Utrecht University

Pore Network Model Extraction and the Influence on Relative Permeability: A Comparison with Percolation Theory

Abstract

This study introduces a new pore network structure model, based on the fundamentals of medial axis thinning and the maximal ball algorithm. Combining both methods provides great potential in preserving both topology and geometry, two key aspects to accurately mimic the respective pore space. To increase the accuracy of the pore network model even further, several characteristic improvements have been applied regarding the merging algorithm and the partitioning of the geometrical features. All alterations are physically viable and enhance the amount of detail, increasing the realism and robustness compared to other models. Ultimately, the model combines proven concepts and correct alteration to improve estimations related to fluid flow properties in porous media.

The model is tested on twelve different samples, incorporating two carbonates and ten sandstones. Results regarding the pore network properties and absolute permeability are directly compared to the medial axis algorithm (Lindquist et al., 1996) and maximal ball algorithm (Dong and Blunt, 2009). The flow properties, including the relative permeability, are extracted using PoreFlow (Raoof et al., 2013). Percolation theory is also introduced to derive the relative permeability, combining critical path analysis and universal scaling. The results generated by this analytical framework are compared to the results generated by PoreFlow, giving new insights into the predictive capabilities of percolation theory. Overall, percolation theory shows great potential, however, accurately estimating the fractal dimensionality is crucial to guarantee accurate predictions.

Floris Denekamp

2nd of July 2019, Utrecht

Table of Contents

1 Introduction

Pore network modelling is applied to simulate fluid flow in porous media at a micro scale, extracting the governing parameters viable at a continuum macro scale. Modelling multiphase flow requires constitutive relations between: i) the saturation and the capillary pressure, ii) the relative permeability and either the saturation or capillary pressure and iii) the solute dispersivity and saturation (Raoof and Hassanizadeh, 2012). Field experiments obtaining these relations are often expensive and time consuming, while analytical solutions are restricted to boundary conditions and oversimplification (Xiong et al., 2016), however, numerical models may provide an opportunity to obtain transport properties.

Two types of numerical pore models have been applied widely; direct numerical solutions (DNSs) and pore network models (PNMs). DBSs promise realistic and reliable results, as they simulate flow in an exact replica of the porous medium. However, these models become inefficient when the amount of interconnected pore throats or pore volume starts to increase (Tartakovsky et al., 2007). DNSs are also computationally demanding, as the entire pore space is discretized. Alternatively, the smallest discretization unit in a PNM is generally either a pore body or a pore throat, drastically decreasing the computational demand but keeping the complexity of the system intact in terms of pore connectivity. PNMs have been used extensively to simulate single- and multiphase fluid flow in porous media and are proven to provide valuable insight into the governing properties (Xiong et al., 2016). Nevertheless, PNMs are applied to idealizations of the pore space using simple geometries, which may lead to less credible reactive transport properties (Raoof et al., 2013). The predictive capability of PNMs depends on i) the process being simulated, ii) the assumptions made in approximating the fluid mechanics and, iii) how well the network structure depicts the actual geometry and topology of the real porous medium (Bhattad et al, 2011). This research will mainly focus on the third dependence, combining current PNM methods to exploit the advantages of each technique and generate a superior pore network model. The flow properties, including the relative permeability in relation to the saturation will be obtained by incorporating PoreFlow (Raoof et al., 2013), a computational tool capable of simulating fluid flow under variable saturated conditions.

Another approach to derive macroscopic properties is by applying the percolation theory to micro-scale connectivity statistics (Ghanbarian et al., 2015). Percolation theory provides an analytical framework to model interconnectivity of disordered networks and porous materials, laying the foundation for universal scaling and critical path analysis (CPA). A key aspect of percolation theory is the critical water content, which is the minimum saturation to create a connected path of saturated pores spanning between the inflow and outflow boundary of a porous medium. Ghanbarian and Hunt (2012) demonstrated that universal scaling can describe the conductivity at saturation higher but close to the critical water content, while CPA generates results at higher saturations. Hence, universal scaling needs to be supplemented by CPA to compute permeability results over the whole saturation spectrum. This research will incorporate both techniques to obtain the relative permeability which will then be compared to the results generated by PoreFlow. This will give a new insight into the predictive capabilities of percolation theory, giving direct comparisons between pore network modelling and percolation theory.

2 Modelling Porous Media

As this research focusses on generating a viable pore network model (PNM), the details about direct numerical solutions (DNSs) are only discussed briefly.

2.1 Direct Numerical Solutions

One of the most popular DNSs is the lattice-Botlzmann (LB) model, which solves a discretized Boltzmann equation of fluid particle distributions that move and interact on a regular lattice with very few degrees of freedom (Ramstad et al., 2010). This limitation in freedom originates from the fact that the LB model is evolved from a cellular automaton, in which particles can only move to the next neighbouring lattice point in one time step. This implementation generates both the strengths and weaknesses of the LB model, as the velocity vectors need to be computed at every lattice point for each particle distribution for every single time step. These extensive computations, in combination with the discretization of the entire pore space results in very accurate fluid flow predictions, making it the current standard method for single phase flow and transport (Blunt et al., 2013). However, multiphase flow adds more complexity, as the interaction between different fluids has major influence on the capillary flow. Consequently, the particle interactions need to be incorporated, further increasing the computational demand of the LB model, which is its biggest limitation. Another important drawback is the numerical instability when handling multiphase flow with large viscosity and density differences (Meakin and Tartakovsky, 2009), making it for example unsuitable for carbon capture storage simulations.

2.2 Pore Network Modelling

The computational demand of DNSs restricts the applicable porous media to small, homogeneous rock samples with narrow pore size distributions (Bultreys et al, 2016). In the future, DNSs are expected to become the benchmark, but up to date, PNMs have been the most successful methods of simulating multiphase fluid flow (Blunt, 2001; Blunt et al., 2013). PNMs find their origin in the discretization of pore bodies and pore throats, simulating fluid flow within and in between the two. However, a pore space is continuous, and therefor discretizing the system into pore bodies and pore throats sometimes results in ambiguous definitions (Hunt et al., 2014). The central idea of PNMs is to simplify the pore space by capturing all the relevant topological and geometrical traits of the system. Successfully capturing these two aspects plays a crucial role in the reliability of the generated PNM.

2.3 PNM Properties

Topology and geometry of the pore space play an important role in the hydraulic properties of a porous medium (Vogel and Roth, 2001). The former defining the location and connectivity of the pore space while the latter characterizes the size and shape. Both parameters influence different hydrogeological properties, although ultimately, they act together. The topology mainly alters the (relative) permeability, which is hugely affected by the connectivity of the system. Nevertheless, can the geometry also be linked to this process, as the shape of the throats influences the wettability and therefor also the connectivity. The credibility of the pore network predictions is thus highly dependent on both the topology and geometry and it is crucial to accurately mimic these characteristics in order to effectively simulate multiphase flow and transport phenomena (Xiong et al, 2016).

2.3.1 Topology

The main topological characteristics of a pore network are the spatial location of the pore bodies and the connectivity of the pore elements (Joekar-Niasar and Hassanizadeh, 2012). There are four widely accepted structures to simulate a three-dimensional pore network in a gird: i) structured regular, ii)

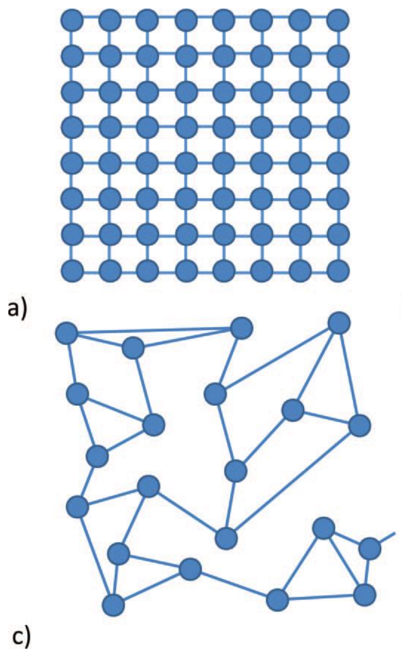


Figure 1. *The four different classifications of pore networks: (a) structured regular, (b) structured irregular, (c) unstructured regular and (d) unstructured irregular (Joekar-Niasar and Hassanizadeh, 2012).*

structured irregular, iii) unstructured regular and iv) unstructured irregular (Figure 1). Whether a pore network is structured or unstructured has to do with the occupation of the pore bodies, which are all equally divided in space for a structured grid. The regular or irregular classification is based on the coordination number of each pore body. In a regular grid all pore bodies have an equal number of throats connected to another pore body, in irregular grids this is not the case. Regular grids are most often used when no direct measurements of the porous material are available and only statistical data is captured (Xiong et al. 2016). However, when direct measurements of the pore space are available an irregular unstructured grid is favourable, generating more feasible results, especially regarding real porous media in which a wide range of coordination numbers is present (Raoof and Hassanizadeh, 2010).

The mean coordination number and the supplementing distribution also provide a valuable insight into the topological randomness of real porous media. The mean coordination number and its range can vary widely between different rocks, increasing with an increase in porosity (Lindquist et al., 2000). Several studies conducted with X-ray computed microtomography and skeletonization algorithms revealed a mean coordination number of 4 for most sandstones (Thovert et al., 1993; Lindquist et al., 1996; Bakke and Oren, 1997; Oren et al., 1997; Oren and Bakke, 2002). The range of the coordination numbers has also been studied, various methods including 2D thin sections (Oren and Bakke, 2003) and X-ray computer tomography (Lindquist, 2000) revealed a range up to 16.

