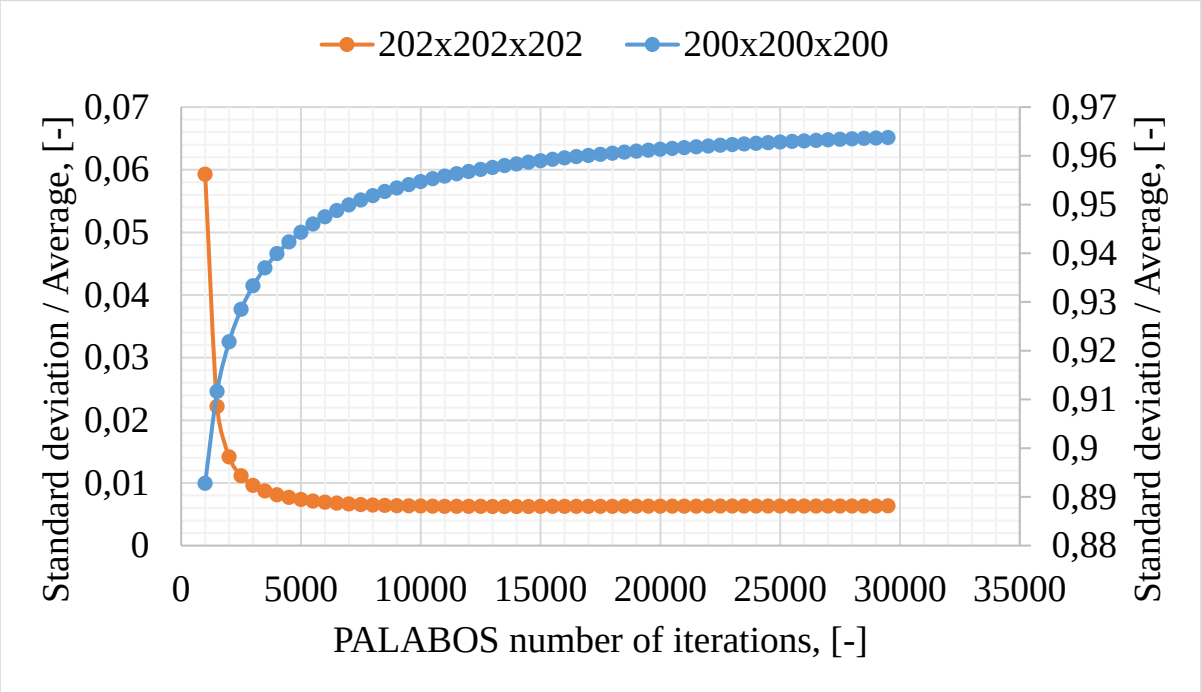


Figure 6.21. Comparison of the StdDev/Average parameter for 200x200x200 and 202x202x202 cases in PALABOS simulations for S4 sample

Note: For Berea #2 and S1 samples it is decided to use 200 lattice cells, since for these samples the parameter *StdDev/Average* has been stabilized after the first 5000 iterations (Figure 6.22), although its value is greater than for other samples. This might be due to having a high velocity in some big pores for these high permeable samples, as 3D velocity plots clearly demonstrate (Figure 6.23).

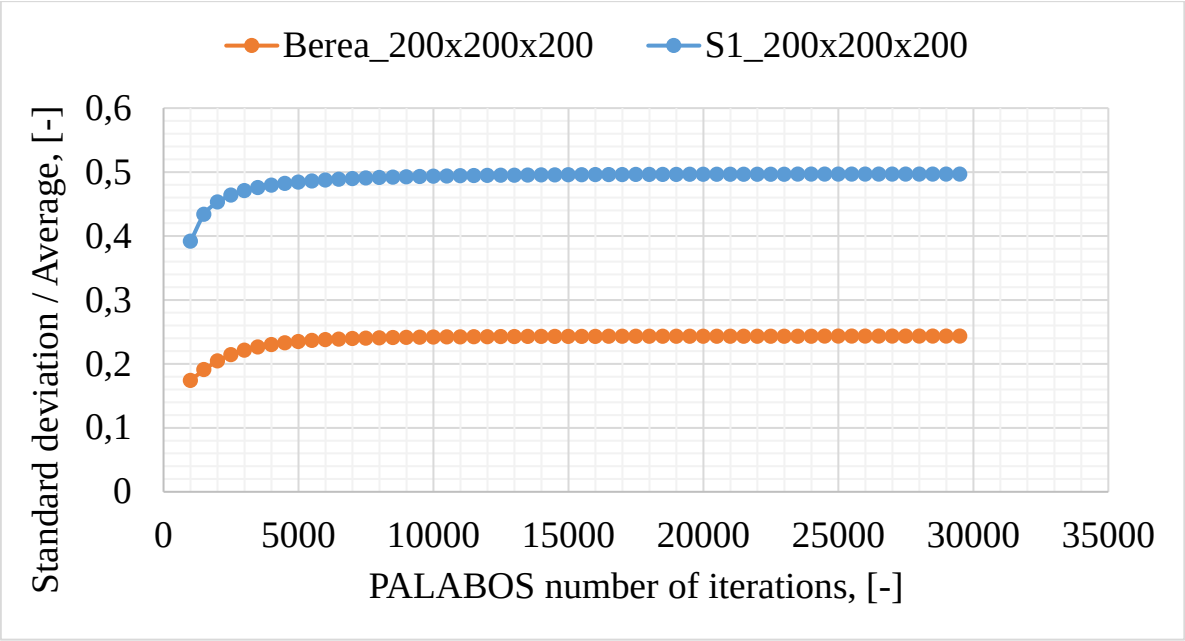


Figure 6.22. StdDev/Average parameter for Berea and S1 samples

6.3.2. Size of digital rock (representative elementary volume)

It is impossible to use the full-size input micro-CT models, since PALABOS does not allow running simulations with grid systems finer than 210 elements. In this case, the value of relaxation time will not be one anymore, and BGK collision operator cannot be used. Therefore, the input models have to be modified, and during permeability calculations and reconstruction process, only their cropped fragments of maximum 200x200x200 voxels can be used. However, it is necessary to check whether this size of analyzed samples is enough according to the REV concept discussed in Chapter 4. Table 6.6 summarizes the obtained results for all given models.

Table 6.6. Comparative table for REV values

Sample	Size of cube side (s_1), pixels	Resolution, µm/pixel	Size of cube side (s_1), µm	First decay length, µm	Characteristic length ≡ REV cube side, µm	Delta (L-s_1), µm
Berea #1	200*	5.42	1084	No decay	Undefined	-
Berea #2	200*	5.345	1069	116	1160	91
S1	200*	8.643	1728.6	220	2200	471.4
S3	200*	9.1	1820	200	2000	180
S4	200*	8.960	1792	190	1900	108
C2	200*	5.345	1069	881	8810	7741

Note: * – the value has been chosen taking into account PALABOS computational limits.

One can see from Table 6.6 that for S3, S4 and Berea #2 samples, the chosen 200^3 voxel size is seemed to be sufficient, since the difference between this constrained size and the necessary REV is quite small, and theoretically it is able to find statistically meaningful transport

properties using these cropped models. For S1 the difference is bigger, but still it might be possible to obtain reasonable results. However, for C2 sample, even the initially given model size is far less than the required REV, so it is most likely that there will be serious variance between LBM and experimental permeability values for this case.

6.3.3. Verification of having a laminar flow

To be ensure that the flow is laminar, it is possible to check whether the permeability stays constant when the applied pressure gradient has been varied (Wang, et al., 2015). In the present work, this has been done by running each simulation several times with different Δp. Due to the computational limits, only two input micro-CT models have been checked. Table 6.7 contains the comparison of three different cases for them.

Table 6.7. Three different cases for checking flow laminarity (S4 and Berea samples)

S4 sample				
Case	**Number of grids**	**Pressure difference, l.u.**	**Output from PALABOS**	**Permeability in physical units, mD**
1	202	5e-04	Average velocity = 5.39e-08 Grad P = 2.49e-06 Permeability = 0.00361072	293.72
2	202	5e-05 (base case)	Average velocity = 5.39e-09 Grad P = 2.49e-07 Permeability = 0.00361027	293.69
3	202	5e-06	Average velocity = 5.39e-10 Grad P = 2.49e-08 Permeability = 0.00361023	293.68
Berea #2 sample				
Case	**Number of grids**	**Pressure difference, l.u.**	**Output from PALABOS**	**Permeability in physical units, mD**
1	200	5e-04	Average velocity = 6.99e-07 Grad P = 2.51e-06 Permeability = 0.0463433	1341.56
2	200	5e-05 (base case)	Average velocity = 6.99e-08 Grad P = 2.51e-07 Permeability = 0.0463082	1340.54
3	200	5e-06	Average velocity = 6.99e-09 Grad P = 2.51e-08 Permeability = 0.0463052	1340.45

Table 6.7 clearly indicates that while varying the pressure difference in 100 times magnitude, permeability remains constant for both cases. This indicates that the main assumption of having a laminar flow is satisfied under the PALABOS simulations.

6.3.4. Results and 3D velocity plots for original and reconstructed cases

Table 6.8 summarizes the results of permeability calculations for both input micro-CT models, and reconstructed media. From the first part of this table, it is possible to conclude that LBM simulations are able to predict absolute permeability successfully with an average overestimation factor of 1.03 and 1.08 for experimental measurements and networks respectively. The greatest deviation, as expected, has been observed for C2 carbonate sample.

However, for the reconstructed cases, several models do not predict permeability in a good manner. This is mostly true for high permeable samples of Berea #2 and S1, as well as for C2 carbonate. The best results have been obtained for the medium permeable sandstone samples, like S3 and S4.

Table 6.8. Final table showing computed permeability values for all studied samples

Micro-CT original models:

Sample	Size	# of lattice cells	Output from PALABOS, lattice units					LBM_k, m^2	Comparison							
			Δp	Grad p	Visc.	v_avg	LBM_k		LBM_k, mD	Experiments and network, mD					Rel.err 1	Rel.err 2
										k_x	k_y	k_z	k_avg	k_net		
S4	200^3	202*	5e-06	2.49e-08	0.167	5.39e-10	3.6e-03	2.87e-13	293.7	273	289	215	259	169	1.13	1.74
C2		201*	5e-06	2.5e-08		2.59e-10	1.7e-03	4.92e-14	49.9	38	161	18	72.3	158	0.69	0.32
B_#2		200*	5e-05	2.51e-07		6.98e-08	4.6e-02	1.34e-12	1340.5	1360	1304	1193	1286	1111	1.04	1.21
S1		200*	5e-05	2.51e-07		3.25e-08	2.2e-02	1.64e-12	1666	1969	1752	1312	1678	1486	0.99	1.12
S3		202*	5e-06	2.49e-08		5.13e-10	3.4e-03	2.85e-13	288.7	143	420	109	224	281	1.29	1.03
														Average	1.028	1.084

Note: * – the value has been chosen taking into account PALABOS computational limits; Rel.err_1 is the ratio of LBM_k over k_avg, while Rel.err_2 is the ratio of LBM_k over k_net

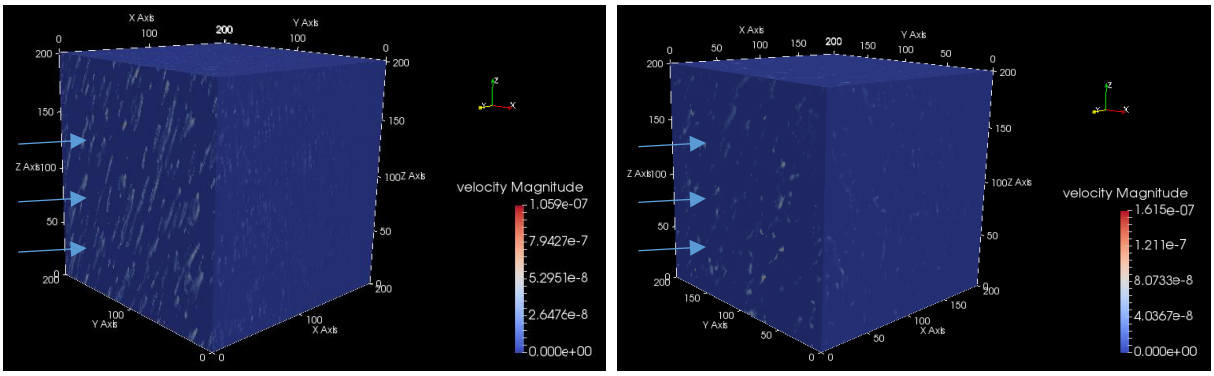
Reconstructed samples and comparison between original & reconstructed models:

Sample	Size	# of lattice cells	Output from PALABOS, lattice units					LBM_k m^2	Comparison						
			Δp	Grad p	Visc	v_avg	LBM_k		LBM_k, mD	LBM_k_micro, mD	k_avg, mD	k_net, mD	Rel.err 1	Rel.err 2	Rel.err 3
S4	200^3	201*	5e-06	2.5e-08	0.167	5.21e-10	3.5e-03	2.8e-13	282.5	293.7	259	169	0.96	1.09	1.67
C2		201*	5e-06	2.5e-08		4.82e-10	3.2e-03	9.2e-14	93.0	49.9	72.3	158	1.86	1.29	0.59
B_#2		200*	5e-05	2.51e-07		3.88e-08	2.6e-02	7.4e-13	753.4	1340.5	1286	1111	0.56	0.59	0.68
S1		200*	5e-05	2.51e-07		9.68e-09	6.4e-03	4.8e-13	490.5	1666	1678	1486	0.29	0.29	0.33
S3		201*	5e-06	2.5e-08		4.85e-10	3.2e-03	2.7e-13	271.3	288.7	224	281	0.94	1.21	0.97
												Average	0.922	0.894	0.848

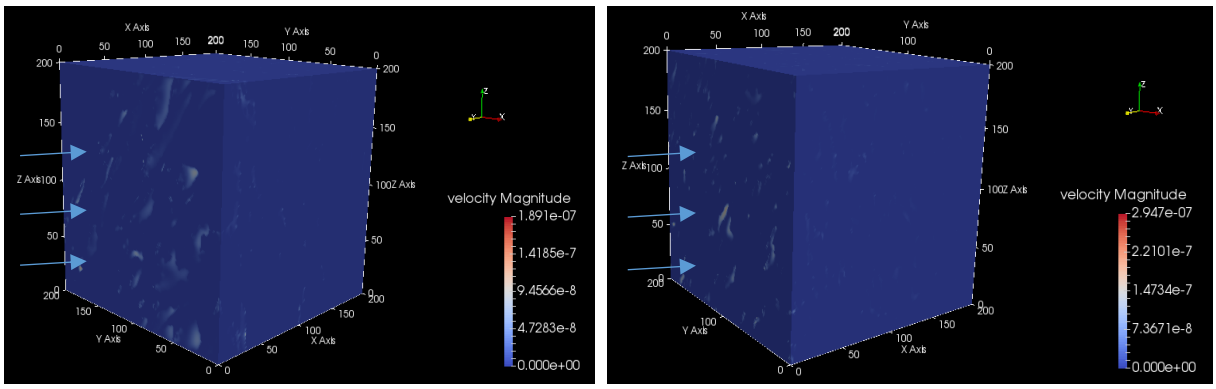
Note: * – the value has been chosen taking into account PALABOS computational limits; Rel.err_1 is LBM_k / LBM_k_micro (i.e. ratio of permeability for the reconstructed case over the permeability for the input micro-CT model), Rel.err_2 is LBM_k / k_avg, and Rel.err_3 is LBM_k / k_net

Figure 6.23 displays 3D velocity distribution plots obtained for S4, C2 and Berea #2 samples (ParaView program is used for visualization of VTK-files). It compares original and reconstructed cases for these models.