Another important parameter to measure in order to preserve the topology is the connectivity coefficient (Figure 2), often described by the Euler-Poincaré characteristic (Hadwiger, 1957). The volumetric Euler number (X_v) can be used (1) to provide an unbiased estimation of the connectivity (Vogel and Roth, 2001). This number is based on the fundamental topological properties of a domain composed of several pores.

$$X_V = \frac{N - C + H}{V} \tag{1}$$

In which N is the number of isolated objects, C is the number of redundant connections or loops, H is the number of completely enclosed cavities or hollow spheres and V is the volume (Figure 2). The volumetric Euler characteristic is a single value, describing the connectivity in the entire medium, decreasing with an increasing connectivity.

2.3.2 Geometry

An important feature of almost all pore network models is that the resistance to flow is assumed to come from the pore throats only (Raoof and Hassanizadeh, 2012). The pore throat acts as a bottleneck while the pore bodies are adopted to be liquid holding containers, defining the porosity of the porous medium. Although this bottleneck effect may be the case for single phase flow, its concept changes entirely when looking at multiphase flow. Raoof and Hassanizadeh (2012) argued that a pore body which is occupied by both the wetting phase and nonwetting phase can have similar flow resistance as a pore throat. This phenomenon is governed by the shape of the concerning volume. The wetting phase may occupy the crevices of pores with a rough or angular cross-section, while the nonwetting phase resides in the centre of the pores (Lenormand et al., 1983). This has not only a considerable effect on the fluid flow, but also on the connectivity and the proportion of the wetting and nonwetting phase. To simulated multiphase flow, a spatially correlated, disordered pore network is required, with each pore allowing multiple phase occupancies (Blunt, 2001).

$$G = \frac{A}{P^2}\ (2) \qquad \textit{(2)}$$

The shape of the angular cross is defined by the shape factor (equation 2), a dimensionless number describing the relationship between the cross-sectional area and its perimeter (Mason and Morrow, 1991). A circle has the highest shape factor, with a value of 0.0795, resulting from an optimal proportion between cross-sectional area and perimeter. For very elongated geometries, which have a small area compared to its perimeter, the shape factor approaches zero (Joekar-Niasar et al., 2010). Many different geometrical shapes (Figure 3) have been considered over the years and even a

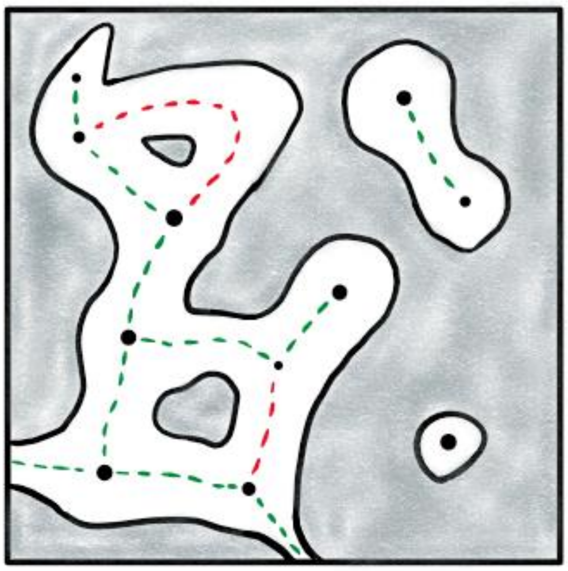


Figure 2. *Schematic drawing of a fictional pore space (white) with skeleton (dashed line) and pore bodies (black dots). There are three isolated objects, two redundant connections (red dashed line) and one completely enclosed cavity (bottom right corner). A redundant connection can be removed without creating a new isolated object, hence, the redundant connection in the bottom could also be one of the other three throats in that specific loop.*

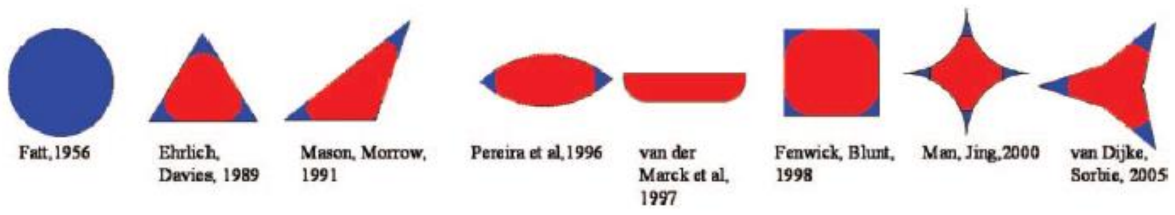


***Figure 3**. Various geometrical shapes used as cross-sectional area in different studies. The red colour represents the nonwetting fluid in the centre of the pore throat while the blue colour represents the wetting fluid, occupying the crevices of the pore (Joekar-Niasar and Hassanizadeh, 2012).*

combination of several shapes has been implemented in various pore network models (Figure 4). In addition to the shape factor, Joekar-Niasar et al. (2010) have shown that the number of vertices also has a significant effect on the hydraulic properties and should be taken into consideration.

Another parameter influencing the flow resistance is the throat length (Figure 5). However, the transition from pore body to pore throat is ambiguous, raising difficulties measuring this entity. A common way to compute the throat length is to use the aspect ratio (3) between the radii of the connected pore bodies (r_1 and r_2) and the throat radius (r_t) in combination with a simplification of the Hagen-Poiseuille law (4).

$$k_1 = \frac{r_1}{r_t} \; and \; k_2 = \frac{r_2}{r_t} \tag{3}$$

$$L_e = L_1 \frac{1 + k_1 + k_1^2}{3k_1^3} + L_2 \frac{1 + k_2 + k_2^2}{3k_2^3} \tag{4}$$

In the most recent years of pore network modelling the three-dimensional pore body shape has shifted from spherical to cubic (Xiong, 2016), accounting for wettability phenomena in the pore body. Joekar-Niasar and Hassanizadeh (2010) presented a pore network model with truncated octahedrons as pore bodies. Although this allowed for simultaneous existence of both wetting and nonwetting fluids and unique shaping of each pore body, it was limited by a maximum coordination number of six. The size of the pore body is often determined by the radius of the maximal inscribed sphere or by

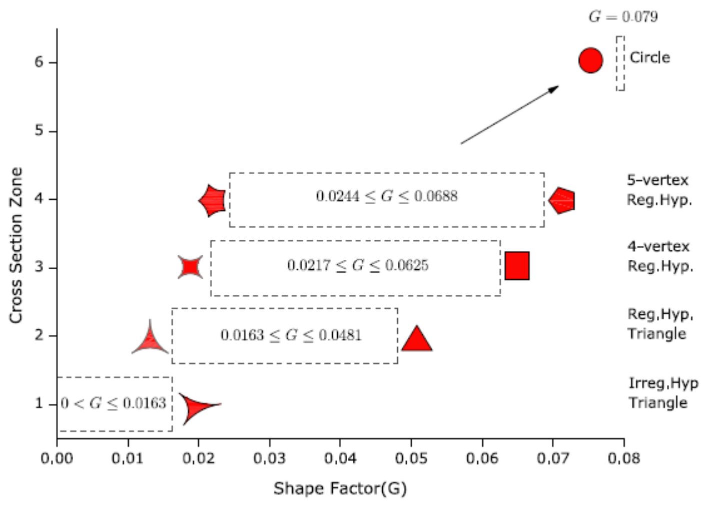


***Figure 4.** Shape factor distribution implemented by Joekar-Niasar (2010) considering irregular hyperbolic triangles (3 vertices), regular hyperbolic polygons (more than 3 vertices) and circles. The range for each geometrical shape differs widely.*

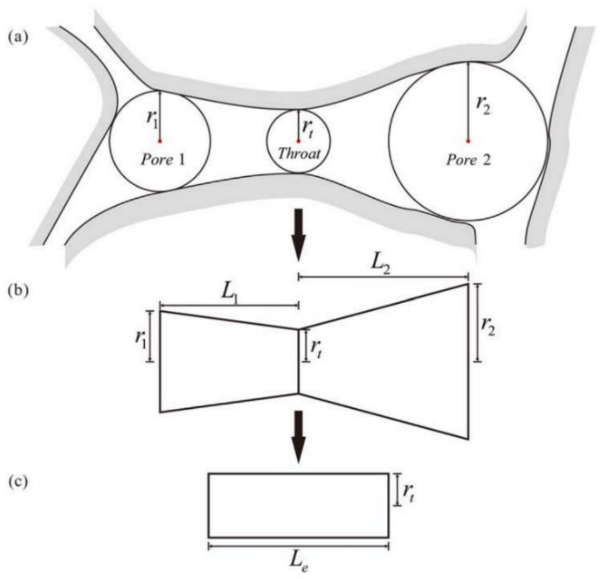


Figure 5*. Schematic drawing for calculating the effective throat length (L_e). The pore space (a) is simplified by sticking together two frustums (b) which is then simplified to a cylindrical tube (c). The shape of the throat is not actually circular, as this is determined by the shape factor. (Yi et al., 2017)*

the radius of the volume equivalent sphere, respectively underestimating and overestimating the pore volume. Ultimately the maximal inscribed sphere estimation is better suited for flow, while the equivalent volume method thrives in regard to solute transport and porosity. The aspect ratio (3) also plays an important role in snap-off, which can influence trapping during imbibition (Joekar-Niasar et al., 2009). Furthermore, must be emphasized that most studies assume the pore throat volume negligible in comparison with the pore body volume, their contribution to the flow resistance is however significant (Raoof and Hassanizadeh, 2012). The size of the pore throat is often related to the maximal inscribed sphere.

2.4 Pore Network Models

There are three main construction technique to generate a pore network model; i) statistical, ii) geological and iii) direct mapping.

2.4.1 Statistical Reconstruction

Statistical reconstructions are based on the hydrological properties of porous material, prospering during a time in which computed microtomography was constrained by its resolution, speed and expenses (Ioannidis and Chatzis, 2000). The essence of stochastical reconstruction is to relate one or more macroscopic transport coefficients to essential geometric and topological attributes of the microstructure (Liang et al., 2000). Although the hydrological characteristics are representative, two different materials can have exactly the same parameters but can spatially vary widely. Liang et al. (2000) found that porous media reconstructed from the same porosity and correlation function can exhibit marked differences in geometry and connectivity, which correlate with differences in specific surface area. Furthermore, Silin and Patzek (2006) showed that the dimensionless capillary pressure curve does not depend on the porosity or the size of the sample, rather it depends on the intrinsic geometry and connectivity of the pore space. It is thus crucial to understand the local intrinsic properties of a porous material in order to simulate fluid flow and hydrogeological transport phenomena. However, the need for up-scaling from the typically small imaged volume to larger domains leads to the need for construction of statistically representative networks (Xiong et al, 2016).

2.4.2 Geological Reconstruction

Geological reconstruction can be compared to the geological process of diagenesis, which is directly its major drawback. The modelled diagenesis procedure is only applicable for regular shaped grains and is thus not applicable for limestone. The pioneers in this field of research were Bryant, Blunt and their co-workers (Bryant and Blunt, 1992; Bryant et al., 1993a, b) who successfully modelled compaction of a sandstone by simulating a pack of equally-sized spheres and moving the centres of the sphere vertically, allowing the spheres to overlap. This model was able to accurately predict relative permeability and capillary pressure. Bakke and Oren improved their work by developing a reconstruction method in which the grains size distribution was included by generating spheres with different radii. Albeit the success of Oren, Bakke and its fellow researchers (Bakke and Oren, 1997; Lerdahl et al., 2000; Oren and Bakke, 2002; Oren et al., 1997), being able to simulate multiphase flow and accurately predicting relative permeability for a variety of water-wet sandstones, there were still two more limiting concerns besides the restriction of spherical grains. First of all is the geological process by which a rock is formed unknown, or very challenging. Furthermore, is the characterization of pore shape and wettability not fully understood (Xiong et al, 2016), especially considering the dynamics of the morphological properties.

2.4.3 Direct Mapping

The most difficult task when creating a pore network model is identifying and specifying the pore bodies and pore throats (Jiang et al., 2007). According to Prodanovic et al. (2006) all pore network models can be categorised into two types, pore detection or throat construction. Currently, most pore network models are based in the former type, first defining the pore bodies before constructing the pore throats. However, the identification of a pore body can be an art on itself. This process can also be classified into two categories, geometry- or topology based. The former method makes use of the distance map within the pores, defining pore bodies as local maxima and pore throats as the connecting local minima. The topological based method makes use of the amount of connections, a

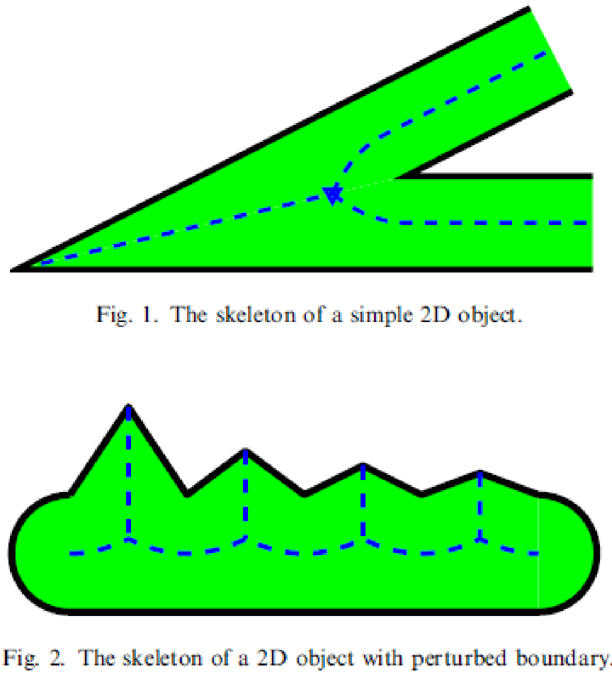


Fig. 1. The skeleton of a simple 2D object.
Fig. 2. The skeleton of a 2D object with perturbed boundary.

Figure 6. *Skeletonization of the pore space by using the medial axis algorithm, difficulties arise due to perturbations on the edge of the pores (Silin and Patzek, 2006).*

pore body will be defined as a junction and pore throats will be defined as nonjunctions (Jiang et al., 2007). The most important feature which both the geometrical and topological based approach have in common is that they both make use of a skeleton, essentially the backbone of each pore network model. The different methods of generating a skeleton will be described in the following paragraphs.

2.4.3.1 Medial Axis Thinning

The medial axis algorithm is based on finding the local maxima in the distance map and reconnecting them by using the local minima (Baldwin et al., 1996). The resulting skeleton is exactly centred in the pore body but does not necessarily preserve the topology (Figure 6), as the path reconstruction may produce redundant points on the skeleton (Jiang et al., 2007). A thinning algorithm is fairly similar but eliminates the redundant points by burning the edges of the pore space (Lindquist et al., 1996). The elimination procedure is iterated until no further voxel can be removed, assigning the residual voxels as the medial axis. The disadvantages of the thinning algorithm arise from the verifiability and reproducibility due to the orientation of the image (Figure 7), altering the order of elimination and significantly changing the results (Silin and Patzek, 2006).

2.4.3.2 Maximal Ball Algorithm

The maximal ball algorithm (Silin and Patzek, 2006) calculates the maximal inscribed sphere for each voxel in the pore space. The largest maximal sphere will consume all the overlaying smaller spheres, generating a network of maximal balls which are defined as the pore bodies, and connecting smaller balls, the pore throats. Al-Kharusi and Blunt (2007) adopted this method and added some hierarchical rules, which are able to handle equally sized spheres. This resulted in more credible results, but also increased the computing time and coordination numbers tremendously. Dong and Blunt (2009) continued improving this model, decreasing the computing time and error by incorporating a searching algorithm. This model however, had flaws concerning the segmentation of smaller pores into one large pore, making it unusable for samples with relatively large pore throats, like limestones. Also, the computational time of the algorithm is still drastically longer than algorithms making use of a medial axis algorithm.

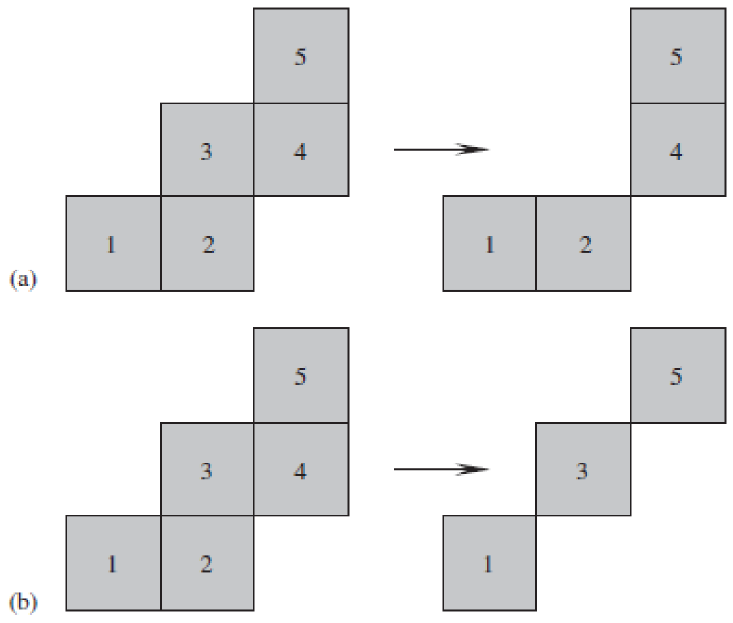


Figure 7. *Different skeletons produced by the same thinning algorithm, resulting from a different order of elimination of redundant voxels (Silin and Patzek, 2006).*