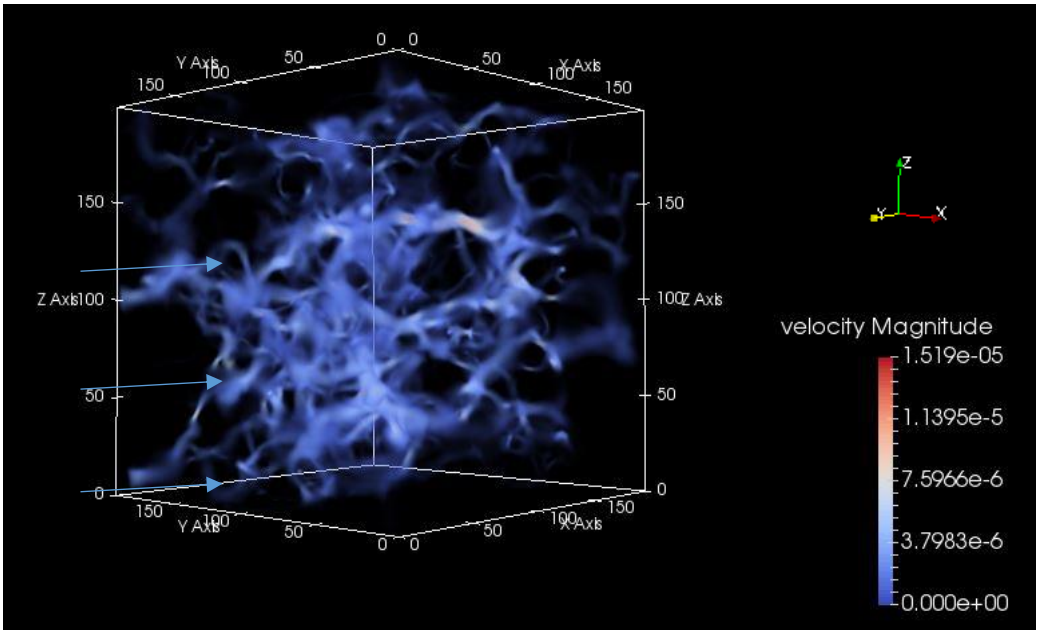
S4 sample



C2 sample



Berea #2 sample



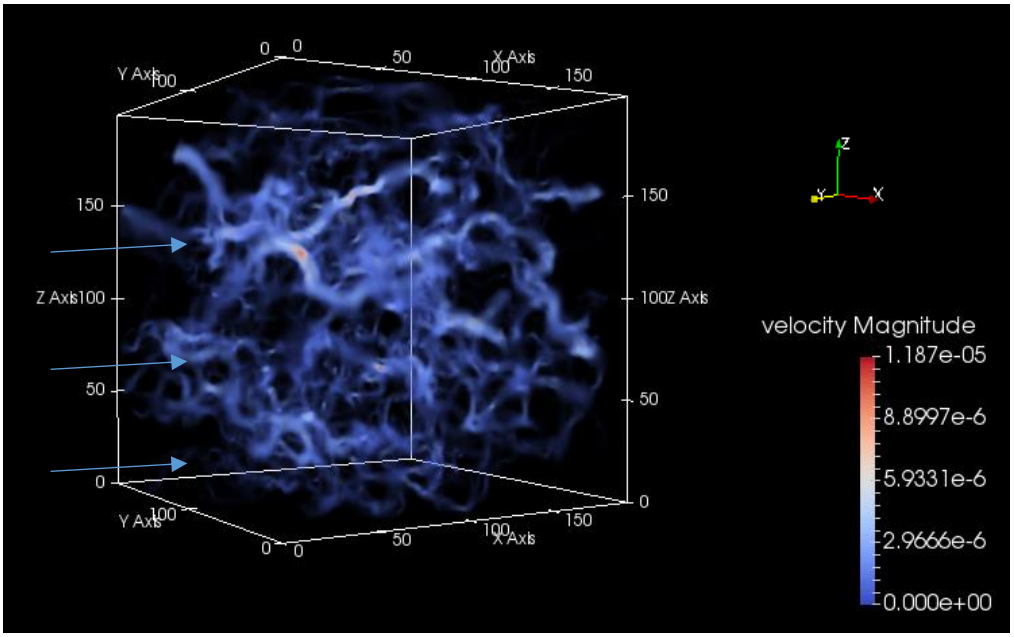


Figure 6.23. Velocity distribution for considered cases.
Real models on left (at top) and reconstructed – on right (at bottom); direction of the flow is showing by arrows

As it is possible to observe from Figure 6.23, the velocity distribution matches with the distribution of pores showing in Figure 6.17, since fluid transportation takes place in pores. Furthermore, it is important to mention that the original micro-CT models are characterized by smoother and bigger channels, while reconstructed models have more tortuous and narrow flow paths (especially, it is visible for the Berea and S1 samples).

Note: The velocity plots for S1 and S3 samples are presented in Appendix A.

6.3.5. Grid spacing and image resolution

Due to PALABOS computational limitations that referred to the single relaxation time approximation, cropping procedure has to be applied. It can be easily done in ImageJ software (Image → Crop), and it does not change the resolution of input images.

In the current work, the image resolution is modified by changing in image scale. First, the scale of input micro-CT model of 200^3 voxel size is decreased by factor 2 (Image → Scale), and then it can be rescaled back to the original size, wherein the resolution will not stay constant, and should decrease in twice. Since voxel dimension increases for the second case, the numerical resolution is deteriorated, and grid spacing increases.

Figure 6.24 indicates the comparison of two cases with different image resolutions.

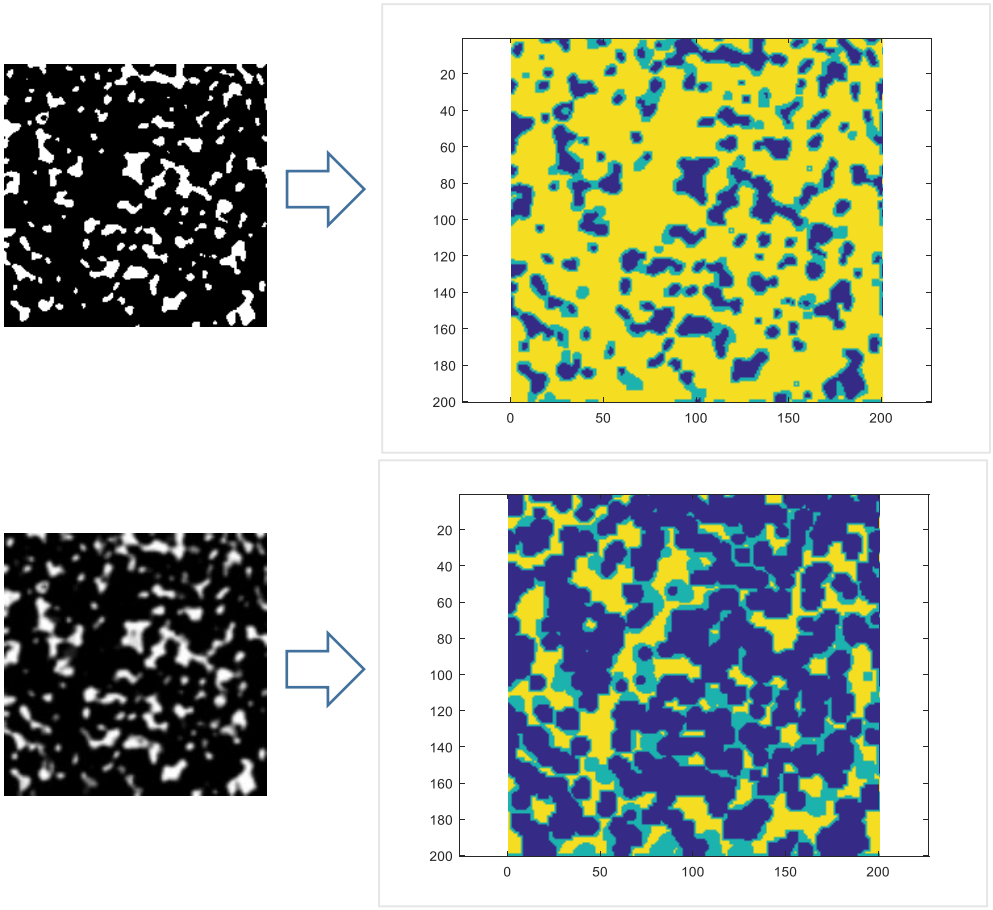


Figure 6.24. Images and DAT-files for S4 sample with two different resolutions (initially given at top and blurred in ImageJ – at bottom)

From this figure, it is clear that the original image is characterized by better image quality and more distinguishable borders between phases due to its higher resolution compared with low-resolution image showing at bottom. It is even more obvious from the comparison of DAT-files. Here, low-resolution image produces a DAT-file, where all pores become connected, and it should cause a dramatic increase in permeability, as it was discussed in Chapter 4. Table 6.9 summarizes results for both cases.

Table 6.9. Impact of the image resolution on permeability

Case	Resolution, µm/pixel	Output from PALABOS	Permeability in physical units, mD
Initially given case	8.96	Average velocity = 5.39e-09 Grad P = 2.49e-07 Permeability = 0.00361027	293.69
Blurred case	17.92	Average velocity = 5.82e-08 Grad P = 2.49e-07 Permeability = 0.0390189	3174.08

As it can be seen from Table 6.9, for a case with lower image quality, the obtained value of permeability is almost 11 times higher than in a case with original resolution. It occurs, because

when too low-resolution input has been used, the pore structure is represented by digital images in a very poor manner. This fact is directly connected with grid spacing issue, as mentioned earlier.

6.4. Cluster analysis and calculation of formation factors

In order to investigate a possible difference between original and reconstructed cases, it was decided to visualize percolation clusters using the MATLAB script *percolation.m* developed by the Department of Physics in the University of Sydney (The_University_of_Sydney, 2017).

6.4.1. Cluster analysis using 2D images

The input for the program is a square lattice of linear dimension L (for this particular case, the input is one of the 2D cut binary image, where L is its linear size in pixels). The number of lattice sites is LxL, and it can be occupied (1) or cannot be occupied (0). Each site is occupied independently from the neighbors with a given probability p, which is the porosity of the considered image, or the ratio of white pixels over their total amount. Occupied sites form clusters, in which sites may or may not be connected to their neighbors. An important parameter showing the connectivity of considered medium is so-called spanning cluster, which extends from one edge of the lattice to another.

If the probability is small (as it happens for the considered samples), then only small clusters are formed, basically, as it can be found from Figure 6.25. Here, the cluster distribution for original and reconstructed cases is compared for two different sandstones – S4 and Berea #2. As follows, from the given information, number of occupied sites in clusters is higher for more permeable Berea sample. In general, reconstructed cases are characterized by smaller number of distinguished clusters. Both cases show no presence of spanning cluster.

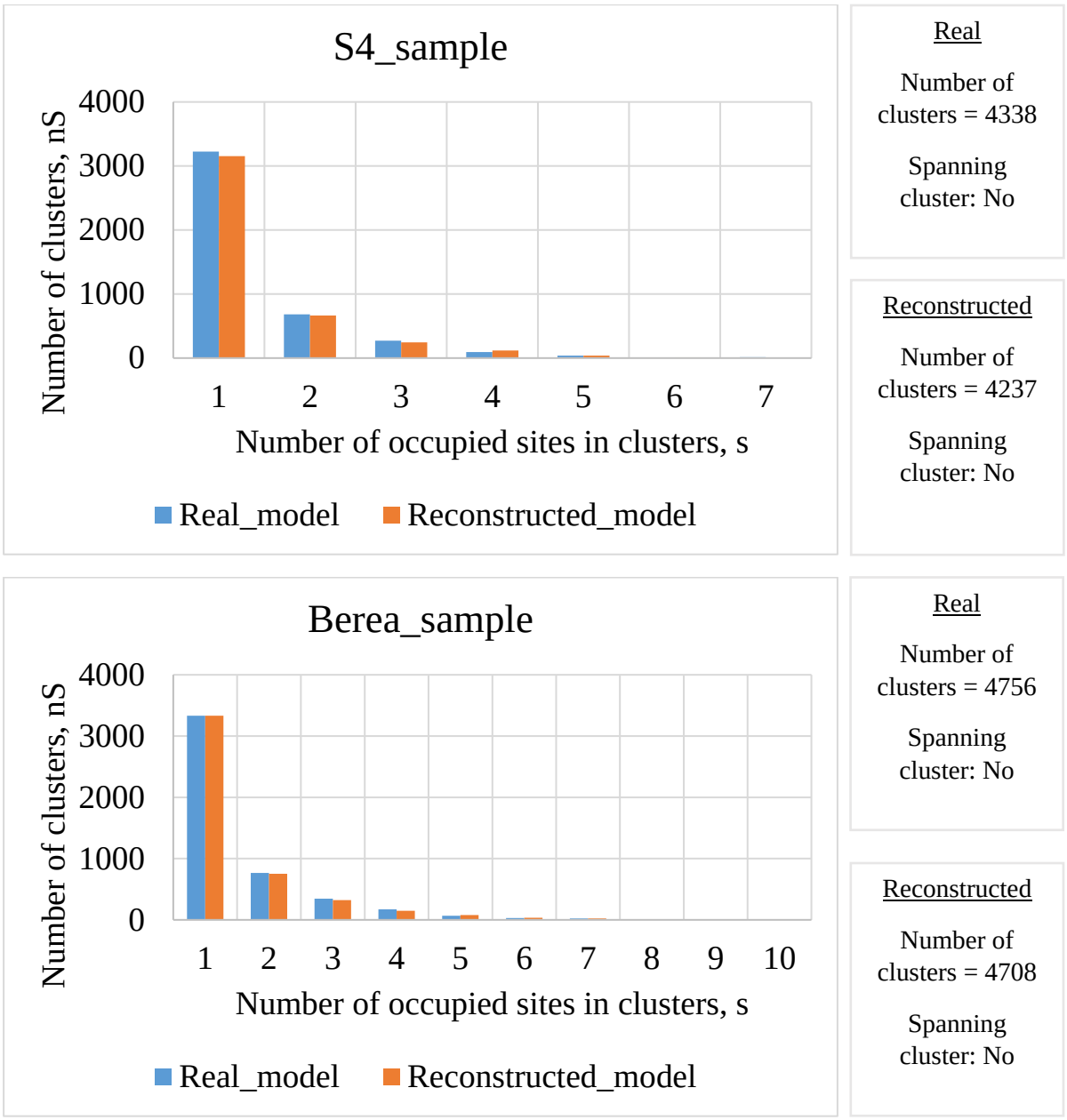


Figure 6.25. Comparison of cluster numbers
for original and reconstructed models of S4 and Berea #2 samples

Although the suggested cluster analysis is helpful as a quick estimation of reconstruction's quality, it cannot fully evaluate the ability of the reconstruction process to represent the given porous media, since only 2D images have been compared. For more detailed 3D analysis, the comparison of formation factors is proposed to be used.

6.4.2. Calculation of formation factors for the digital images

The estimation of formation factors for the digital images has been done by using of *TauFactor* MATLAB application developed in the Imperial College London (Cooper, 2015). The main

principle is to calculate the decrease in diffusive transport caused by convolution in the geometry of heterogeneous media using input microstructural image data.

The results of formation factors' comparison for original and reconstructed cases are presented in Table 6.10. As it can be observed, formation factors of original input models are in a good agreement with values obtained by Dr. Hu Dong, since the average error is less than 6 %. For C2 sample, *TauFactor* could not define the formation factor, giving *Inf* value.

Table 6.10. Comparison of formation factors and specific surface areas for all samples

#	Formation factor					Specific surface area, μm^{-1}			
	Dong	**Original**	**Reconst.**	**Rel.err 1**	**Rel.err 2**	**Measured**	**Original**	**Reconst.**	**Δ, 1e-03**
S4	82.8	81.73	149.45	0.987	1.829	24.15e-03	20.4e-03	25.6e-03	5.2
C2	169.8	-	284.61	-	-	-	12.6e-03	22.1e-03	9.5
B2	24.1	23.56	27.38	0.978	1.136	26.90e-03	24.0e-03	40.1e-03	16.1
S1	41.3	35.88	114.65	0.869	3.195	14.11e-03	12.3e-03	14.8e-03	2.5
S3	52.4	48.99	122.18	0.935	2.507	23.76e-03	20.2e-03	24.1e-03	3.9
			Average	0.942	2.167			Average	7.4

Note: Rel.err_1 is the ratio of formation factor for the original model over the value stated in Dong, 2007; Rel.err_2 is the ratio of formation factor for the reconstructed model over the formation factor for the original model; the measured values have been taken from (Rabbani & Jamshidi, 2014); Δ is the difference between SSA of reconstructed and original models

From Table 6.10, it is also possible to see that the reconstructed cases tend to overestimate formation factors significantly (by a factor of 2.2), meaning that the conductivity in a case of stochastic reconstruction is underestimated (formation factor is inverse of conductivity).

Furthermore, using the same *TauFactor* application, the surface area per volume can be computed for each case. From the table one can see that the specific surface area values for the reconstructed cases are higher than for the original cases. This is due to having more edgy borders between rock and void phases, as well as more structural complexity in general, as Figure 6.16 indicates. If permeability has been matched for original and reconstructed cases, the only way to keep the same values for k is to increase the formation factor in the numerator to compensate the increase in surface area in the denominator, according to Equation 4.17.

6.5. Pore network modeling

6.5.1. Watershed algorithm for image segmentation and pore size distribution

Figure 6.26 displays the obtained results after image segmentation (for Sherwood and Bridport images respectively).