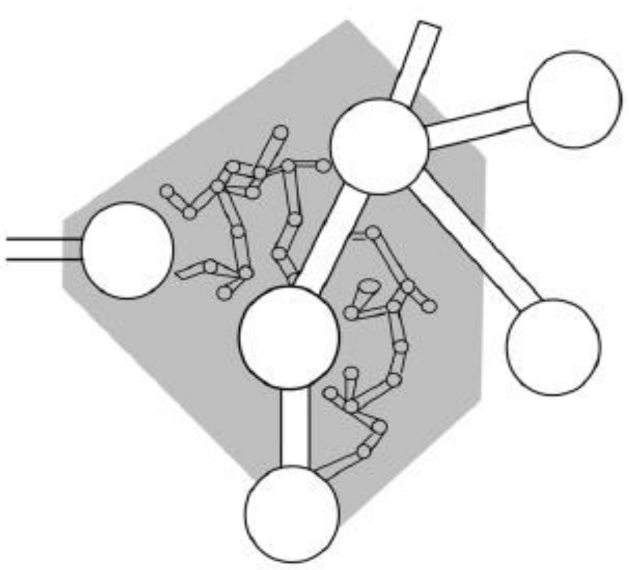


Figure 8. *A pore network model with two distinct length scales incorporated (Xiong et al., 2016).*

2.4.4 Two-scale Reconstruction

Another type of PNM is two-scale reconstruction, which cannot be distinctively separated from the prior PNM methods as it makes use of either statistical reconstruction or direct mapping. Despite the advances of pore network modelling, the simulation of transport properties in heterogeneous rocks still remains an open issue due to the very broad pore size distributions in some porous media (Xiong et al., 2016). Several methods have been proposed to overcome this issue, most noticeable the advances by Jiang et al (2013) and Prodanovic et al. (2015) who integrated network models with distinct length scales (Figure 8). However, the shortcoming of these models is that it is computationally costly due to the large number of network elements (Bultreys et al., 2015). This reconstruction method will not be discussed in further detail because its shortcomings contradict with our research objectives.

3 Flow Simulations

The flow simulations through the extracted pore network model are simulated using PoreFlow (Raoof et al., 2013). One key feature of PoreFlow is the further discretization of pore bodies, which allows multiphase flow to occur (Figure 9). Under partially saturated conditions, the nonwetting phase may occupy a large volume of the pore body, allowing the wetting phase to only persist at the edges, drastically decreasing the conductance. Ignoring this effect may result in heavy underestimation of the relative permeability (Bryant and Blunt, 1992; Al-Kharusi and Blunt, 2008).

Two important assumptions are made before computation; the fluid is incompressible and the flow is laminar. The sides of the pore network perpendicular to the flow direction are set to constant head boundaries, while the sides parallel to the flow direction are no flow boundaries. The discharge through a drained pore throat is calculated using the Hagen-Poiseuille equation (5).

$$q_{ij,tot} = \sum_{k=1}^{N_{edge}} g_{ij,k} \frac{P_j - P_i}{l_{ij}} \tag{5}$$

In which q_{tot} is the total volumetric flow rate, g_k is the conductance of one specific edge, P is the pressure and l is the length. All parameters are computed between two pore bodies, i and j, hence generating the parameters for each throat specifically. The number of throat edges is implemented to simulate partial saturation. Each pore body is represented by a cube while the shape of the throat is dependent on the shape factor.

The continuity equation is applied on the pore body (6), utilising the fact that the sum of all discharges should be zero.

$$\sum_{n=1}^{N_{edge}^{CU,i}} q_{i,n} + \sum_{k=1}^{z_i} q_{ij,tot} = 0 \tag{6}$$

In which $\mathrm{N}_{\mathrm{edge}}^{\mathrm{CU,i}}$ is the number of edges through which corner unit i is connected to other corner units n, within the same pore body. The first term thus represents the discharge within the different corner units. The second term, in which z_i is the coordination number, represents the discharge for the connected throats.

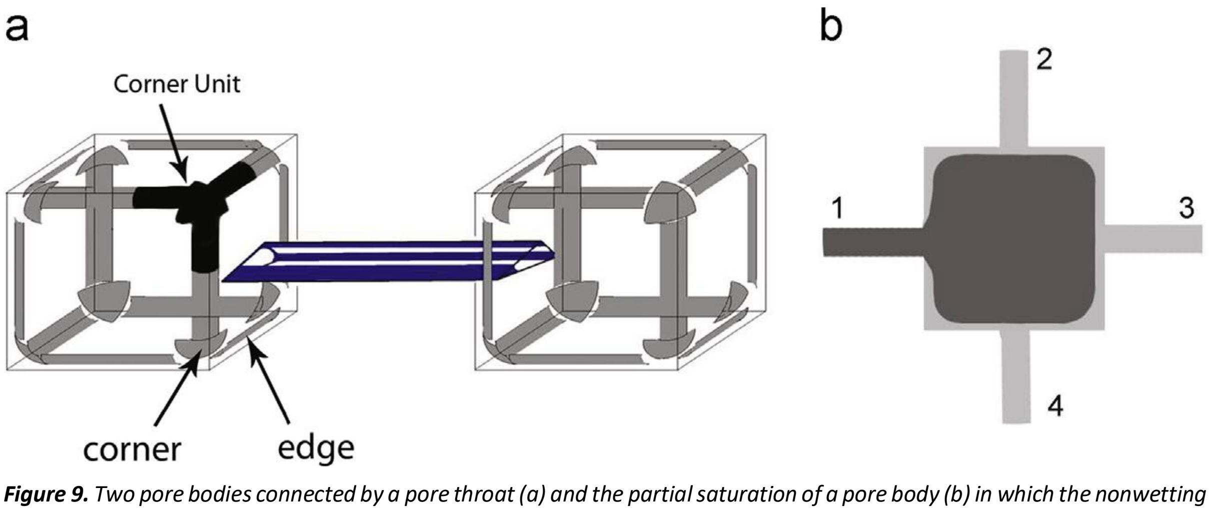


Figure 9. *Two pore bodies connected by a pore throat (a) and the partial saturation of a pore body (b) in which the nonwetting phase is represented by the dark grey colour and the wetting phase by the light grey colour (Raoof et al, 2013).*

The relative permeability (k_r) at a given capillary pressure is calculated using the total discharge through the network (Q_{tot}) in combination with Darcy's law (7).

$$k_r = \frac{\mu Q_{tot}}{kA\frac{dP}{dl}} \tag{7}$$

In which μ is the dynamic viscosity, k the absolute permeability and A is the area.

4 Percolation Theory

Universal scaling and critical path analysis are two supplementing theories originating from percolation theory. The prior is applicable for volumetric water contents higher but close to the critical water content while the latter operates for higher water saturations. Both theories are applied to find the rate-limiting conductance value (Hunt, 2005). Critical path analysis (CPA) was developed by Ambegaokar et al. (1971) to study electron transport in amorphous semiconductors. Fortunately, electrical conductivity is defined completely analogously to the hydraulic conductivity in a porous medium (Hunt, 2005), making it also applicable for this study. Critical path analysis prevails when the porous medium is highly heterogeneous with a broad pore size distribution dominated by pore throats whose radii are slightly larger than the critical radius. The rest of the pore throats do not significantly contribute to flow, and therefore, can safely be ignored (Ghanbarian et al., 2016).

4.1 Fractal Dimensionality

Fractal dimensionality is one of the features incorporated in CPA. Fractal models provide an analytical framework for prediction, being able to represent highly complex natural media with a paucity of relevant parameters (Hunt and Ewing, 2009). Rieu and Sposito (1991) were the first to establish a model (8) which related the fractal pore dimensionality (D) to a finite range of pore sizes ($r_{min} \leq r \leq r_{max}$) and the porosity (ϕ).

$$\varphi = 1 - \left(\frac{r_{min}}{r_{max}}\right)^{3-D} \tag{8}$$

The Rieu and Sposito (RS) model was intended to represent soils, not fractured rock. Also, the model can only be utilised when both the pore space and solid matrix are fractal and self-similar, making it scale invariant. This was verified for varies porous media, in particular soils, by Turcotte (1986). One complication of the RS model is the use of a discrete, truncated random fractal, generating only pore throats with a discrete radius, instead of a continuum representation. This issue was solved by Hunt and Gee (2002b) by exploiting a probability density function (9) of the pore throat radius which has a continuous, power-law form.

$$W(r) = \frac{3-D}{r^{3-D}} r^{-1-D} \tag{9}$$

The porosity should be equal to the integral of the volume of the pore (r^3) times the probability density function, W(r), over the minimum and maximum pore size, $r_{min} \leq r \leq r_{max}$ (10). This approach agrees with Rieu and Sposito (1991), which has been verified to predict pressure-saturation curves by several different studies (Filgueira et al, 1999; Bird et al., 2000; Hunt and Gee, 2002a).

$$\varphi = \frac{3-D}{r_{max}^{3-D}} \int_{r_{min}}^{r_{max}} r^3 r^{-1-D} dr = 1 - \left(\frac{r_{min}}{r_{max}}\right)^{3-D} \tag{10}$$

For fractal description of the solid volume of a porous medium to be accurate, a log-log representation of the cumulative volume versus the particle size should produce a straight line (Hunt and Gee, 2002b). If the pore space is also fractal over a finite range of pore radii, the fractal description is probably valid over the same range. This prediction agrees with the Arya-Paris model (1981) and implies that the radius of both the largest and smallest pore is fairly consistent with the largest and smallest particle