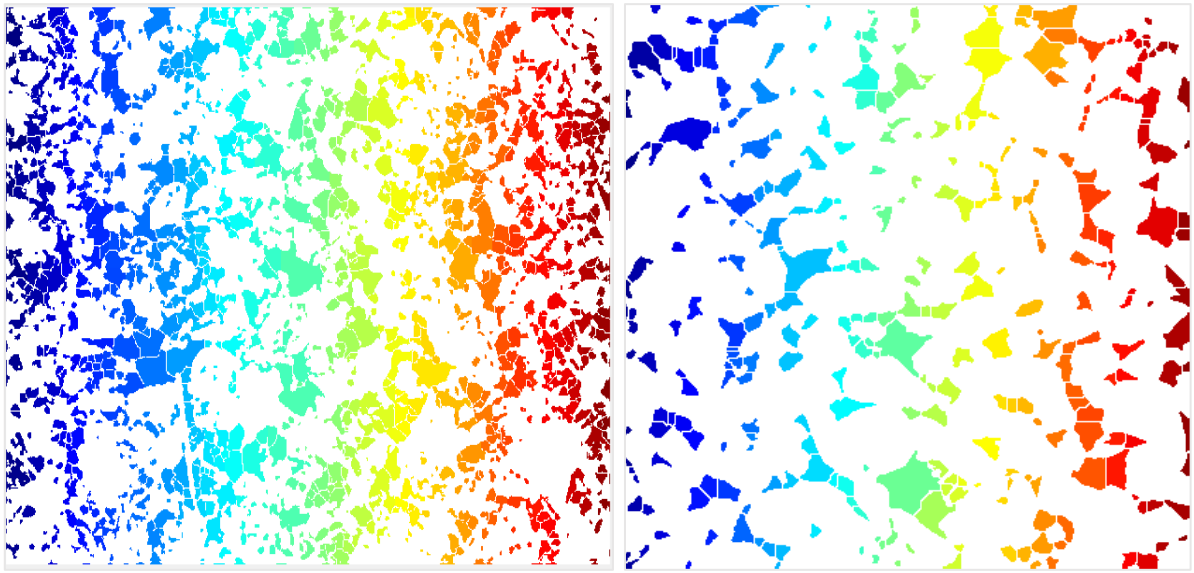


Figure 6.26. Segmented 2D sample images showing pores and throats.
MATLAB label2rgb command is used for visualization purpose

After successful segmentation procedure, it can be possible to find pore size distribution, since pores and throats are separated now. First, it is required to label each pore using `bwlabel` command in MATLAB. After that, it is necessary to check for all pixels whether the chosen element in the binary image is zero or non-zero in the distance map (i.e. if it is pore or not), and if so the pore counter is filled with this element. Afterwards, one may determine size of the obtained pores (assuming that they are circular), as well as it is able to create a histogram of relative frequency and to build a cumulative distribution function for detected pores. To obtain radius of pores in physical quantities, it is necessary to convert pixels into microns using resolution values from Table 6.1.

Figure 6.27 displays the pore size distribution obtained for input thin section images.

Image #1 (Sherwood sample)

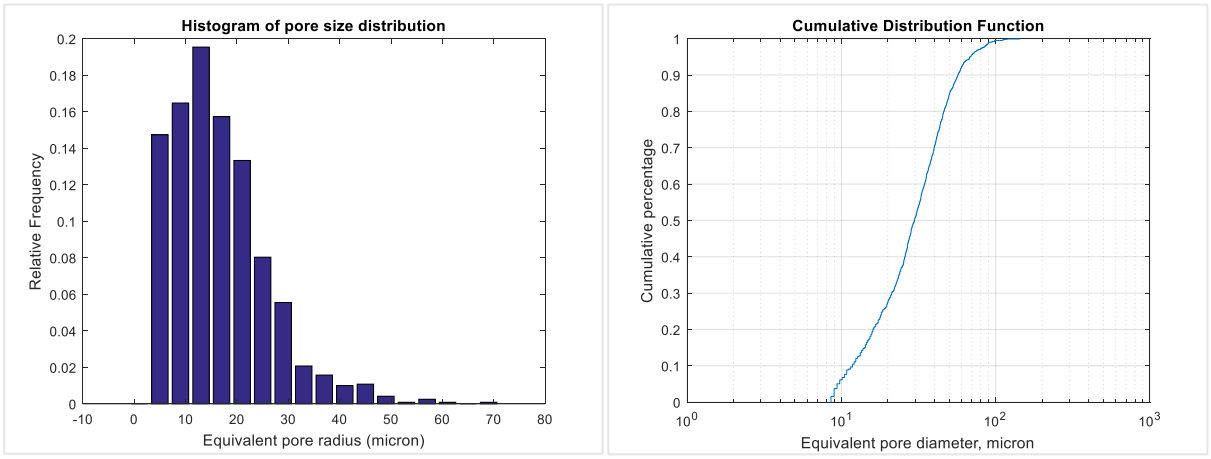


Image #2 (Bridport sample)

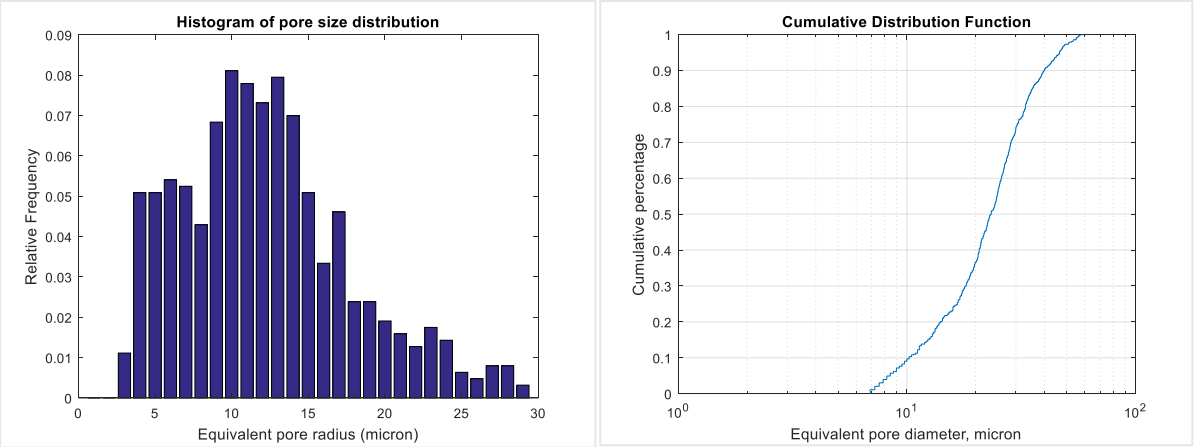


Figure 6.27. Pore size distribution obtained for segmented binary images of thin sections from the Wessex Basin

The Bridport sample can be considered as a strongly consolidated sample compared with the Sherwood sample (which has significant porosity and permeability values), and that is why its average radius is smaller.

As it was mentioned in Chapter 3, it may be possible to use empirical formula to find an average pore radius (diameter). Table 6.11 tots up the results of average pore radius estimation obtained by the two discussed methods.

Table 6.11. Values of average pore radius for analyzed images and samples

Image	**Average pore radius, microns**		
	2D watershed	**Equation 3.12**	**Abs. difference**
Sherwood	16.5	10.09	6.41
Bridport	12.2	7.67	4.53
Sample	**Average pore radius, microns**		
	2D watershed	**Equation 3.12**	**Abs. difference**
S4	28.27	16.27	12.00
C2	21.27	9.1e-05	21.27
Berea #2	21.21	19.57	1.64
S1	38.07	35.38	2.69
S3	26.82	16.48	10.34

As follows from results comparison, in general, the values of average pore radius estimated by Equation 3.12 are smaller than those obtained by the watershed algorithm, although for several samples the difference between the two methods is not so significant. The main shortcoming of the proposed watershed technique is that it uses only a single 2D image, and maybe it is not representative, especially in the case of micro-CT models. Moreover, as it has been mentioned before, the assumption of having only circular pores may not be true, and there can be pores

with other shapes. Furthermore, the observed overestimation of pore radius by watershed method for S4 and S3 samples may be related to their resolution, which is lower than for Berea #2 sample. For C2 model, it seems that both methods cannot predict the average pore radius accurately. For watershed method, it may happen due to having of several huge pores on 2D image, so the mean value in this case can be significantly overestimated. For the empirical formula, obtaining such a small radius may be related to some errors in the correlation length estimation, since as it was discussed previously it was not possible to match analytical correlation function with the original one, as Figure 6.14 indicates. The typical range of pore radius for chalk is between 0.7 and 4.3 μm (Talukdar, 2002).

It is possible to compare the obtained pore radius value for Berea sample with those from Dr. Hu Dong's thesis. There, the values of the average pore radius estimated by pore networks built using process based method and maximal ball algorithm are 19.17 and 15.36 microns respectively (Dong, 2007). They are quite close to the value obtained by Equation 3.12.

6.5.2. Using of OpenPNM for pore networks extracted by maximal ball algorithm

As it has been emphasized in Chapter 5, direct LBM simulations for the multi-phase flow can be very difficult to implement due to the requirement of having bigger REV and possible BGK solution instability. Therefore, the pore network modeling can be a reliable option to predict permeabilities of each flow component.

It is possible to use the OpenPNM package (www.openpnm.org) for pore network simulations. It has five main objects: `Network`, `Geometry`, `Physics`, `Phase`, `Algorithm`, and has been coded in Python language (numerical operators are computed using NumPy and SciPy packages), which is a powerful, free object-oriented programming language. The main advantage of using Python is that no compilation of source code is required.

It is necessary to modify four input DAT-files with information about pore networks extracted by the maximal ball method from the Imperial College London to one common MAT-file, which then can be used as an input to OpenPNM software package. The import can be done directly in OpenPNM using, for example, Spyder – the Scientific PYthon Development EnviRonment. The code is presented in Appendix G, and Figure 6.28 shows the visual comparison of both networks. As it can be seen, they look more or less identical, although in OpenPNM the pore sizes are fixed, and only color displays the pore size distribution. However,

in Rhino software (Imperial College), it is possible to vary both sizes and colors for showing different pore dimensions. The color bar situated below network images in Figure 6.28 is used to display color range for the networks extracted by Dong.

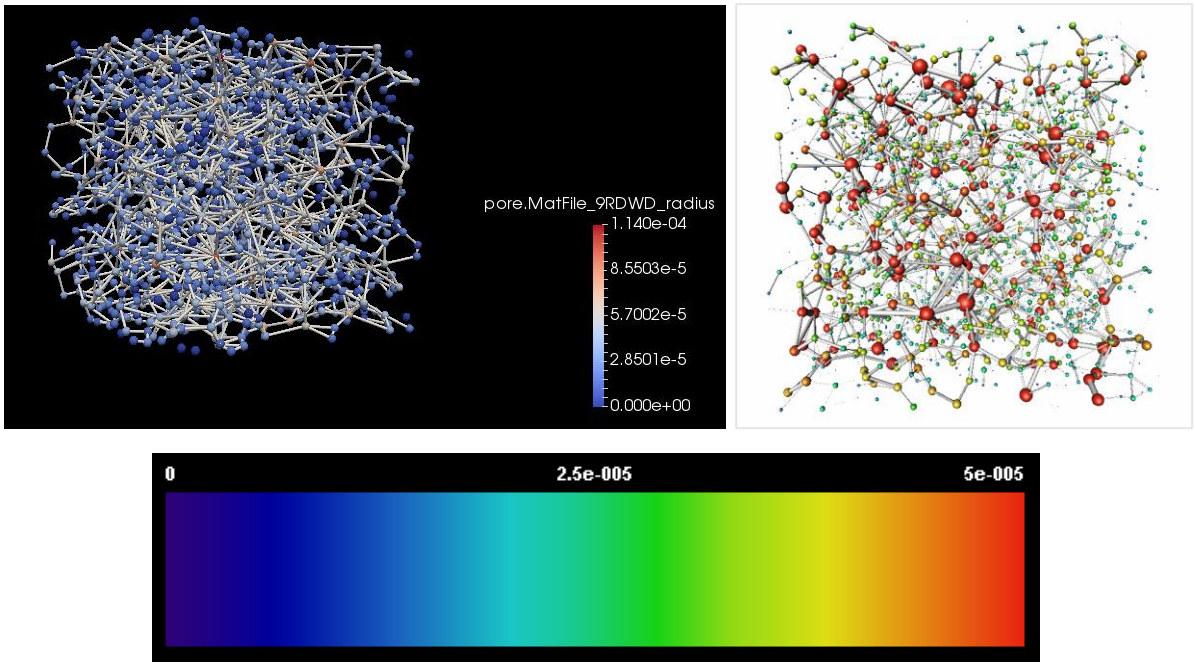


Figure 6.28. Comparison of pore networks for S1 sample (extracted in OpenPNM – on left, from Dr. Dong's thesis – on right)

After network import, it can be possible to use obtained networks to find capillary pressure and relative permeabilities. For example, it is shown in Appendix G that by using only 12 strings the mercury intrusion porosimetry simulation for regular lattice networks can be performed. The resulting curve is presented in Figure 6.29.

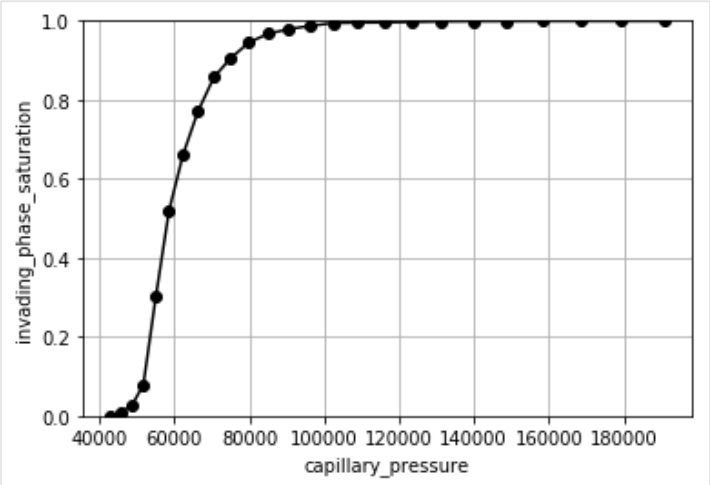


Figure 6.29. Capillary pressure curve obtained in OpenPNM for the mercury intrusion porosimetry using a regular lattice network

Chapter 7.
Discussions and Conclusions

7.1. Discussions

Although a brief analysis and interpretation of obtained results has been already given in the previous chapter, here, the more generalized discussions with focus on possible source of errors are presented. For convenience, they are divided by the subchapters.

7.1.1. Image Analysis

In the current thesis, several thin section images and micro-CT models had been analyzed. The given thin sections were in color, since they were obtained by blue epoxy impregnation. The first step was to propose a suitable way of their binarization. It was required to have binary images, since for digital images the porosity was suggested to be defined as the ratio of white or black pixels over the total number of them. The main challenge for thin sections was to avoid counting of cement / organic intrusions as a part of porous media. It was shown that using of double thresholding for gray images or multiple dimension thresholding for colored images provided a fast and reliable technique for realistic porosity estimation from binary images. The difference between the obtained results and measured porosity values was very small. The same was true for given micro-CT sequences (although for them binarization was not needed) – the proposed method of porosity estimation showed a good consistency with given porosity values from the Imperial College London with less than 1 % error.

Possible challenge: In some cases, if there is microporosity and clay minerals are presented, having only black and white colors within images can be not enough.

7.1.2. Stochastic reconstruction and analysis of autocorrelation graphs

The proposed one-cut GRF reconstruction had been considered as a simple method of obtaining real pore structure with relatively easy implementation in MATLAB. Two first statistical moments – one- and two-point correlation functions used as matching criteria during the stochastic reconstruction. Some significant improvements of reconstruction process had been achieved by applying the opening procedure and other post-processing techniques. Although, as further analysis had been shown, it was impossible to reproduce exactly the same porous structure, however, several properties, like permeability and porosity, were able to be captured very accurately for some samples by using these modified pore media.

Moreover, it had been noticed that the autocorrelation plots contained important information about pore connectivity and required size of REV, and could be used as a quick analysis of input micro-CT data.

In particular, it was shown that the carbonate sample could not be reconstructed in a good manner, due to its heterogeneous complex structure and small size of the input model.

Possible challenge: In several cases when complex, highly structured and continuous patterns may occur, having only the void fraction (porosity) and void-phase autocorrelation function of the target medium as reconstruction criteria can be insufficient.