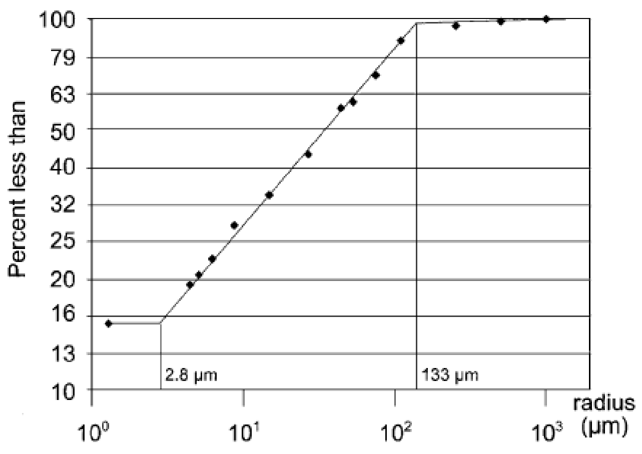


Figure 10. *Graphical construction of finding r_{min} (2.8 µm) and r_{max} (133 µm) (Hunt and Gee, 2002).*

radius. The fractal dimensionality can thus be found when both the proportion between the largest and smallest pore and the porosity is known (Figure 10).

$$D = 3 - \frac{\log(1-\varphi)}{\log\left(\frac{r_{min}}{r_{max}}\right)} \tag{11}$$

4.2 Critical Path Analysis

Laminar flow in a cylindrical tube can be described by Poiseuille law (12), in which µ is the dynamic fluid viscosity, g is the conductance, r the radius and l the length of the tube.

$$g = \frac{\pi}{8\mu}\frac{r^4}{l} \tag{12}$$

However, pore throats appear in many different geometries. The assumption is made that the effective radius and length of identical shaped throats scale identically to the characteristic radius and length. Furthermore, self-similarity implies that r α l, which results in the following proportionality (13).

$$g \propto r^3 \tag{13}$$

The proportionality between the probability density function of a pore throat and the radius of a pore throat (14) can be found using equation (9).

$$W(r) \propto r^{-1-D} \tag{14}$$

Furthermore, the relation between the probability density functions of the throat radii and the conductance can be expressed.

$$W(r)dr = W(g)dg \tag{15}$$

Using equations (13), (14) and (15) the proportionality of the probability density function of the conductance can be found.

$$W(g) \propto g^{-\frac{D}{3}} \tag{16}$$

Combining critical path analysis with the theory from fractal dimensionality ultimately generates an equation for the unsaturated hydraulic conductivity. To achieve this goal, the lower limit of equation (10) is altered to the critical radius (r_c), which is the smallest radius encountered on a percolation path. This results in the critical water content, which is equivalent to the minimum amount of water generating a percolation path through the porous medium.

$$\theta_c = \frac{3-D}{r_{max}^{3-D}} \int_{r_c}^{r_{max}} r^3 r^{-1-D} dr = 1 - \left(\frac{r_c}{r_{max}}\right)^{3-D} \tag{17}$$

The partial saturation is accounted for by altering the upper limit of equation (10). In theory this should represent film flow permitting the porous medium to adjust the water content by entirely emptying the pores larger than the equilibrium radius. This complies with the Young-Laplace equation, larger pores empty before smaller ones. Hysteresis is not considered, as pores throats cannot empty partially.

$$\theta = \frac{3-D}{r_{max}^{3-D}} \int_{r_{min}}^{r} r^3 r^{-1-D} dr = \left(\frac{r^{3-D} - r_0^{3-D}}{r_{max^{3-D}}}\right) \tag{18}$$

The percolation condition of the lower boundary should also be applied to the partial saturation.

$$\theta_c = \frac{3-D}{r_{max}^{3-D}} \int_{r_c}^{r} r^3 r^{-1-D} dr = \left(\frac{r}{r_{max}}\right)^{3-D} - \left(\frac{r_c(\theta)}{r_{max}}\right)^{3-D} \tag{19}$$

Combining equations (17), (18) and (19) gives an equation for the critical pore throat radius.

$$r_c(\theta) = r_c(\theta = \varphi) \left[\frac{1 - \varphi + \theta - \theta_c}{1 - \theta_c}\right]^{\frac{1}{3-D}} \tag{20}$$

Critical path analysis states that the hydraulic conductivity of a porous medium is controlled by the hydraulic conductance of the rate-limiting pore throat (Hunt, 2001), which is proportional to the cube of the critical radius (13). This results in a formula for the unsaturated hydraulic conductivity (Hunt and Gee, 2002).

$$K(\theta) = K_s \left[\frac{1 - \varphi + \theta - \theta_c}{1 - \theta_c}\right]^{\frac{3}{3-D}} \tag{21}$$

4.3 Balberg Nonuniversality

Balberg (1987) determined that distributions with a negative power-law distribution (22) approaching g = 0 should obey equation (23) to describe the conductance, otherwise leading to non-universal exponents of conduction. This theory, i.e. Balberg nonuniversality, uses the average resistance along the critical path rather than the largest resistance.

$$W(g) \propto g^{-\alpha} \tag{22}$$

$$\sigma \propto (p - p_c)^{\frac{\alpha}{1-\alpha}} \tag{23}$$

As equation (16) indeed has a negative power-law distribution we need to apply Balberg nonuniversality, which yields equation (24) for the conductance depending on the water content.

$$\sigma \propto (\theta - \theta_c)^{\frac{\alpha}{1-\alpha}} \tag{24}$$

As the result of Balberg needs to be mimicked, the same assumption is applied that the conduction paths close to zero conductance is basically 1D in character. To achieve this the bulk resistance distribution is cut off at g_c^{-1} and integrated, with the result that $\langle g^{-1}\rangle^{-1}$ is always given by $g_c^{-\alpha}$ (Hunt, 2005). This method is applied to equation (21) resulting in equation (25) (Hunt, 2005).

$$K(\theta) = K_s \left[\frac{1 - \varphi + \theta - \theta_c}{1 - \theta_c}\right]^{\frac{D}{3-D}} \tag{25}$$

4.4 Transition from CPA to Universal Scaling

A problem arises when the water content approaches the critical water content. A requirement of percolation theory is that the hydraulic conductivity vanishes at the critical water content, which is not valid for equation (25). Universal scaling provides the solution, which states that the unsaturated hydraulic conductivity should vanish according to equation (26).

$$K(\theta) = K_0(\theta - \theta_c)^{\mu} \tag{26}$$

Universal scaling (26), or percolation scaling, implies that the hydraulic conductivity is based on the topology of the porous medium depending on the occupation of the pore throats. On the other hand, critical path analysis (25) depends on the geometry, or bottleneck, of the pores. It is evident that these theories should supplement each other to get the best results. However, the upper limit of percolation scaling has never been answered satisfactorily (Ghanbarian et al, 2014). So far the best way of finding this cross-over water content is by setting equal the hydraulic conductivity equations (27) and their derivatives (28) at the cross-over saturation (Θ_x).

$$K_s\left[\frac{1-\varphi+\theta_x-\theta_c}{1-\theta_c}\right]^{\frac{D}{3-D}}=K_0(\theta_x-\theta_c)^2 \tag{27}$$

$$\frac{D}{3-D}\frac{K_s}{(1-\theta_c)^{\frac{D}{3-D}}}\,(1-\varphi+\theta_x-\theta_c)^{\frac{D}{3-D}-1}=2K_0(\theta_x-\theta_c) \tag{28}$$

Note that μ is replaced by 2, which is valid for three-dimensional porous media. Solving these equations for the cross-over water content yields equation (29).

$$\theta_x=\theta_c+\frac{2(1-\varphi)}{\frac{D}{3-D}-2} \tag{29}$$

Now K_0 can be determined from equation (27) as well.

$$K_0=K_s\left[\frac{1-\varphi+\theta_x-\theta_c}{1-\theta_c}\right]^{\frac{D}{3-D}}(\theta_x-\theta_c)^{-2} \tag{30}$$

4.5 Critical Volume

A crucial value intertwined with percolation theory is the critical water content (Θ_c), which is equivalent to the minimum amount of water generating a percolation path through the porous medium. For natural porous media the critical water content is unknown (Ghanbarian and Hunt, 2012). However, Moldrup et al. (2001) developed an empirical formula (31) relating the surface area (A) and volume (V) to the critical water content (R^2 = 0.99). Typical values range from 0.09 for sandy soils, to 0.13 for loamy soils and 0.19 for clayey soils (Moldrup, 2001).

$$\theta_c=0.039\left(\frac{A}{V}\right)^{0.52} \tag{31}$$

The drawback of using this method is the use of specific surface area data which is not always available. Another possibility to extract the critical water content is from the water retention curve. According to van Genuchten (1980), the critical water content is equivalent to the last measured water content before the gradient ($d\Theta/dh$) becomes zero. However, this method requires pressure-saturation curves to be available.

It is important to emphasize that there are quite a few restrictions coupled to percolation theory. First of all, is the theory not applicable for multiphase flow, as pore throats are completely filled by either water or air due to a contact angle of the wetting phase (water) being nearly 0 degrees. This allows the approach to use cylindrical tubes, each with a radius determined from the pore-throat size distribution. Furthermore, pore bodies are not considered as flow is only limited by the constraints applied by the throats, which also results in the absence of trapping. Besides, the pore size distribution of the throats is assumed to be power-law functions for CPA to work. Lastly, universal scaling is applicable for saturation values close to the critical water content until the crossover saturation ($S_c \leq S \leq S_x$; $\Theta_c \leq \Theta \leq \Theta_x$) while CPA is applicable from the crossover saturation until fully saturated conditions ($S_x \leq S \leq 1$; $\Theta_x \leq \Theta \leq \phi$).

5 Method

5.1 Pore Network Model Extraction

A pore network model must satisfy the following criteria to predict multiphase flow properties: i) topology preservation, ii) a single voxel width skeleton, iii) medial location of skeleton and iv) integration of geometry (Jiang et al., 2007). Several steps need to be performed to accomplish these criteria; i) the CT-images have to be processed for binarization and then skeletonized, ii) the pore body and throat locations need to be determined and iii) their geometrical features need to be partitioned.

5.1.1 Binarization and Skeletonization

The provided porous media samples are represented by stacks of black-and-white images. The grey scale of these images range from zero until the bit size of the image, in which the former represents solid and the latter corresponds to pore space. Everything in between needs to be assigned to these categories as well, this is done by choosing a threshold which coincides with the porosity of the sample. However, the images need to be processed before the threshold can be determined, which is done in *Fiji* (Schindelin et al., 2012).

The first step in this process is to alter the *colour balance* to assign the voxels which are either clearly solid or pore space to its corresponding value. Then a *3D median filter* is applied to remove noise but retain the edges of the grains. Next, the *threshold* needs to be determined by matching the proportion of pore space voxels divided by the total amount of voxels to the experimentally found porosity. However, *3D fill holes* should be applied before to remove physically impossible floating grains.

When the threshold is calibrated correctly the binarized images can be skeletonized using *Skeletonize3D* (Arganda-Carreras et al., 2010). This plugin makes use of the parallel thinning algorithm developed by Lee et al. (1994) which is able to extract both the medial surfaces and the medial axes of a three-dimensional object. The algorithm finds all removable surface voxels of the pore space for each iteration in a symmetrical manner, assuring the medial position of the remaining pore voxels. The iteration process continues until the width of the skeleton is only one voxel. A key feature of the algorithm is the preservation of the connectivity, this is done by checking the Euler characteristic and the number of connected objects (Homann, 2007). The first three research criteria are satisfied by using this algorithm. Another advantage of using this algorithm is the short computation time, which is much faster than the maximal ball method developed by Silin and Patzek (2006). The created skeleton is analysed with *AnalyzeSkeleton,* which provides detailed information about the branches: start locations, end location, contributing voxels, Euclidean length and tortuous length. The binarized images are also analysed with *BoneJ2*, which computes the surface area, volume and the fractal dimensionality. The surface area is extracted using a marching cube algorithm (Lorensen and Cline, 1997). The fractal dimensionality is computed using a box-counting algorithm (Fazzalari and Parkinson, 1996), which returns the number of boxes (N) and the size of the boxes (s) for each iteration. The eventual fractal dimensionality is calculated by fitting a linear regression line through the returned values on a log scale. A simple manual for the binarization procedure is provided in Appendix C.

5.1.2 Locating Pore Bodies and Throats

The method developed in this research separates pore bodies and pore throats by using the topological features of the skeleton. Every skeleton voxel can be partitioned into one of the following four groups: i) a junction voxel with three or more connected voxels, ii) a non-junction voxel with exactly two connected voxels, iii) a dead-end voxel with one connected voxel or iv) an isolated voxel with no connected voxels. All non-junction voxels will be considered throats while the other types of voxels will represent pore bodies. This procedure of classifying pore bodies and pore throats coincides with the method used by Jiang et al. (2006) and exceeds in topology preservation.

A problem emerging by using this classification and the algorithm developed by Lee et al. (1994) is that one irregular shaped pore body can consist of multiple junction voxels (Figure 11), resulting in one pore body consisting of several smaller pore bodies. Consequently, there is a possibility of finding throats within this irregular shaped pore body artificially increasing the resistance. Other studies solved this problem by merging two pore bodies with a short throat between them and assigning the average location of those two pore bodies to the newly emerged pore body (Lindqiust et al., 1996; Jiang et al., 2007; Bultreys et al., 2015). However, determining which throats are short enough in order to merge the connected pore bodies is an arbitrary process and should thus be avoided. Also, changing the location of the newly emerged pore body by taking the average locations causes the geometry to change. Here we propose to use the maximal ball algorithm for the overlapping pore bodies. This method accounts for the problems mentioned above and does not generate unrealistically high coordination numbers.

The pore body merging algorithm pursues the following 7-stage procedure:

1. Determine the maximum inscribed sphere for each pore body using the distance map between the solid and pore space voxels.
2. Define clusters by looking at which maximum inscribed spheres overlap. Each maximum inscribed sphere allocated to a cluster has to overlap with at least one other maximum inscribed sphere to become a cluster.
3. Order the maximum inscribed spheres allocated to each cluster by decreasing size.
4. Assign the first master of each cluster. The master can be defined as the largest inscribed sphere in a cluster (Silin and Patzek, 2006). The location of the master is fundamental for defining the location of the irregular shaped pore body, as it has the largest share in the volume of the pore space.
5. Examine the next maximum inscribed sphere in the cluster. If it the maximum inscribed sphere overlaps with a larger master or larger slave, it becomes a slave, if not, it becomes a master.
6. Repeat step 5 until all the allocated pore bodies to a cluster are either a master or a slave.
7. Remove all the slaves from the clusters and assign the essential data coupled to a slave to the corresponding master.

The algorithm is superior in preserving the geometry of the porous medium, as the fluid flow through an irregular shaped pore should not be restricted by throats artificially generated within this pore. An additional phenomenon occurring are pore bodies with a coordination number of two due to the merger of a dead-end voxel and a junction voxel. However, this does not give any issues regarding the flow simulation and gives more detail about the pore network.

Pore bodies with a coordination number of one, and a pore body radius smaller or equal to the throat radius are also removed. This is done to reduce the amount of dead-end pores artificially created by the medial-axis algorithm due to perturbation in the pore surface. Pore bodies with a coordination number of zero are removed, as they are insignificant for the flow and the porosity, however, isolated clusters are maintained. Although these clusters do not contribute to the overall flow as well, they are important to establish the porosity and could influence the connectivity when fractures or dissolution effects come into play.

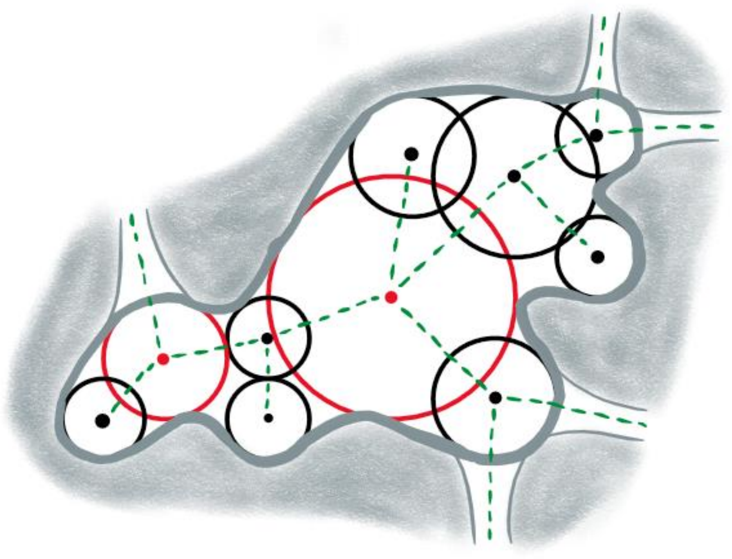


Figure 11. *Schematic drawing of an irregular shaped pore. The maximum inscribed spheres are all part of one cluster, in which the red spheres represent the masters and the grey spheres represent the slaves. The green dotted line represents the medial axis of the network.*

Pore bodies which are connected by multiple throats are also altered, albeit slightly different. The throat area, throat perimeter and throat radius are all individually summed and appended to a single throat representing the multiple throats. The average throat length of the multiple throats is appended as the new throat length of the single pore throat. These alterations give the most realistic results as most multi connected pore bodies are represented by two pore throats which have similar geometries, dictating the replacement throat to be an average of the two.

5.1.3 Partitioning Geometrical Features

Although the maximum inscribed sphere is used to merge the overlapping pore bodies, a watershed segmentation algorithm is used to determine the final radius of the pore bodies. The pore body radius will be equal to the radius of a sphere with an equivalent volume. This is done to get a correct representation of the pore volume which is important for solute transport and multiphase flow. Before the pore body radii are calculated, the throat volume is deducted to not overestimate the porosity. The throat volume is calculated using the formula for a cylinder dependent on the throat length and throat radius. The watershed segmentation makes use of markers, acting as minima, and a new distance map between these markers and the solid. The markers are equal to the inscribed spheres of the pore bodies. The pore space is filled up from these markers outwards, until the entire pore space is segmented to a pore body. The binary map of the sample is used as a mask to only partition the

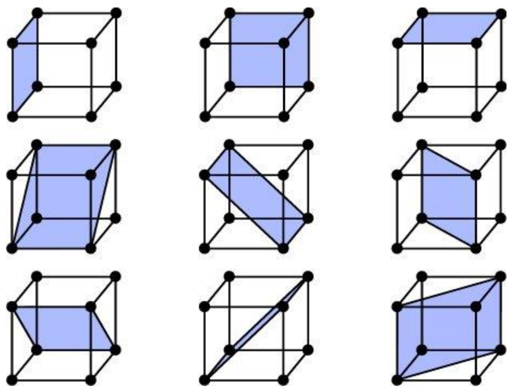


Figure 12. *The nine orientations of the planes incorporated in the pore network model.*

pore space. The watershed segmentation is also used to determine the boundary pores of the sample, which can either by i) inlet pores, ii) outlet pores or iii) side pores. Partitioning the boundary pores to one of these categories is done by looking at the coordinates of the voxels allocated to a specific segment. E.G. when the Z-axis is chosen as the flow direction, all segments in which pore voxels are found with a z-coordinate of 0 can be defined as inlet pores and all pore voxels with a z-coordinate equal to the length of the sample can be defined as outlet pores. This same technique can then be applied to the x-axis and the y-axis, however, all boundary pores will be assigned to be side pores.