7.1.3. Absolute permeability estimation and checking of reconstruction quality

For accurate prediction of absolute permeability using LBM, several important modeling parameters had to be considered, e.g. the size of digital rock chosen as representative elementary volume and image resolution. It was shown that for all samples, except the carbonate, the chosen size of models was enough to be REV. The effect of decreasing in image resolution showed a huge impact on the estimated permeability, making it overestimated significantly. As practice indicates, the typical limit of acceptable resolution should not exceed 10 μm/pixel in order to get reasonable values of permeability. The role of image resolution is also important in the estimation of the average pore radius. When the watershed 2D method was used, it was discovered that samples with lower resolution (S3 and S4) had been characterized by a big overvalue of pore radius, compared with empirical formula using statistical information.

According to the obtained absolute permeabilities for the original models, the good agreement with experimental data had been observed with the average overestimation factor of 1.028. In addition, the comparison of k values for the original micro-CT models and for the reconstructed cases indicated a very good consistency for medium permeable S3 and S4 sandstones with only 3.8 and 6 % difference respectively, but with a general underestimation for high permeable samples (up to 70.6 %). The average underestimation factor was equal to 0.922 compared with LBM permeability for original models, and 0.894 in comparison with experimental measurements. However, even though porosity and permeability were matched successfully for the reconstructed models, they might be completely different porous media, since other properties could be different, e.g. velocity and pore distributions, as well as formation factors and specific surface area values, as it was shown during the analysis of reconstruction quality. It had been noticed that if porosity of reconstructed models was matched perfectly with a given value, then a significant underestimation of permeability and overestimation of formation factor took place, as it was in a case of S1 sample (490.5 vs. 1666 mD, and 114.7 vs. 35.9 for formation factors). For the Berea sample, however, the porosity of the reconstructed model was a bit overrated from the beginning (23.7 vs. 19.7 %), but it allowed to get closer permeability match (753.4 vs. 1340.5 mD), as well as almost the same formation factor value (27.4 vs. 23.6).

7.1.4. Pore network modeling

As it had been discussed earlier, the modeling of multi-phase transport properties directly on pore space images was a challenging and non-trivial task. In particular, the computational demands due to having of BGK single relaxation time might constrain these simulations to small image samples, although to predict relative properties accurately, bigger REV size were required. Therefore, it was decided to apply a pore network modeling approach, which had substituted complex real pore structures with simplified models consisting of balls and sticks.

OpenPNM is a powerful, open-sourced pore network modeling package, which was coded in Python. As it was stated in the OpenPNM website, the main aim was to provide the porous media community with a general and flexible framework for tackling various PNM problems from the same code base. The use of Python as a main programing language allows to create short and elegant codes. For instance, it was shown that to implement the mercury intrusion porosimetry simulation using a simple regular lattice network in OpenPNM, only 12 strings of code were required (for network import – only 8 strings were needed). However, due to time limits it was not possible to perform any capillary pressure & relative permeability calculations

using imported networks extracted by maximal balls method. This can be considered as an important part that needs to be improved in future.

7.2. Recommendations for the future work

The future work may include a further testing of suggested permeability estimation using more input micro-CT models with different properties. Furthermore, several efforts can be done to improve existing stochastic reconstruction procedure, e.g. by using higher order statistics as constraints. Finally, more detailed study of pore network modeling is required, for instance, different pore extraction method can be checked in addition to maximal ball algorithm. Furthermore, it can be interesting to determine relative permeabilities and capillary pressures in OpenPNM using imported pore networks and to compare the results with those from the Imperial College London database.

7.3. Conclusions

It was possible to prove that suggested digital core analysis technique could be a reliable alternative of routine laboratory experiments, at least, for widely used homogeneous sandstone samples, like Berea or Fontainebleau. For carbonate sample, however, it had been challenging to estimate main reservoir properties due to its complex heterogeneity, and therefore, more advanced methods should be applied in future, e.g. nano-CT scanning.

The most important factors, which had influence on the results, were image resolution, representative sample size and proper simulation set-up (i.e. different approaches for single- and multi-phase flow, boundary conditions, etc.).

P.S. The author is deeply convinced that this work can be considered as a good starting point for investigation of digital rock analysis at NTNU IGP. Thank you!

References

1) Alshibli, K. A. & Reed, A. H., 2010. *Advances in computed tomography for geomaterials (GeoX 2010).* 1st ed. London: ISTE Ltd.

2) Berg, C. F., 2014. Permeability Description by Characteristic Length, Tortuosity, Constriction and Porosity. *Transport in Porous Media,* Issue 103, pp. 381-400.

3) Bowen, D. & Tanner, B. K., 2006. *X-Ray Metrology in Semiconductor Manufacturing.* New York: CRC Press.

4) Bultreys, T., Wesley, D. B. & Veerle, C., 2016. Imaging and image-based fluid transport modeling at the pore scale in geological matterials: A practical introduction to the current state-of-the-art. *Earth-Science Reviews,* Issue 155, pp. 93-128.

5) Comsol_webpage, 2017. *Multiphysics Cyclopedia.* [Online] Available at: https://www.comsol.com/multiphysics/navier-stokes-equations [Accessed 24 May 2017].

6) Cooper, S. J., 2015. *TauFactor: MATLAB app,* London: Imperial College London.

7) Degruyter, W., 2011. *Geophysics: compute the permeability of a 3D Porous Medium.* [Online]
Available at: http://www.palabos.org/documentation/tutorial/permeability.html [Accessed 24 May 2017].

8) Dong, H., 2007. *Micro-CT imaging and pore network extraction,* London: Imperial College London.

9) Dong, H., Touati, M. & Blunt, M. J., 2007. *Pore network modeling: analysis of pore size distribution of Arabian core samples.* Bahrain, Presented at the SPE Middle East Oil & Gas Show and Conference, 11-14 March. SPE-105156-MS.

10) Enbaia, A. & Ramdzani, I., 2014. Pore size and geometry of reservoir rocks used as key factor for drilling and completion fluid design of oil wells. *European Scientific Journal,* Issue May, pp. 102-113.

11) Feng, J., Li, C., Cen, S. & Owen, D., 2014. Statistical reconstruction of two-phase random media. *Computers and Structures,* Issue 137, pp. 78-92.

12) Guiet, J., Reggio, M. & Teyssedou, A., 2011. Implementation and application of the lattice Boltzmann method using MATLAB. *Engineering Applications of Computational Fluid Mechanics,* 5(1), pp. 117-126.

13) Guo, E.-Y.et al., 2014. Accurate modeling and reconstruction of three-dimensional percolating filamentary microstructures from two-dimensional micrographs via dilation-erosion method. *Materials Characterization,* Issue 89, pp. 33-42.

14) Hill, R., 1963. Elastic properties of reinforced solids: some theoretical principles. *Journal of the Mechanics and Physics of Solids,* 11(5), pp. 357-372.

15) ImageJ_webpage, 2015. *ImageJ Documentation Wiki.* [Online] Available at: http://imagejdocu.tudor.lu/doku.php?id=macro:radially_averaged_autocorrelation [Accessed 24 May 2017].

16) Keehm, Y. & Mukerji, T., 2004. *Permeability and Relative Permeability from Digital Rocks: Issues on Grid Resolution and Representative Elementary Volume.* Denver, SEG Int'l Exposition and 74th Annual Meeting.

17) Krüger, T. et al., 2017. *The Lattice Boltzmann Method: Principles and Practice.* s.l.:Springer International Publishing.

18) Manwart, C., Torquato, S. & Hilfer, R., 2000. Stochastic reconstruction of sandstones. *Physical Review E,* 62(1), pp. 893-899.

19) MathWorks_Documentation, 2017. *Morphological Dilation and Erosion.* [Online] Available at: http://se.mathworks.com/help/images/morphological-dilation-and-erosion.html [Accessed 24 May 2017].

20) MathWorks_Documentation, 2017. *Noise Removal.* [Online] Available at: https://se.mathworks.com/help/images/noise-removal.html [Accessed 24 May 2017].

21) Mooney, S. & Korošak, D., 2009. *Using of complex networks to model 2-D and 3-D soil porous architecture.* Vienna, EGU General Assembly.

22) Nichols, G., 2009. *Sedimentology and stratigraphy.* 2nd ed. Chichester: John Wiley & Sons Ltd.

23) Okabe, H., 2004. *Pore-scale modelling of carbonates,* London: Imperial College London.

24) Okabe, H. & Blunt, M. J., 2005. Pore space reconstruction using multiple-point statistics. *Journal of Petroleum Science and Engineering,* Issue 46, pp. 121-137.

25) Øren, P.-E. & Bakke, S., 2003. Reconstruction of Berea sandstone and pore-scale modelling of wettability effects. *Journal of Petroleum Science and Engineering,* Issue 39, pp. 177-199.

26) Rabbani, A. & Jamshidi, S., 2014. Specific surface and porosity relationship for sandstones for prediction of permeability. *International Journal of Rock Mechanics & Mining Sciences,* Issue 71, pp. 25-32.

27) Rabbani, A., Jamshidi, S. & Salehi, S., 2014. An automated simple algorithm for realistic pore network extraction from micro-tomography images. *Journal of Petroleum Science and Engineering,* Issue 123, pp. 164-171.

28) Richa, R., Mukerji, T., Mavko, G. & Keehm, Y., 2006. *Image analysis and pattern recognition for porosity estimation from thin sections.* New Orleans, Proc., SEG Annual Meeting, 1968-1972.

29) Roberts, A. P., 1997. Statistical reconstruction of three-dimensional porous media from two-dimensional images. *Physical Review E,* 56(3), pp. 3203-3212.

30) Sakellariou, A. et al., 2003. *Micro-CT facility for imaging reservoir rocks at pore scales.* Dallas, Proc., SEG Technical Program Expanded Abstracts 2003, 1664-1667. doi: 10.1190/1.1817625.

31) Silin, D., Jin, G. & Patzek, T., 2003. *Robust Determination of the Pore Space Morphology in Sedimentary Rocks.* Denver, Colorado, Presented at the SPE Annual Technical Conference and Exhibition, 5-8 October. SPE-84296-MS.

32) Soulaine, C., 2017. *Prof. Soulaine web-page (Stanford University).* [Online] Available at: https://web.stanford.edu/~csoulain/ [Accessed 24 May 2017].

33) Succi, S., 2001. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond.* Oxford: Clarendon.

34) Sukop, M. et al., 2013. Evaluation of permeability and non-Darcy flow in vuggy macroporous limestone aquifer samples with lattice Boltzmann methods. *Water Resouces Research,* Volume 49, pp. 216-230.

35) Sukop, M. & Thorne, D. J., 2006. *Lattice Boltzmann modeling: an introduction for geoscientists and engineers.* 2nd ed. Berlin: Springer.

36) Talukdar, M., 2002. *Ekofisk chalk: core measurements, stochastic reconstruction, network modeling and simulation,* Trondheim: NTNU.

37) Talukdar, M., Torsaeter, O. & Ioannidis, M., 2002. Stochastic Reconstruction of Particulate Media from Two-Dimensional Images. *Journal of Colloid and Interface Science,* Issue 248, pp. 419-428.

38) The_University_of_Sydney, 2017. *An introduction to percolation theory: A MATLAB simulation.* [Online] Available at: http://www.physics.usyd.edu.au/teach_res/mp/doc/percolation.htm [Accessed 1 June 2017].

39) Torsæter, O. & Abtahi, M., 2003. *Experimental Reservoir Engineering Laboratory Work Book.* 1st ed. Trondheim: Norwegian University of Science and Technology.

40) Wang, J. et al., 2015. Study of the influence of porous structure on the permeability of rock using Lattice Boltzmann method. *Procedia Engineering,* Issue 102, pp. 1835-1841.

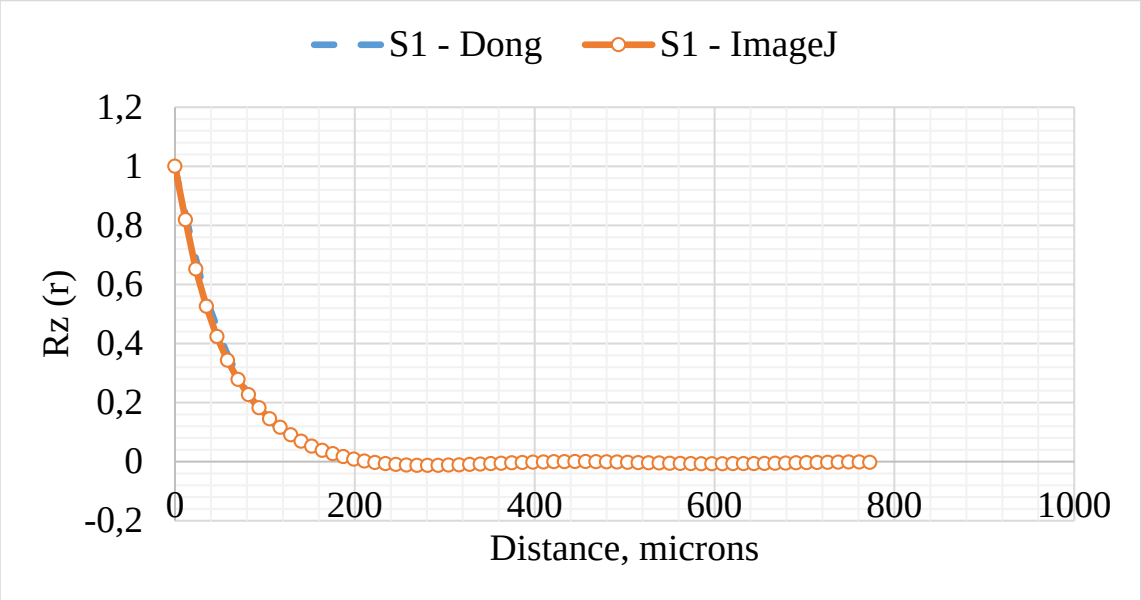
41) Yeong, C. & Torquato, S., 1998. Reconstructingrandom media. II. Three-dimensional media from two-dimensional cuts. *Physical Review E,* 58(1), pp. 224-233.

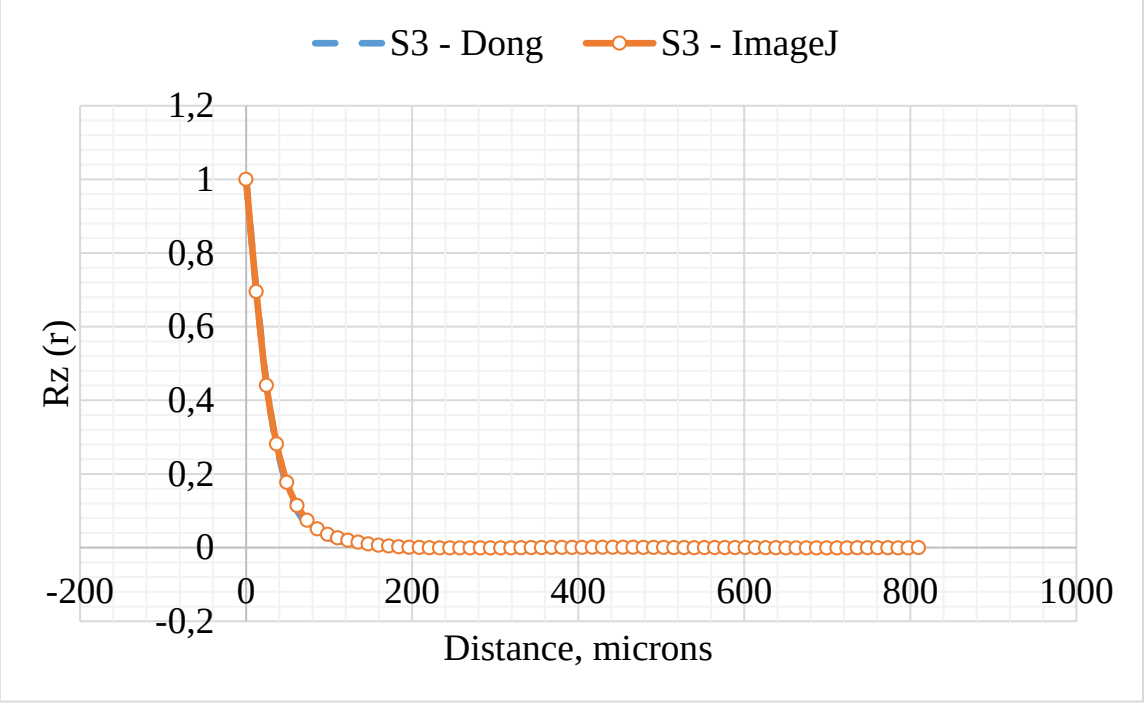
42) Yuan, H., 2007. *Stochastic reconstruction of snow microstructure from X-ray microtomography images,* Fairbanks: University of Alaska Fairbanks.