There are three geometrical throat properties which need to be determined: i) the shape factor, ii) the throat radius and iii) the throat length. The shape factor is determined at the throat voxel with the smallest inscribed sphere, which is found using the distance map. If several throat voxels are found with the same smallest inscribed sphere, the median voxel of those is used. The area and perimeter are scanned at this same location in nice directions, only keeping the values allocated to the smallest area. The smallest area is most likely the bottleneck within a pore throat and is thus vital to predict the overall flow. Nine plane directions (Figure 12) are taken into account because cubic voxels are being used. The maximum error between the orientation of the plane and the direction of the throat is therefore 22.5°, which results in a maximum error of 4% for very low shape factors. Here we propose a new algorithm which can both determine the area and the perimeter of a throat by finding the polygon making up the shape of the throat. The algorithm is not affected by isolated solids within the pore throat.

1. Scan the pore space in a horizonal and a vertical line from the throat location until solid is reached. The voxels found are allocated to the throat polygon and are shaped like a cross.
2. Scan vertically from the voxels which were found with the horizontal scan and vice versa for the other voxels. The scan stops for each row or column individually when solid is reached. Assign the newly found voxels to the polygon.
3. Repeat step two for the voxels which are solely found by either the horizontal or vertical scan. However, the scan is also stopped when a voxel is found which is already assigned to the polygon, this prevents the algorithm from checking voxels multiple times. The repetition of step two continues until no new voxels are found.
4. Remove the duplicate voxels.

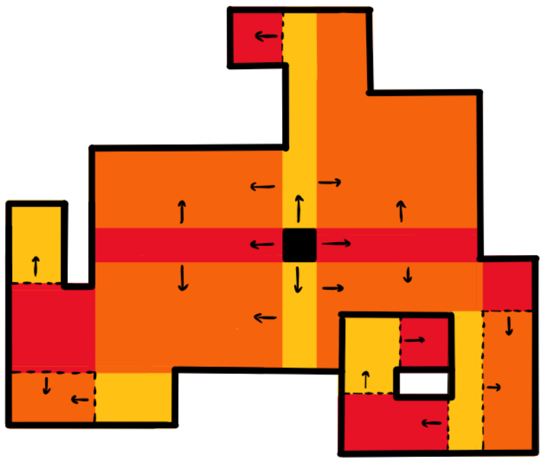


Figure 13. *Schematic drawing of the scanning procedure of a pore throat cross section. The black square represents the throat location from which the first horizontal scan (red) and a vertical scan (yellow) is performed. The arrows indicate the direction in which the scan is performed and the black dotted lines represent the boundary of the scan. Orange pixels are both found by the horizontal and vertical scan.*

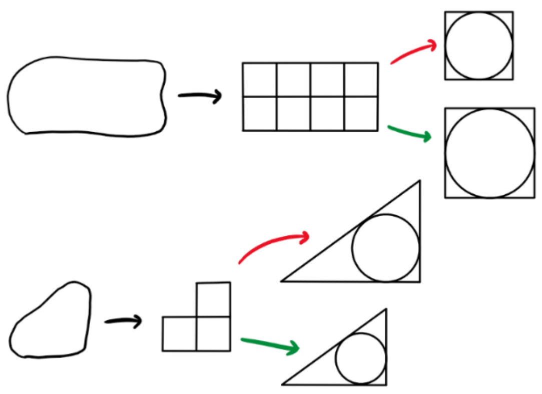


Figure 14. *Schematic drawing of the throat analysation process using the conventional method (red) and our proposed method (green). The cross section of the throat (left) is first binarized (middle) and then converted to pore network data (right). The conventional method tends to underestimate the throat radius for larger pores (top), while it overestimates the throat radius for smaller pores (bottom), as the radius of the minimum inscribed sphere cannot be smaller than 1 voxel. The conventional method generates a throat radius of 1 voxel for both throats while the inradius is equal to 1.33 voxel for the upper throat and 0.75 voxel for the lower throat.*

The area and perimeters can be calculated by counting the number of voxels and by looking at the amount of solid neighbours for each voxel assigned to the polygon. A multiplication factor for both the area and perimeter is applied for diagonally oriented planes. The shape of the polygon only consists of 90° angles, again, due to the use of cubic voxels. Jiang et al. (2007) studied the effects of rounding the corners of the polygon on the shape factor but this did not give conclusive results. The shape factor can both be over- and underestimated due to rougher surfaces or more rounded corners, unfortunately these factors are strongly dependent on resolution and sample type.

The throat radius is, conventionally, determined by taking the minimum inscribed sphere. Here we propose to use the inradius dependent on the shape of the throat as it gives scale independent results. The inradius coincides with the scale independent shape factor proposed by Bultreys et al. (2018). Most rock samples are dominated by pores with a throat radius smaller than 3 voxels wide. The average throat radii found by Dong and Blunt (2009) was smaller than 2 voxels for nine out of the twelve rock samples studied. This can give ambiguous throat radii as the shape can heavily influence this parameter, especially for throats with a width of only 1 voxel (Figure 14). Three different equations have been introduced, dependent on the shape of the throat. Equation (32) is used for shape factors smaller than 0.0481, equation (34) is used for shape factors larger than 0.0625 and equation (33) is used for everything in between. The inradius can also compensate for the fact that the medial-axis is not always perfectly located in the middle of the pore space, accounting for the geometrical flaws caused by the skeletonization algorithm. For throats which have a smaller inradius than inscribed sphere radius, the inscribed sphere radius is taken, although the radius of the inscribed sphere needs to be larger than three voxels. This is done to compensate for extremely rough pore surfaces which are artificially created due to resolution issues (Figure 15).

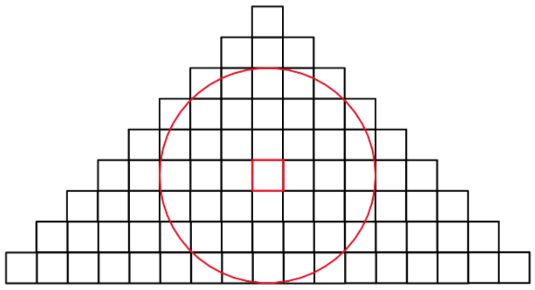


Figure 15. *Schematic drawing of triangular shaped throat with an inscribed sphere radius of four voxels. The area of the throat is 81 while the perimeter is 52 generating an inradius of 3.1 voxels, which is clearly an underestimation.*

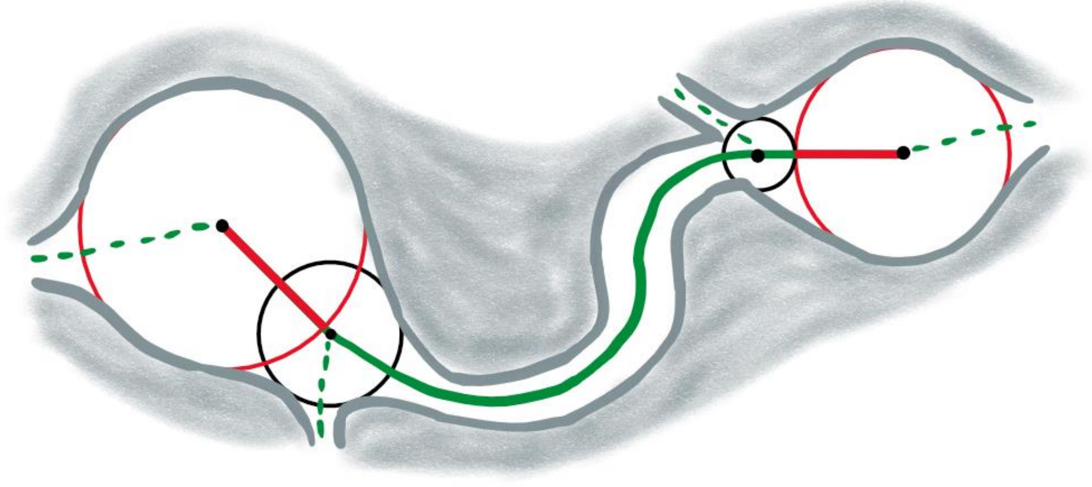


Figure 16. *Schematic drawing of two pore bodies both consisting of one slave and one master, connected by a throat. The eventual throat length is represented by the green line, which is dependent on the tortuous length between the initially connected pore bodies, the Euclidean distance between the masters and the slaves and the radius of the inscribed spheres of the masters (red lines).*

$$r_{triangle} = \frac{A}{0.5P} \tag{32}$$

$$r_{rectangle} = 0.5\sqrt{A} \tag{33}$$

$$r_{circle} = \sqrt{\frac{A}{\pi}} \tag{34}$$

The throat length is determined using equation (3) and (4). The pore body radius is equal to the volume equivalent radius and the throat radius is calculated using the method described above. The eventual throat length is, next to the formulas mentioned above, also altered by deducting the radius of the two connecting master and by adding the Euclidean distance between a possible master and slave (Figure 16) before multiplying it with the aspect ratio. Hence, the throat is assumed to be tortuous outside the pore body but Euclidean within. The pore body radius is deducted because there should be no friction applied within the pore body.