43) Yuan, H., Lee, J. & Guilkey, J., 2010. Stochastic reconstruction of the microstructure of equilibrium form snow and computation of effective elastic properties. *Journal of Glaciology,* 56(197), pp. 405-414.

Appendix A
Information for remained micro-CT models (Berea #1, S1, S3)

The plots of autocorrelation functions





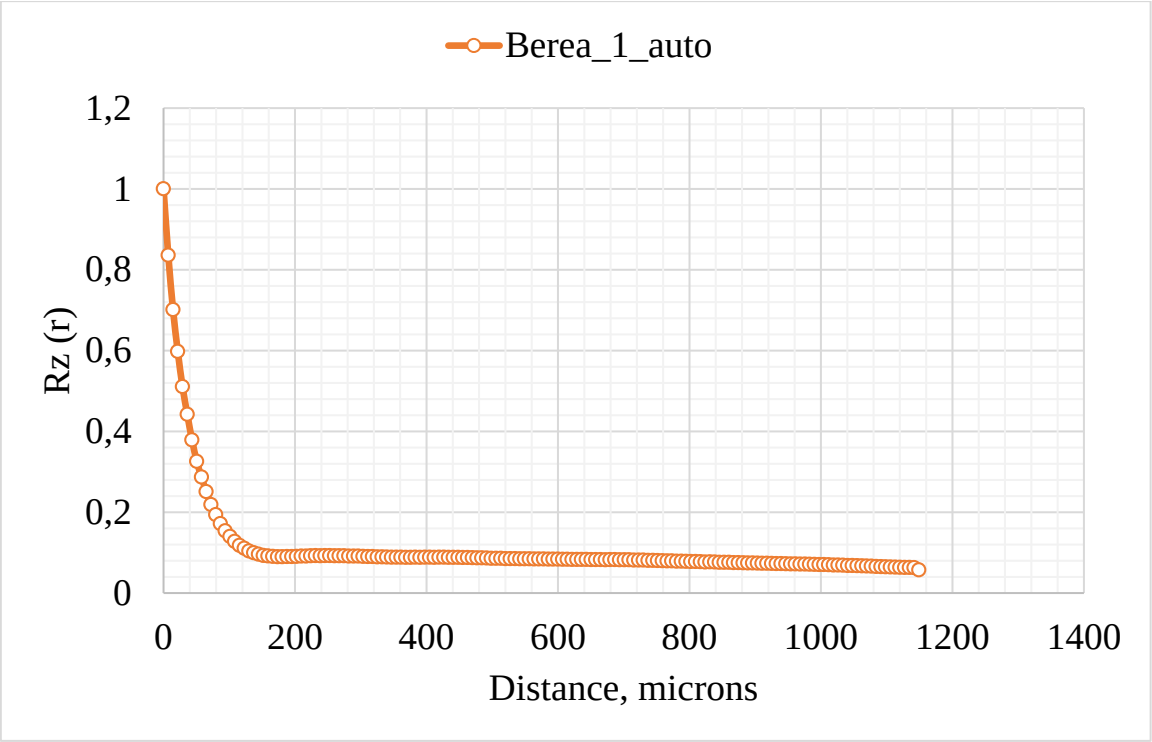
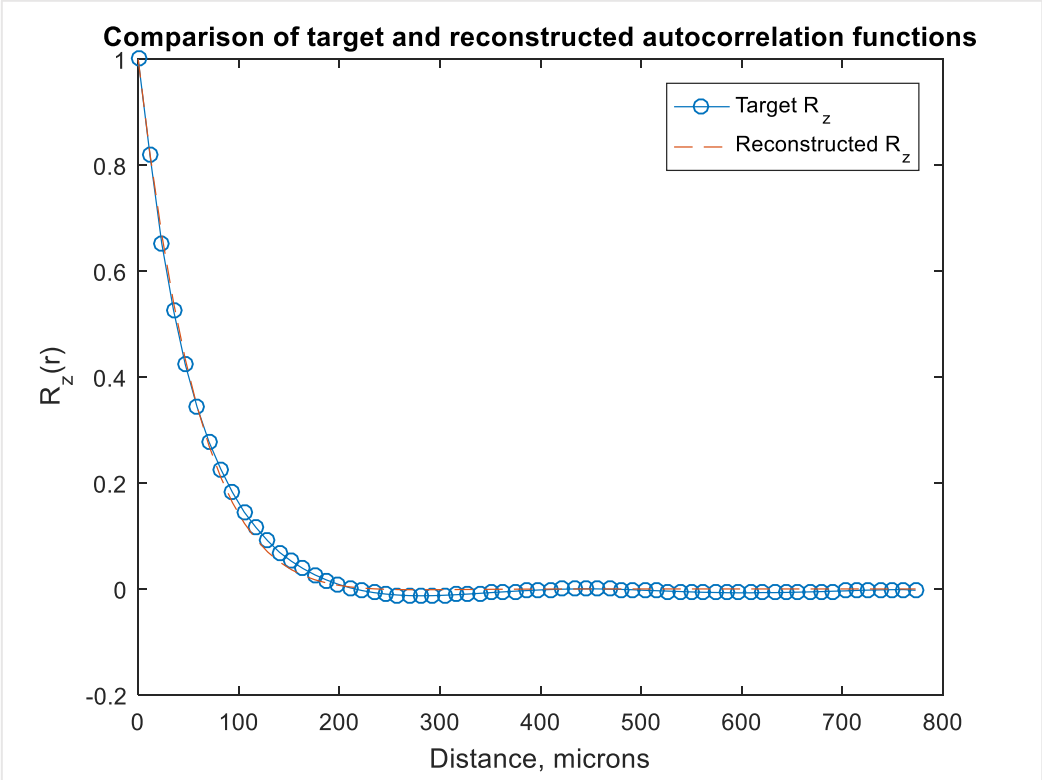


Figure A.1. Autocorrelation functions for three reminded samples.
Those from ImageJ are shown in orange, from Dong's thesis – in blue

Comparison of target and reconstructed autocorrelation functions

S1 sample



S3 sample

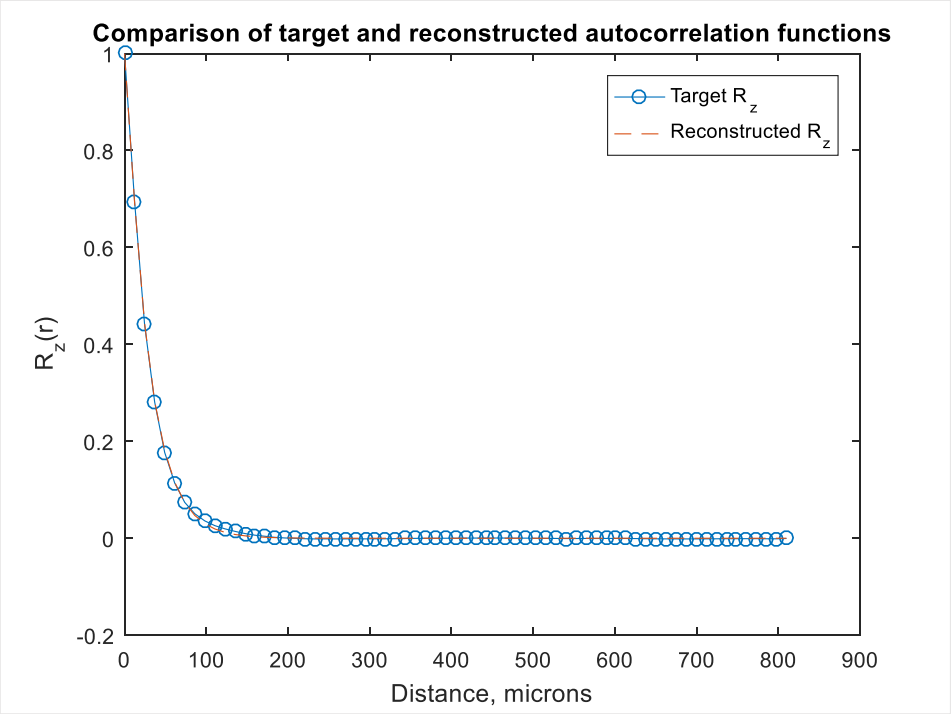
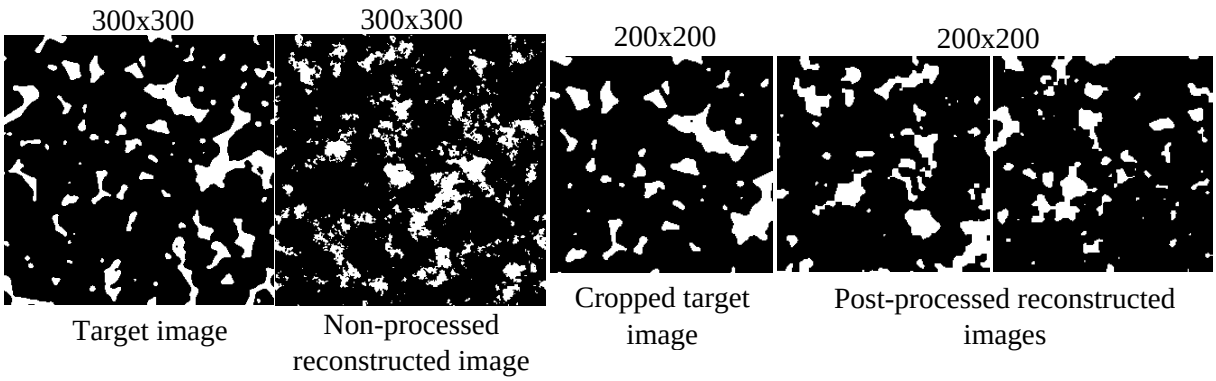


Figure A.2. Comparison of target and reconstructed autocorrelation functions for the remained micro-CT models

Reconstruction procedure

S1 sample



S3 sample

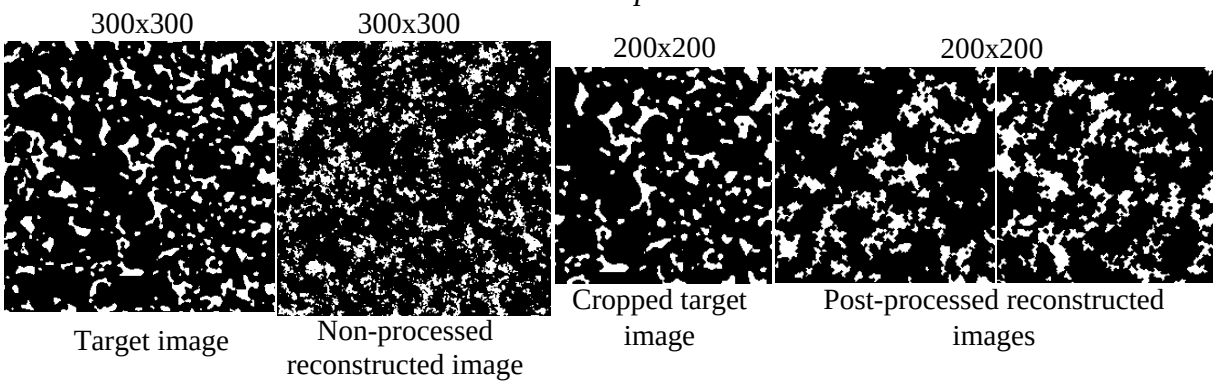
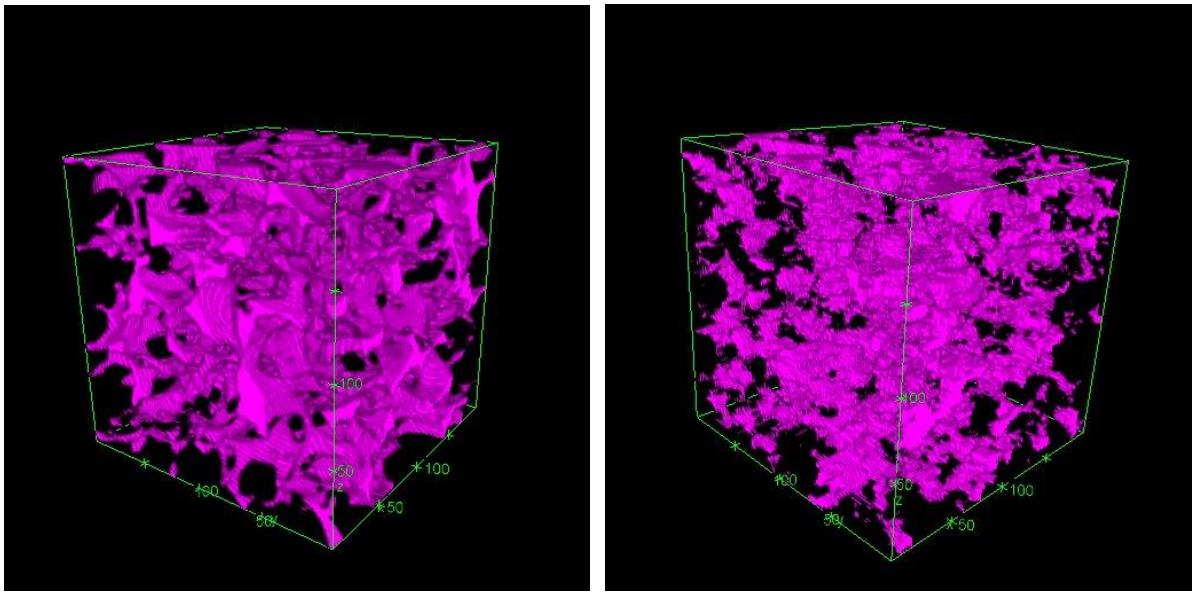


Figure A.3. Comparison of original micro-CT images with randomly chosen 2D cuts of the reconstructed media

Table A.1. Comparison of porosity for the target and reconstructed images

S1		S3	
Image #	**Porosity, φ [%]**	**Image #**	**Porosity, φ [%]**
Target image	14.03	Target image	16.86
Micro-CT image sequence	14.21	Micro-CT image sequence	16.86
Reconstructed sequence	14.94	Reconstructed sequence	16.59

S1 sample



S3 sample

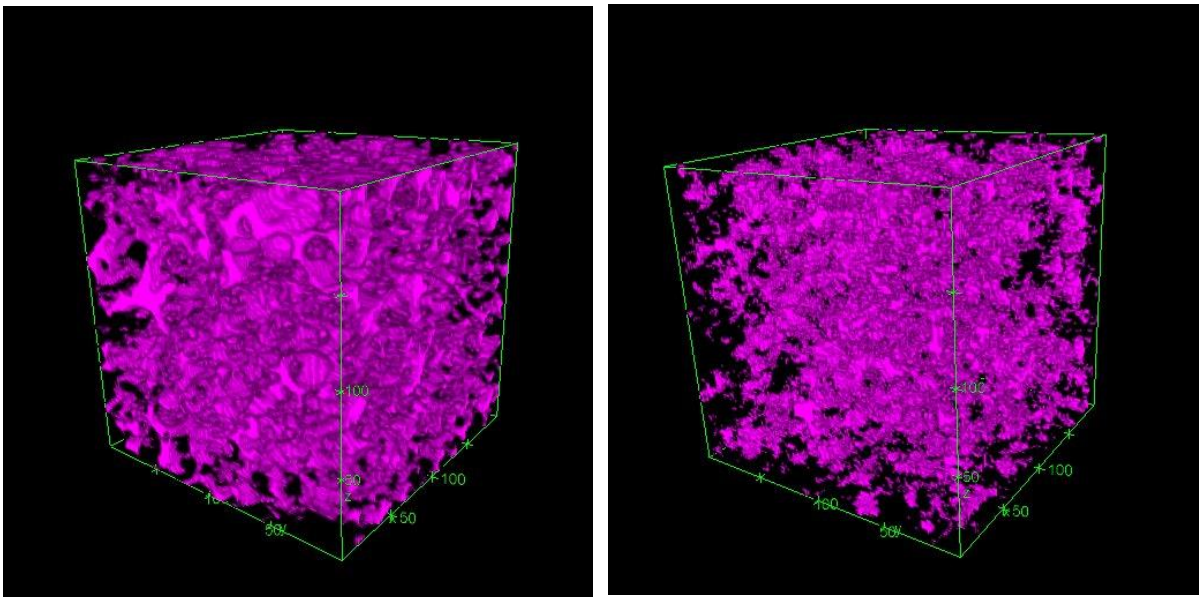
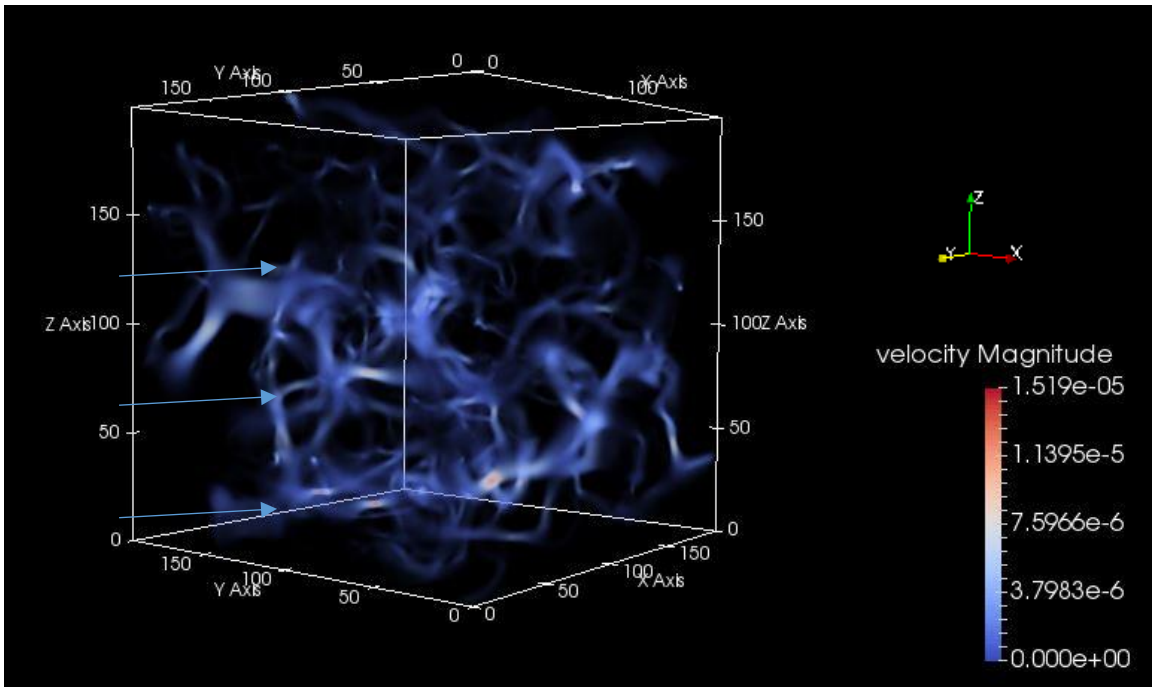
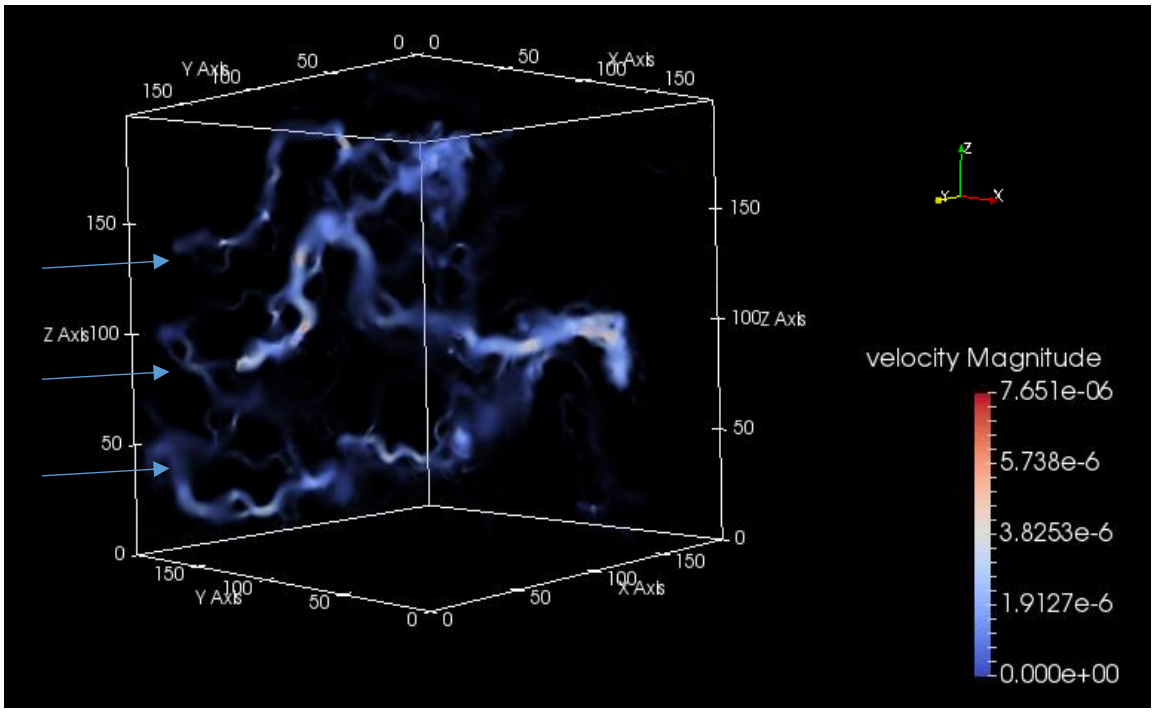


Figure A.4. The pore space of original (left) vs. reconstructed (right) for remaining models; pores are shown in magenta color

3D velocity plots

S1 sample





S3 sample

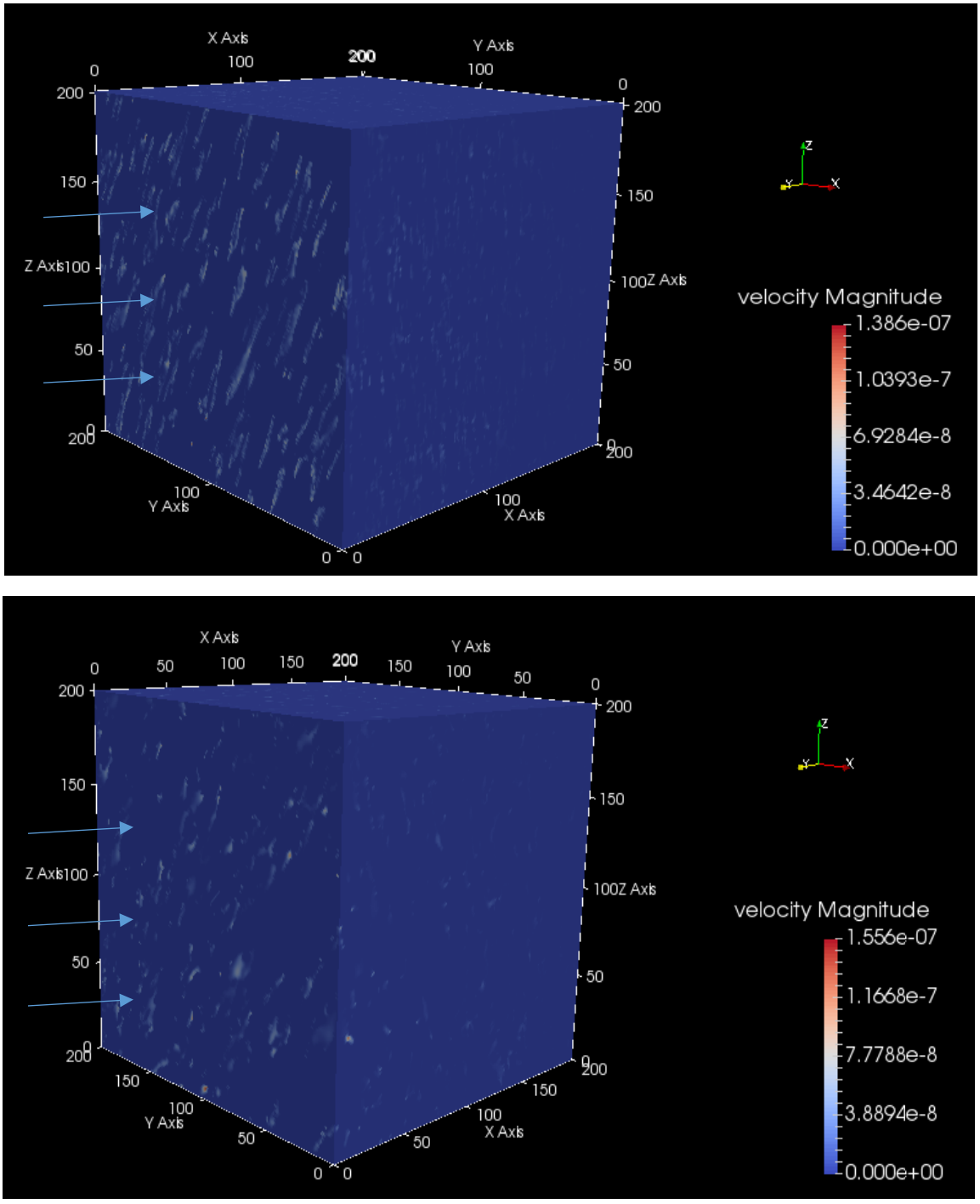


Figure A.5. Velocity distribution for remaining models.
Original are at top and reconstructed – at bottom; direction of the flow is showing by arrows

Note: Due to the chosen delta p, the velocity values for S3 sample were too small to show them in Volume mode. Therefore, the blue background has to be added to visualize velocity distribution in this case.

Appendix B
MATLAB code: binarization and PSD

```
%%%%%%%%%%%%%%%%    MATLAB code - Image Binarization       %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%   Part of the master's thesis (TPG4920)   %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%              Spring 2017                  %%%%%%%%%%%%%%%%
```

Multiple dimension threshold:

```
clear all; clc; close all;
rgbimage = imread('Sherwood_sand.tiff'); % Sherwood
% rgbimage = imread('Bridport_sand.tiff'); % Bridport
figure(1)
imshow(rgbimage);
% Extract the individual red, green, and blue color components
red = rgbimage(:, :, 1);
green = rgbimage(:, :, 2);
blue = rgbimage(:, :, 3);
% Display the histogram for the red component
redhist = imhist(red, 256);
% Display the histogram for the green component
greenhist = imhist(green, 256);
% Display the histogram for the blue component
bluehist = imhist(blue, 256);
% Plot all three histograms on the same plot
figure(2)
lineWidth = 2;
hold off;
plot(redhist, 'r', 'LineWidth', lineWidth);
hold on;
grid on;
plot(greenhist, 'g', 'LineWidth', lineWidth);
plot(bluehist, 'b', 'LineWidth', lineWidth);
title('RGB components histogram');
% Set the x axis range manually to be 0-255.
xlim([0 255]);
ylabel('Number of pixels'); xlabel('Color tones');
legend ('Red component','Green component','Blue component','Location',
'NorthWest');
% Thresholding
threshold_red = 231; % for red component
red_t = red < threshold_red;
threshold_blue = 118; % for blue component
blue_t = blue > threshold_blue;
% Bridport values
% threshold_red = 191; % for red component
% red_t = red < threshold_red;
% threshold_green = 240; % for red component
% green_t = green < threshold_green;
```

```
% threshold_blue = 128; % for blue component
% blue_t = blue > threshold_blue;
% rgb_t = uint8(red_t & green & blue_t);
rgb_t = uint8(red_t & green & blue_t);
maskedrgbimage = uint8(zeros(size(rgb_t))); % Initialize
maskedrgbimage(:,:,1) = rgbimage(:,:,1) .* rgb_t;
maskedrgbimage(:,:,2) = rgbimage(:,:,2) .* rgb_t;
maskedrgbimage(:,:,3) = rgbimage(:,:,3) .* rgb_t;
% imshow(maskedrgbImage);
% title('Masked Original Image');
b = im2bw(maskedrgbimage);
binary = imcomplement(b);
figure (3)
imshow(binary);
% Counting the amount of white and black pixels
pixels = numel(binary);
whitepix = sum(binary(:));
blackpix = pixels - whitepix;
% Porosity will be found as the percentage of black pixels to the total
amount of them
phi = blackpix / pixels;
```

Double threshold and pore size distribution:

```
clear all
clc; close all;
% Read input images
a = imread('Sherwood_sand.jpg');
% a = imread('Bridport_sand.jpg');
g = rgb2gray(a); % transform to gray-colored image
% Gray image
figure (1)
imshow(g);
title ('Gray image');
% Histogram of image data (y-axis: Number of pixels, x-axis: Color tones)
=>
% show distribution of pixels based on color pattern (for gray images the
default is 256)
figure (2)
imhist (g);
title ('Gray image data histogram');
ylabel('Number of pixels'); xlabel('Gray-level value');
% Based on the histogram, set custom defined min-max thresholds for the
gray image
% for Bridport
% min_1 = 127;
% max_1 = 171;
% 127 and 163
% % for Sherwood
min_1 = 127;
max_1 = 201;
% 127 and 188
% Threshold -> everything out of min & max becomes 0
g1 = g;
g1(g1<min_1|g1>max_1) = 0;
figure (3)
imshow(g1); title ('Gray image after thresholding');
b = im2bw(g1); % inversed binary image (pores - white, grains - black)
bw1 = imcomplement(b); % 'anti-inversed' binary image
% Filtering
```

```
B1 = medfilt2(bw1,[7 7]); % median filter to decrease noise
B1 = bwareaopen(B1,10);
figure(4)
imshow(B1);
% Counting the amount of white and black pixels
pixels = numel(B1);
whitepix = sum(B1(:));
blackpix = pixels - whitepix;
% Porosity is found as a percentage of black pixels to their total amount
phi = blackpix / pixels;
% Segmentation
Ed = -bwdist(bw1); % Euclidian distance function; bw1
B = medfilt2(Ed,[7 7]); % median filter for the distance map (avoid
oversegmentation)
Ld1 = watershed(B);
% Pore size distribution
[s1, s2] = size(B);
Pr = zeros(s1,s2);
for i = 1:s1
for j = 1:s2
if B1(i,j)==0 && Ld1(i,j)~= 0 % find pores
Pr(i,j) = 1;
end
end
end
Pr = bwareaopen(Pr,10); % pores greater than 10 pixels are considered
[Pr_L,Pr_n] = bwlabel(Pr); % here Pr_n - number of connected objects, Pr_L
- matrix of labeled elements
z = regionprops(Pr_L,'Area');
Area = [z.Area];
% Pixels to microns
% Resolution = 1310/677; % Bridport [micron/pixel]
Resolution = 2320/968; % Sherwood
k = regionprops(Pr_L,'area');
R = Resolution.*sqrt(Area)./sqrt(pi); % Pore radius (A = pi*R^2);
Diam = 2*R;
Number_of_categories = 20;
% Histogram of relative frequency
Pore_rad_av = mean(R); % Average pore radius [microns]
St_dev = std(R); % Standard deviation of pore radius [micron]
Rel_Frequencies =
hist(R,(1:round(max(R)/Number_of_categories):round(max(R)))./sum(sum(hist(
R,(1:round(max(R)/Number_of_categories):round(max(R))))));
formatSpec = 'Average pore radius is %6.4f microns ';
fprintf(formatSpec,Pore_rad_av);
figure(5)
bar((1:round(max(R)/Number_of_categories):round(max(R))),Rel_Frequencies);
title('Histogram of pore size distribution')
xlabel('Equivalent pore radius (micron)'); ylabel('Relative Frequency');
% Cumulative distribution function
figure(6)
m = cdfplot(Diam);
set(gca,'XScale','log')
title('Cumulative Distribution Function')
xlabel('Equivalent pore diameter, micron'); ylabel('Cumulative
percentage');
q = label2rgb(Pr_L); % show pore & throats
figure(7)
imshow(q);
```

Appendix C
ImageJ macro for the autocorrelation

```
// ImageJ macro to calculate the Radially Averaged Autocorrelation
Function,
// Corrected for finite size effects
// The output is normalized to a value of 1 at zero radius
// and corrected for effects of the finite image size.
// (No need to extend the image size for avoiding edge effects)
//
// Use with a binary input image.
// Needs the "Radial Profile" plugin,
http://rsb.info.nih.gov/ij/plugins/radial-profile.html
//
// Version: 2008-May-14 Michael Schmid
//
requires("1.42p");
doStack=false;
if (nSlices()>1) doStack = getBoolean("Get RDF from full stack?");
setBatchMode(true);
firstSlice=getSliceNumber();
lastSlice=getSliceNumber();
if (doStack) {
  firstSlice=1;
  lastSlice=nSlices();
}
width=getWidth;
height=getHeight;
getPixelSize(unit, pixelWidth, pixelHeight);
run("Select None");
//maxRadius may be modified, should not be larger than 0.3*minOf(width,
height);
maxRadius=0.3*minOf(width, height);
minFFTsize=1.3*maxOf(width, height);
title=getTitle();
size=4;
while(size<minFFTsize) size*=2;
for (slice=firstSlice; slice<=lastSlice; slice++) {
  //make autocorrelation of particle image
  tempTitle="temp-"+random();
  run("Duplicate...", "title="+tempTitle);
  tempID=getImageID();
  getRawStatistics(nPixels, mean);
  run("Canvas Size...", "width="+ size+" height="+ size+" position=Center
zero");
  makeRectangle(floor((size-width)/2), floor((size-height)/2), width,
height);
  run("Make Inverse");
  run("Set...", "value="+mean);
  run("Select None");
  getRawStatistics(nPixels, mean);
```

```
  run("FD Math...", "image1=["+tempTitle+"] operation=Correlate
image2=["+tempTitle+"] result=AutoCorrelation do");
  psID=getImageID();
  run("Subtract...", "value="+(nPixels*mean*mean));
  selectImage(tempID);
  close();

  //make autocorrelation reference to correct finite image size effects
  newImage("frame", "8-bit White", width, height, 1);
  run("Set...", "value=255");
  tempID=getImageID();
  rename(tempTitle);
  run("Canvas Size...", "width="+ size+" height="+ size+" position=Center
zero");
  run("FD Math...", "image1=["+tempTitle+"] operation=Correlate
image2=["+tempTitle+"] result=AutoCorrReference do");
  refID=getImageID();
  imageCalculator("Divide", psID,refID);
  selectImage(refID);
  close();
  selectImage(tempID);
  close();

  //prepare normalized power spectrum for radial averaging
  selectImage(psID);
  circleSize=2*floor(maxRadius)+1;
  norm = getPixel(size/2, size/2);
  run("Divide...", "value="+norm);
  run("Specify...", "width="+circleSize+" height="+circleSize+"
x="+(size/2+0.5)+" y="+(size/2+0.5)+" oval centered");
  run("Radial Profile", "x="+(size/2+0.5)+" y="+(size/2+0.5)+"
radius="+floor(maxRadius)-1);
  Plot.getValues(x, y);
  plotID=getImageID();
  selectImage(psID);
  close();
   if (slice==firstSlice) ySum = newArray(y.length);
  for (i=0; i<y.length; i++)
    ySum[i]+=y[i]/nSlices;
  selectImage(plotID);
  close();
}
if (pixelWidth==pixelHeight) {
  for (i=0; i<x.length; i++)
    x[i] *= pixelWidth;
} else
  unit = "pixels";
if (doStack) title = title + " (stack)";
Plot.create("Autocorrelation of "+title, "Distance ("+unit+")", "Normalized
Autocorrelation", x, ySum);

setBatchMode(false);
```

Appendix D
MATLAB code of 3D reconstruction (on the example of S1 sample)

```
%%%%%%%%%%%%%%%%    MATLAB code - 3D Reconstruction        %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%   Part of the master's thesis (TPG4920)   %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%   Based on code of Yuan Hongyan (in a red frame) %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%                Spring 2017                %%%%%%%%%%%%%%%%
%% Image analysis
clear all; clc; close all;
delete('M:\MATLAB codes for thesis\S1\*.bmp');
delete('M:\MATLAB codes for thesis\S1\S1_filtered\*.bmp'); % delete images
from previous reconstruction
delete('M:\MATLAB codes for thesis\S1\S1_recon_cropped\*.bmp');
delete('M:\MATLAB codes for thesis\S1\S1_recon_cropped_dil_er\*.bmp');
a1 = imread('S1_inversed0110.bmp'); % read input image
% Counting the amount of white and black pixels
pixels = numel(a1);
whitepix = sum(a1(:));
blackpix = pixels - whitepix;
% Porosity will be found as the percentage of black pixels to the total
amount of them
phi_single = whitepix / pixels;
%average porosity
phi = 0.141302778; % found as an average value for all images in micro-CT
model
Resolution = 8.683; % [microns/pixel]
[s1,s2]=size(a1);
%% Two-point- and autocorrelation correlation functions [found using the
plagin in ImageJ]
Dist = ...
    xlsread('Data_Imperial_College_samples.xlsx','S1_auto','A3:A69'); %
read data from the Excel-file
Dist2 = Dist*Resolution;
Rz = ...
    xlsread('Data_Imperial_College_samples.xlsx','S1_auto','B3:B69'); %
read data from the Excel-file
% Plot Rz
figure (2)
plot(Dist2, Rz, '-.r','LineWidth', 2);
title('Target autocorrelation function');
xlabel('Distance, microns'); ylabel('R_z(r)');
legend ('R_z (Image J)','Location', 'NorthEast');
% Find and plot target 2-point correlation function
S2 = phi.^2 + Rz.*(phi-phi.^2);
figure (3)
plot(Dist2, S2, '-rx','MarkerSize',3);
title('Target pore-pore correlation function');
xlabel('Distance, microns'); ylabel('S_2(r)');
legend ('S_2 (ImageJ)','Location', 'NorthEast');
%% Solution of non-linear equation => filter for the GRF algorithm
```

```
cinit1 = [30 0.08 299];
% Correlation function from paper of A. Roberts "Statistical reconstruction
% of 3D porous media from 2D images" (eq. 8), or equation (3.10) in this
work
%  Here c(1) is a correlation length, c(2) is a cutoff scale and c(3) is a
%  domain scale
analyt = @(c,r)((exp(-r./c(1))-(c(2)./c(1)).*exp(-
r./c(2))).*sin(2.*pi.*r./c(3))).//((1-c(2)./c(1)).*(2.*pi.*r./c(3)));
c1 = lsqcurvefit(analyt,cinit1,Dist2,Rz);
f1 = analyt(c1,Dist2);
figure (4)
plot(Dist2,Rz,'-o',Dist2,f1,'--')
title('Comparison of target and reconstructed autocorrelation functions')
xlabel('Distance, microns')
ylabel('R_z(r)')
legend ('Target R_z','Reconstructed R_z','Location', 'NorthEast');
%% Stochastic reconstruction by GRF
V = s1.^3; % cube
ImFolder = 'M:\MATLAB codes for thesis\S1';
alpha = sqrt(2)*erfinv(1-2*phi); % equation (3.15)
c = c1; % coefficents obtained from 'Solution of non-linear equation'
section
% Power spectral density function -> from Roberts, (eq. 9), or equation
(3.12)
% in this work
PSD = @(freq,c) pi^(-2)*(c(1)-c(2))^(-
1)*c(1)^4*c(3)^4/((c(3)^2+c(1)^2*(freq*c(3)-
2*pi)^2)*(c(3)^2+c(1)^2*(freq*c(3)+2*pi)^2))...
    -pi^(-2)*(c(1)-c(2))^(-1)*c(2)^4*c(3)^4/((c(3)^2+c(2)^2*(freq*c(3)-
2*pi)^2)*(c(3)^2+c(2)^2*(freq*c(3)+2*pi)^2));
% Define matrix
Yijk = zeros(s1,s1,s1);
Eff_rad = s1/2/s1;
% case 1
for ii = 2:s1/2+1
    for jj = 2:s1/2+1
        for kk = 2:s1/2+1
            freq = sqrt((ii-1)^2+(jj-1)^2+(kk-1)^2)/s1;
            if freq <= Eff_rad
                Yijk(ii,jj,kk) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
                Yijk(s1+2-ii,s1+2-jj,s1+2-kk) = conj(Yijk(ii,jj,kk));
            end
        end
for kk = s1/2+2:s1
    freq = sqrt((ii-1)^2+(jj-1)^2+(kk-s1-1)^2)/s1;
         if freq <= Eff_rad
                Yijk(ii,jj,kk) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
                Yijk(s1+2-ii,s1+2-jj,s1+2-kk) = conj(Yijk(ii,jj,kk));
         end
end
    end
    for jj = s1/2+2:s1
        for kk = 2:s1/2+1
            freq = sqrt((ii-1)^2+(jj-s1-1)^2+(kk-1)^2)/s1;
         if freq <= Eff_rad
                Yijk(ii,jj,kk) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
                Yijk(s1+2-ii,s1+2-jj,s1+2-kk) = conj(Yijk(ii,jj,kk));
         end
        end
    for kk = s1/2+2:s1
            freq = sqrt((ii-1)^2+(jj-s1-1)^2+(kk-s1-1)^2)/s1;
```

```
        if freq <= Eff_rad
               Yijk(ii,jj,kk) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
               Yijk(s1+2-ii,s1+2-jj,s1+2-kk) = conj(Yijk(ii,jj,kk));
        end
    end
    end
end
% case2: ii = 1
for jj = 2:s1/2+1
    for kk = 2:s1/2+1
           freq = sqrt((jj-1)^2+(kk-1)^2)/s1;
        if freq <= Eff_rad
               Yijk(1,jj,kk) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
               Yijk(1,s1+2-jj,s1+2-kk) = conj(Yijk(1,jj,kk));
        end
    end
    for kk = s1/2+2:s1
           freq = sqrt((jj-1)^2+(kk-s1-1)^2)/s1;
        if freq <= Eff_rad
               Yijk(1,jj,kk) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
               Yijk(1,s1+2-jj,s1+2-kk) = conj(Yijk(1,jj,kk));
        end
   end
end
% case3: jj = 1
for ii = 2:s1/2+1
    for kk = 2:s1/2+1
           freq = sqrt((ii-1)^2+(kk-1)^2)/s1;
        if freq <= Eff_rad
               Yijk(ii,1,kk) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
               Yijk(s1+2-ii,1,s1+2-kk) = conj(Yijk(ii,1,kk));
        end
    end
   for kk = s1/2+2:s1
           freq = sqrt((ii-1)^2+(kk-s1-1)^2)/s1;
        if freq <= Eff_rad
               Yijk(ii,1,kk) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
               Yijk(s1+2-ii,1,s1+2-kk) = conj(Yijk(ii,1,kk));
        end
   end
end
% case4: kk = 1
for ii = 2:s1/2+1
    for jj = 2:s1/2+1
           freq = sqrt((ii-1)^2+(jj-1)^2)/s1;
        if freq <= Eff_rad
               Yijk(ii,jj,1) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
               Yijk(s1+2-ii,s1+2-jj,1) = conj(Yijk(ii,jj,1));
        end
    end
   for jj = s1/2+2:s1
           freq = sqrt((ii-1)^2+(jj-s1-1)^2)/s1;
        if freq <= Eff_rad
               Yijk(ii,jj,1) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
               Yijk(s1+2-ii,s1+2-jj,1) = conj(Yijk(ii,jj,1));
        end
   end
end
% case5: ii = jj = 1
for kk = 2:s1/2
    freq = (kk-1)/s1;
```

```
        if freq <= Eff_rad
                Yijk(1,1,kk) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
                Yijk(1,1,s1+2-kk) = conj(Yijk(1,1,kk));
        end
end
% case6: ii = kk = 1
for jj = 2:s1/2
    freq = (jj-1)/s1;
        if freq <= Eff_rad
                Yijk(1,jj,1) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
                Yijk(1,s1+2-jj,1) = conj(Yijk(1,jj,1));
        end
end
% case7: jj = kk = 1
for ii = 2:s1/2
    freq = (ii-1)/s1;
        if freq <= Eff_rad
                Yijk(ii,1,1) = (randn+1i*randn)*sqrt(0.5*V*PSD(freq,c));
                Yijk(s1+2-ii,1,1) = conj(Yijk(ii,1,1));
        end
end
% case 8: points of Yijk are real numbers
freq = s1/2/s1;
Yijk(1,1,s1/2+1) = randn*sqrt(V*PSD(freq,c));
Yijk(s1/2+1,1,1) = randn*sqrt(V*PSD(freq,c));
Yijk(1,s1/2+1,1) = randn*sqrt(V*PSD(freq,c));
Yijk(1,1,1) = randn*sqrt(V*PSD(0,c));
% solve for Ylmn - FFT
yLMN = ifftn(Yijk); %3D inverse FFT
clear Yijk;
yLMN = sign(yLMN - alpha);
for i = 1:s1
imwrite(yLMN(:,:,i),[ImFolder '/' 'S1_images' int2str(000+i) '.bmp']); %
write cross sections
end
my_folder = 'M:\MATLAB codes for thesis\S1\';
ImFolder = 'M:\MATLAB codes for thesis\S1\S1_filtered'; % mkdir(ImFolder);
f = fullfile(my_folder,'*.bmp'); f = dir(f); % list bmp-files in the
current folder
fileNames = {f.name};
ff = natsortfiles(fileNames); % sort files in normal order (1,2,3,...
instead of 1,11,100,...)
for z = 1:s1
  Im{z} = imread([my_folder,(ff{z})]);
  II = bwareaopen(Im{z},5); % II = medfilt2(I,[5 5]); % opening & median
filter to delete reconstruction's artifacts
  imwrite(II,[ImFolder '/' 'S1_filtered' int2str(000+z) '.bmp']); % write
filtered images
end
%% Find porosity of reconstructed sections, further image cropping and
dilation-erosion
fold = 'M:\MATLAB codes for thesis\S1\S1_filtered\';
fold3 = 'M:\MATLAB codes for thesis\S1\S1_recon_cropped\';
fold4 = 'M:\MATLAB codes for thesis\S1\S1_recon_cropped_dil_er\';
find_fil = fullfile(fold,'*.bmp');
fil_files = dir(find_fil);
fil_names = {fil_files.name};
f_sort = natsortfiles(fil_names);
% Counting the amount of white and black pixels, cropping
for yy = 1:s1
Images{yy} = imread([fold,(f_sort{yy})]);
```

```
%Images{yy} = im2bw(Images{yy});
SmallImages{yy} = imcrop(Images{yy}, [0 0 150 150]); % image cropping
imwrite(SmallImages{yy},[fold2 '/' 'S4_recon_cropped' int2str(000+yy)
'.bmp']);
% Porosity will be found as the percentage of black pixels to the total
amount of them
pixs_1{yy} = numel(Images{1,yy});
white_pixs_1{yy} = sum(Images{1,yy}(:)); black_pixs_1{yy} = pixs_1{yy} -
white_pixs_1{yy};
poro_1{yy} = white_pixs_1{yy} / pixs_1{yy};
se = strel('sphere',1); %sphere 2
dilatedimages{yy} = imdilate(SmallImages{yy},se);
erodedimages{yy}= imerode(dilatedimages{yy},se);
imwrite(erodedimages{yy},[fold3 '/' 'S4_recon_dilated_eroded'
int2str(000+yy) '.bmp']);
pixs_2{yy} = numel(erodedimages{1,yy});
white_pixs_2{yy} = sum(erodedimages{1,yy}(:)); black_pixs_2{yy} =
pixs_2{yy} - white_pixs_2{yy};
poro_2{yy} = white_pixs_2{yy} / pixs_2{yy};
end
```

Appendix E
MATLAB code of creating DAT-file

Function (provided by www.palabos.org):

```
function createDAT(numFiles,path,baseInput,baseOutput)

tic
basename = [path '/' baseInput]; % base name of the bmp files
fid = fopen(baseOutput, 'w');    % open the output file to write in

%%%%%%%%%%%%%%%%%%%%% INLET SLICE %%%%%%%%%%%%%%%%%%%%%%%%

% fname = [basename num2str(1, '%0.4i') '.bmp'];
% fnameu = [basename num2str(2, '%0.4i') '.bmp'];
fname = [basename num2str(1) '.bmp'];
fnameu = [basename num2str(2) '.bmp'];

BB = imread(fname ,'bmp');
CC = imread(fnameu ,'bmp');
nx = size(BB,2)
ny = size(BB,1)
B = zeros(ny,nx);

wholeGeom=zeros(ny,nx,2);

wholeGeom(:,:,1) = BB;
wholeGeom(:,:,2) = CC;

indexMin = find(wholeGeom==0);
indexMax = find(wholeGeom>0);

wholeGeom(indexMin) = 255;
wholeGeom(indexMax) = 0;

%i
rA = circshift(wholeGeom,[0,1,0]);
lA = circshift(wholeGeom,[0,-1,0]);
fA = circshift(wholeGeom,[1,0,0]);
bA = circshift(wholeGeom,[-1,0,0]);
rfA = circshift(wholeGeom,[1,1,0]);
rbA = circshift(wholeGeom,[-1,1,0]);
lfA = circshift(wholeGeom,[1,-1,0]);
lbA = circshift(wholeGeom,[-1,-1,0]);

%i+1
uA=circshift(wholeGeom,[0,0,1]);
urA = circshift(wholeGeom,[0,1,1]);
ulA = circshift(wholeGeom,[0,-1,1]);
```

```
ufA = circshift(wholeGeom,[1,0,1]);
ubA = circshift(wholeGeom,[-1,0,1]);
urfA = circshift(wholeGeom,[1,1,1]);
urbA = circshift(wholeGeom,[-1,1,1]);
ulfA = circshift(wholeGeom,[1,-1,1]);
ulbA = circshift(wholeGeom,[-1,-1,1]);

for i = 1:nx
        for j = 1:ny
            if (wholeGeom(j,i,1) == 255 && rA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && lA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && fA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && bA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && rfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && rbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && lfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && lbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && uA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && urA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && ulA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && ufA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && ubA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && urfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && urbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && ulfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 255 && ulbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,1) == 0)
                B(j,i) = 0;
            else
                B(j,i) = 2;
            end
        end
end

image(30*B)
axis equal
drawnow

% printing first slice
fprintf(fid, '%i\n', B);

%%%%%%%%%%%%%%%%%%%%%% INTERNAL SLICES %%%%%%%%%%%%%%%%%%%%%%%%
```

```
for ii = 2:numFiles-1
ii
% fname = [basename num2str(ii, '%0.4i') '.bmp'];
% fnamed =[basename num2str(ii-1, '%0.4i') '.bmp'];
% fnameu = [basename num2str(ii+1, '%0.4i') '.bmp'];
fname = [basename num2str(ii) '.bmp'];
fnamed =[basename num2str(ii-1) '.bmp'];
fnameu = [basename num2str(ii+1) '.bmp'];
AA = imread(fnamed ,'bmp');
BB = imread(fname ,'bmp');
CC = imread(fnameu ,'bmp');

wholeGeom = zeros(ny,nx,3);
wholeGeom(:,:,1) = AA;
wholeGeom(:,:,2) = BB;
wholeGeom(:,:,3) = CC;

indexMin = find(wholeGeom==0);
indexMax = find(wholeGeom>0);

wholeGeom(indexMin) = 255;
wholeGeom(indexMax) = 0;

%i
rA = circshift(wholeGeom,[0,1,0]);
lA = circshift(wholeGeom,[0,-1,0]);
fA = circshift(wholeGeom,[1,0,0]);
bA = circshift(wholeGeom,[-1,0,0]);
rfA = circshift(wholeGeom,[1,1,0]);
rbA = circshift(wholeGeom,[-1,1,0]);
lfA = circshift(wholeGeom,[1,-1,0]);
lbA = circshift(wholeGeom,[-1,-1,0]);

%i-1
dA=circshift(wholeGeom,[0,0,-1]);
drA = circshift(wholeGeom,[0,1,-1]);
dlA = circshift(wholeGeom,[0,-1,-1]);
dfA = circshift(wholeGeom,[1,0,-1]);
dbA = circshift(wholeGeom,[-1,0,-1]);
drfA = circshift(wholeGeom,[1,1,-1]);
drbA = circshift(wholeGeom,[-1,1,-1]);
dlfA = circshift(wholeGeom,[1,-1,-1]);
dlbA = circshift(wholeGeom,[-1,-1,-1]);

%i+1
uA=circshift(wholeGeom,[0,0,1]);
urA = circshift(wholeGeom,[0,1,1]);
ulA = circshift(wholeGeom,[0,-1,1]);
ufA = circshift(wholeGeom,[1,0,1]);
ubA = circshift(wholeGeom,[-1,0,1]);

urfA = circshift(wholeGeom,[1,1,1]);
urbA = circshift(wholeGeom,[-1,1,1]);
ulfA = circshift(wholeGeom,[1,-1,1]);
ulbA = circshift(wholeGeom,[-1,-1,1]);

for i = 1:nx
        for j = 1:ny
            if (wholeGeom(j,i,2) == 255 && rA(j,i)==0)
                B(j,i) = 1;
```

```
            elseif (wholeGeom(j,i,2) == 255 && lA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && fA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && bA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && rfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && rbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && lfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && lbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && drA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dlA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && drfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && drbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dlfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dlbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && uA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && urA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && ulA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && ufA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && ubA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && urfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && urbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && ulfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && ulbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 0)
                B(j,i) = 0;
            else
                B(j,i) = 2;
            end
        end
end

image(30*B)
axis equal
drawnow
```

```
fprintf(fid, '%i\n', B);
end

%%%%%%%%%%%%%%%%%%%% OUTLET SLICES %%%%%%%%%%%%%%%%%%%%%%%%

% fname = [basename num2str(numFiles, '%0.4i') '.bmp'];
% fnamed =[basename num2str(numFiles-1, '%0.4i') '.bmp'];
fname = [basename num2str(numFiles) '.bmp'];
fnamed =[basename num2str(numFiles-1) '.bmp'];

AA = imread(fnamed ,'bmp');
BB = imread(fname ,'bmp');

wholeGeom = zeros(ny,nx,2);
wholeGeom(:,:,1) = AA;
wholeGeom(:,:,2) = BB;

indexMin = find(wholeGeom==0);
indexMax = find(wholeGeom>0);

wholeGeom(indexMin)=255;
wholeGeom(indexMax)=0;

%i
rA = circshift(wholeGeom,[0,1,0]);
lA = circshift(wholeGeom,[0,-1,0]);
fA = circshift(wholeGeom,[1,0,0]);
bA = circshift(wholeGeom,[-1,0,0]);
rfA = circshift(wholeGeom,[1,1,0]);
rbA = circshift(wholeGeom,[-1,1,0]);
lfA = circshift(wholeGeom,[1,-1,0]);
lbA = circshift(wholeGeom,[-1,-1,0]);

%i-1
dA = circshift(wholeGeom,[0,0,-1]);
drA = circshift(wholeGeom,[0,1,-1]);
dlA = circshift(wholeGeom,[0,-1,-1]);
dfA = circshift(wholeGeom,[1,0,-1]);
dbA = circshift(wholeGeom,[-1,0,-1]);
drfA = circshift(wholeGeom,[1,1,-1]);
drbA = circshift(wholeGeom,[-1,1,-1]);
dlfA = circshift(wholeGeom,[1,-1,-1]);
dlbA = circshift(wholeGeom,[-1,-1,-1]);

for i = 1:nx
        for j = 1:ny
            if (wholeGeom(j,i,2) == 255 && rA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && lA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && fA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && bA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && rfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && rbA(j,i)==0)
                B(j,i) = 1;
```

```
            elseif (wholeGeom(j,i,2) == 255 && lfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && lbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && drA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dlA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && drfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && drbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dlfA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 255 && dlbA(j,i)==0)
                B(j,i) = 1;
            elseif (wholeGeom(j,i,2) == 0)
                B(j,i) = 0;
            else
                B(j,i) = 2;
            end
        end
end

image(30*B)
axis equal
drawnow

% printing last slice
fprintf(fid, '%i\n', B);
fclose(fid);
toc
```

Example of the script (for S1 sample):

```
numFiles = 200;
path = 'S1_200';
baseInput = 'S1_200';
baseOutput = 'S1_200.dat';
createDAT_init(numFiles,path,baseInput,baseOutput);
```

Appendix F
The permeability.cpp code (PALABOS)

```
/* This file is part of the Palabos library.
 *
 * Copyright (C) 2011-2015 FlowKit Sarl
 * Route d'Oron 2
 * 1010 Lausanne, Switzerland
 * E-mail contact: contact@flowkit.com
 *
 * The most recent release of Palabos can be downloaded at
 * <http://www.palabos.org/>
 *
 * The library Palabos is free software: you can redistribute it and/or
 * modify it under the terms of the GNU Affero General Public License as
 * published by the Free Software Foundation, either version 3 of the
 * License, or (at your option) any later version.
 *
 * The library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU Affero General Public License for more details.
 *
 * You should have received a copy of the GNU Affero General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
/* Main author: Wim Degruyter */
```

```
#include "palabos3D.h"
#include "palabos3D.hh"
#include <vector>
#include <cmath>

using namespace plb;
using namespace std;

typedef double T;
#define DESCRIPTOR descriptors::D3Q19Descriptor

// This function object returns a zero velocity, and a pressure
which decreases
//   linearly in x-direction. It is used to initialize the
particle populations.
class PressureGradient {
public:
    PressureGradient(T deltaP_, plint nx_) : deltaP(deltaP_),
nx(nx_)
    { }
```

```
22      void operator() (plint iX, plint iY, plint iZ, T& density,
23  Array<T,3>& velocity) const
24      {
25          velocity.resetToZero();
26          density = 1. - deltaP*DESCRIPTOR<T>::invCs2 / (T)(nx-1) *
27  (T)iX;
28
29      }
30  private:
31      T deltaP;
32      plint nx;
33  };
34
35  void porousMediaSetup( MultiBlockLattice3D<T,DESCRIPTOR>& lattice,
36      OnLatticeBoundaryCondition3D<T,DESCRIPTOR>* boundaryCondition,
37      MultiScalarField3D<int>& geometry, T deltaP)
38  {
39      const plint nx = lattice.getNx();
40      const plint ny = lattice.getNy();
41      const plint nz = lattice.getNz();
42
43      pcout << "Definition of inlet/outlet." << endl;
44      Box3D inlet (0,0,      1,ny-2, 1,nz-2);
45      boundaryCondition->addPressureBoundary0N(inlet, lattice);
46      setBoundaryDensity(lattice, inlet, 1.);
47
48      Box3D outlet(nx-1,nx-1, 1,ny-2, 1,nz-2);
49      boundaryCondition->addPressureBoundary0P(outlet, lattice);
50      setBoundaryDensity(lattice, outlet, 1. -
51  deltaP*DESCRIPTOR<T>::invCs2);
52
53      pcout << "Definition of the geometry." << endl;
54      // Where "geometry" evaluates to 1, use bounce-back.
55      defineDynamics(lattice, geometry, new
56  BounceBack<T,DESCRIPTOR>(), 1);
57      // Where "geometry" evaluates to 2, use no-dynamics (which
58  does nothing).
59      defineDynamics(lattice, geometry, new
60  NoDynamics<T,DESCRIPTOR>(), 2);
61
62      pcout << "Initilization of rho and u." << endl;
63      initializeAtEquilibrium( lattice, lattice.getBoundingBox(),
64  PressureGradient(deltaP, nx) );
65
66      lattice.initialize();
67      delete boundaryCondition;
68  }
69
70  void writeGifs(MultiBlockLattice3D<T,DESCRIPTOR>& lattice, plint
71  iter)
72  {
73      const plint nx = lattice.getNx();
74      const plint ny = lattice.getNy();
75      const plint nz = lattice.getNz();
76
77      const plint imSize = 600;
78      ImageWriter<T> imageWriter("leeloo");
79
80      // Write velocity-norm at x=0.
```

```
    imageWriter.writeScaledGif( createFileName("ux_inlet", iter,
6),
        *computeVelocityNorm(lattice, Box3D(0,0, 0,ny-1, 0,nz-1)),
        imSize, imSize );

    // Write velocity-norm at x=nx/2.
    imageWriter.writeScaledGif( createFileName("ux_half", iter,
6),
        *computeVelocityNorm(lattice, Box3D(nx/2,nx/2, 0,ny-1,
0,nz-1)),
        imSize, imSize );

}


void writeVTK(MultiBlockLattice3D<T,DESCRIPTOR>& lattice, plint
iter)
{
        VtkImageOutput3D<T> vtkOut(createFileName("vtk", iter, 6),
1.);
        vtkOut.writeData<float>(*computeVelocityNorm(lattice),
"velocityNorm", 1.);
        vtkOut.writeData<3,float>(*computeVelocity(lattice),
"velocity", 1.);
}

T computePermeability (
    MultiBlockLattice3D<T,DESCRIPTOR>& lattice, T nu, T deltaP,
Box3D domain )
{
    pcout << "Computing the permeability." << endl;

    // Compute only the x-direction of the velocity (direction of
the flow).
    plint xComponent = 0;
    plint nx = lattice.getNx();

    T meanU = computeAverage (
        *computeVelocityComponent (lattice, domain, xComponent )
);


    pcout << "Average velocity     = " << meanU             <<
endl;
    pcout << "Lattice viscosity nu = " << nu                <<
endl;
    pcout << "Grad P               = " << deltaP/(T)(nx-1)
<< endl;
    pcout << "Permeability         = " << nu*meanU /
(deltaP/(T)(nx-1)) << endl;

    return meanU;
}

int main(int argc, char **argv)
{
    plbInit(&argc, &argv);

    if (argc!=7)
```

```
140     {
141         pcout << "Error missing some input parameter\n";
142         pcout << "The structure is :\n";
143         pcout << "1. Input file name.\n";
144         pcout << "2. Output directory name.\n";
145         pcout << "3. number of cells in X direction.\n";
146         pcout << "4. number of cells in Y direction.\n";
147         pcout << "5. number of cells in Z direction.\n";
148         pcout << "6. Delta P .\n";
149         exit (EXIT_FAILURE);
150     }
151     std::string fNameIn  = argv[1];
152     std::string fNameOut = argv[2];
153 
154     const plint nx = atoi(argv[3]);
155     const plint ny = atoi(argv[4]);
156     const plint nz = atoi(argv[5]);
157     const T   deltaP = atof(argv[6]);
158 
159     global::directories().setOutputDir(fNameOut+"/");
160 
161     const T omega = 1.0;
162     const T nu    = ((T)1/omega-0.5)/DESCRIPTOR<T>::invCs2;
163 
164     pcout << "Creation of the lattice." << endl;
165     MultiBlockLattice3D<T,DESCRIPTOR> lattice(nx,ny,nz, new
166 BGKdynamics<T,DESCRIPTOR>(omega));
167     // Switch off periodicity.
168     lattice.periodicity().toggleAll(false);
169 
170     pcout << "Reading the geometry file." << endl;
171     MultiScalarField3D<int> geometry(nx,ny,nz);
172     plb_ifstream geometryFile(fNameIn.c_str());
173     if (!geometryFile.is_open()) {
174             pcout << "Error: could not open geometry file " <<
175 fNameIn << endl;
176             return -1;
177     }
178     geometryFile >> geometry;
179 
180     pcout << "nu = " << nu << endl;
181     pcout << "deltaP = " << deltaP << endl;
182     pcout << "omega = " << omega << endl;
183     pcout << "nx = " << lattice.getNx() << endl;
184     pcout << "ny = " << lattice.getNy() << endl;
185     pcout << "nz = " << lattice.getNz() << endl;
186 
187     porousMediaSetup(lattice,
188 createLocalBoundaryCondition3D<T,DESCRIPTOR>(), geometry, deltaP);
189 
190     // The value-tracer is used to stop the simulation once it has
191 converged.
192     // 1st parameter:velocity
193     // 2nd parameter:size
194     // 3rd parameters:threshold
195     // 1st and second parameters ae used for the length of the
196 time average (size/velocity)
197     util::ValueTracer<T> converge(1.0,1000.0,1.0e-4);
198 
```

```
199     pcout << "Simulation begins" << endl;
200     plint iT=0;
201 
202     const plint maxT = 30000;
203     for (;iT<maxT; ++iT)
204     {
205         if (iT % 20 == 0) {
206             pcout << "Iteration " << iT << endl;
207         }
208         if (iT % 500 == 0 && iT>0) {
209             writeGifs(lattice,iT);
210         }
211 
212         lattice.collideAndStream();
213         converge.takeValue(getStoredAverageEnergy(lattice),true);
214 
215         if (converge.hasConverged())
216         {
217             break;
218         }
219     }
220 
221     pcout << "End of simulation at iteration " << iT << endl;
222 
223     pcout << "Permeability:" << endl << endl;
224     computePermeability(lattice, nu, deltaP,
225 lattice.getBoundingBox());
226     pcout << endl;
227 
228     pcout << "Writing VTK file ..." << endl << endl;
229     writeVTK(lattice,iT);
230     pcout << "Finished!" << endl << endl;
231 }
```

Appendix G
OpenPNM examples

Import network models generated by maximal ball algorithm (Imperial College London):

```
import OpenPNM as op
import OpenPNM.Utilities.IO as io
pa = r'C:\Users\Radmila\Desktop\files'
pref = 'S1'
statoil = io.Statoil;
pn = op.Network.MatFile()
statoil.load(network=pn, path=pa, prefix=pref)
op.export_data(pn)
```

The following code shows how to perform the mercury intrusion porosimetry simulation using a simple regular lattice network:

```
import OpenPNM as op
pn = op.Network.Cubic(shape=[10, 10, 10], spacing=0.0001)
geo = op.Geometry.Stick_and_Ball(network=pn, pores=pn.Ps,throats=pn.Ts)
Hg = op.Phases.Mercury(network=pn)
Air = op.Phases.Air(network=pn)
phys = op.Physics.Standard(network=pn, phase=Hg, pores=pn.Ps,throats=pn.Ts)
MIP = op.Algorithms.Drainage(network=pn)
MIP.setup(invading_phase=Hg, defending_phase=Air)
MIP.set_inlets(pores=pn.pores(['top', 'bottom']))
MIP.run()
MIP.plot_drainage_curve()
op.export_data(network=pn, filename='test', fileformat='VTK')
```