5.2 Pore Network Properties

A total of twelve rock samples (Figure 17) have been analysed in this research; two limestone samples (Carbonate C1 – C2) and ten sandstone samples (one Berea together with S1 – S9). The Berea and S1 sandstone are known to be homogeneous and isotropic while the two carbonate samples are heterogeneous and anisotropic (Dong and Blunt, 2009). S3 and S4 are coarse samples captured with a relatively low resolution, raising difficulties for less robust algorithms (Yi et al., 2017). The twelve samples have also been used by Dong and Blunt (2009) who extracted a pore network model using their maximal ball (MB) algorithm and Yi et al. (2017) who extracted a pore network model using the medial axis algorithm (MA) developed by Lindquist and Venkatarangan (1999). An advantage of using these samples is the fact that the binarization process was already performed and made available. This ensures that the input for our pore network model is an exact replica of the input used for the other studies. Our pore network properties are directly compared to the results obtained by the MB and MA algorithm.

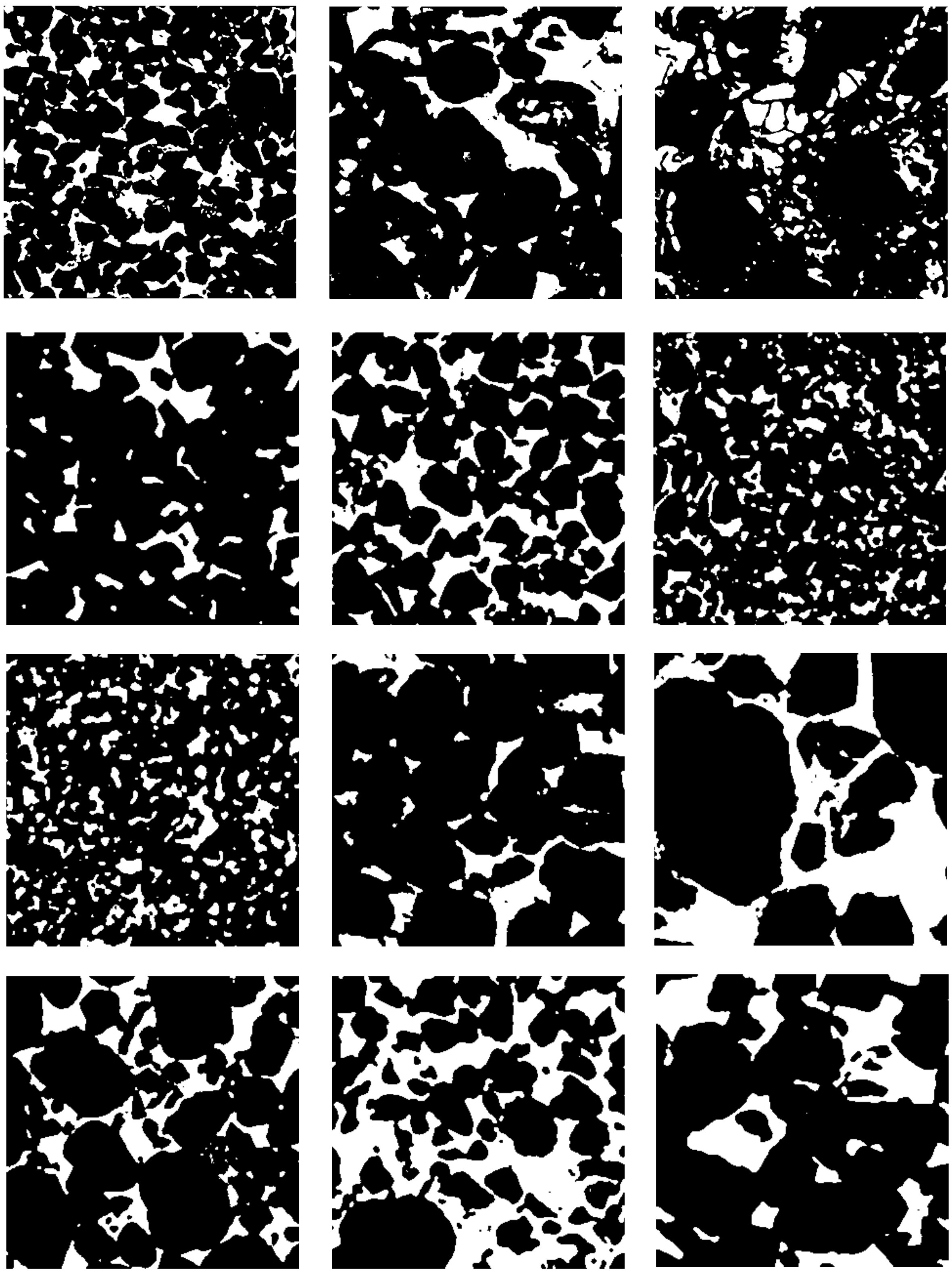


Figure 17. *Binary images of the 12 rock samples used in this research. From top left to bottom right in Latin script order: Berea sandstone, C1 – C2, S1 – S9.*

5.3 Fluid Flow Simulations

All fluid flow simulation for our pore network model were performed in PoreFlow (Raoof et al., 2013). The absolute permeability measurements are directly compared to the results obtained by using the maximal ball (MB) algorithm developed by Dong and Blunt (2009) and the medial axis (MA) algorithm developed by Lindquist and Venkatarangan (1999). These two algorithms are chosen since the code is freely available and are already calibrated for the samples used. Furthermore, the fundamentals of our models are based on these techniques as well, using the medial axis algorithm created by Lee et al. (1994) to extract the skeleton while applying to same merging principles introduced by Dong and Blunt (2009). As both MA and MB methods are involved in the creation of our PNM, it is now referred to as the COMBO model. The absolute permeability results are also compared to the Lattice Boltzmann simulations, which are both performed by Dong and Blunt (2009) and Yi et al. (2017).

5.4 Percolation Theory

The relative permeability in relation to the water content is computed using both PoreFlow and the percolation theory. Four different results are compared: i) the results of PoreFlow, ii) the results of universal scaling and CPA (equations (29), (30), (31)), and (iii and iv) two fitted curves with one different constraint. The surface area, volume and fractal dimensionality, which are required for method 2, are directly extracted from the binarized images in *Fiji*. The porosity is experimentally obtained, and the absolute permeability is set equal for all methods to the absolute permeability conducted by PoreFlow. The fitting procedure is performed using a nonlinear least square optimization between the results of PoreFlow and equations (25) and (26). The optimization is performed on four different parameters with 6 crucial constraints (*Table 1*).

The last two constraints are equal to equation (27) and (28). The single constraint which differs between method 3 and 4 is the minimum value of the critical water content. For method 3 this value is set to be larger than zero while for method 4 this value should be equal or larger than the last measured water content before the gradient of the saturation capillary pressure curve (dΘ/dh) becomes zero. This last constraint is added because, under normal conditions, the water content cannot drop below water contents indicated by the PCS-curve.

Table 1. *The constraints and parameter used for the nonlinear least square optimization.*

Fitting Parameters	Constraints
D	$2.0 \leq D < 3.0$
Θ_c	$\theta_c < \theta_x$
Θ_x	$\theta_c < \theta_x < \theta_s$
K_0	$K_0 > 0$
	$K_s \left[\frac{1-\varphi+\theta_x-\theta_c}{1-\theta_c}\right]^{\frac{D}{3-D}} = K_0(\theta_x-\theta_c)^2$
	$\frac{D}{3-D}\frac{K_s}{(1-\theta_c)^{\frac{D}{3-D}}}(1-\varphi+\theta_x-\theta_c)^{\frac{D}{3-D}-1} = 2K_0(\theta_x-\theta_c)$

6 Results

6.1 Pore Network Properties

The medial axis algorithm (MA), maximal ball algorithm (MB) and our algorithm (COMBO) are compared in terms of porosity (ϕ), amount of pore bodies (nPB), amount of pore throats (nPT), average coordination number ($\overline{CN}$), average pore body radius ($\overline{\boldsymbol{R_{PB}}}$), average pore throat radius ($\overline{\boldsymbol{R_{PT}}}$) and average pore throat length ($\overline{\boldsymbol{L_{PT}}}$) (Table 2). These parameters are chosen as they are the input for PoreFlow, thus directly influencing the flow parameters.

The network porosity of the COMBO model is most often equal or slightly smaller than the real porosity. Exceptions are C1, S1 and S6, for which the network porosities are slightly larger than the real porosity and another exception is S8, for which the network porosity is 4.6 percent lower than the real porosity. The network porosity of the COMBO model is calculated by PoreFlow and is depending on the pore body radius, throat radius, shape factor and throat length.

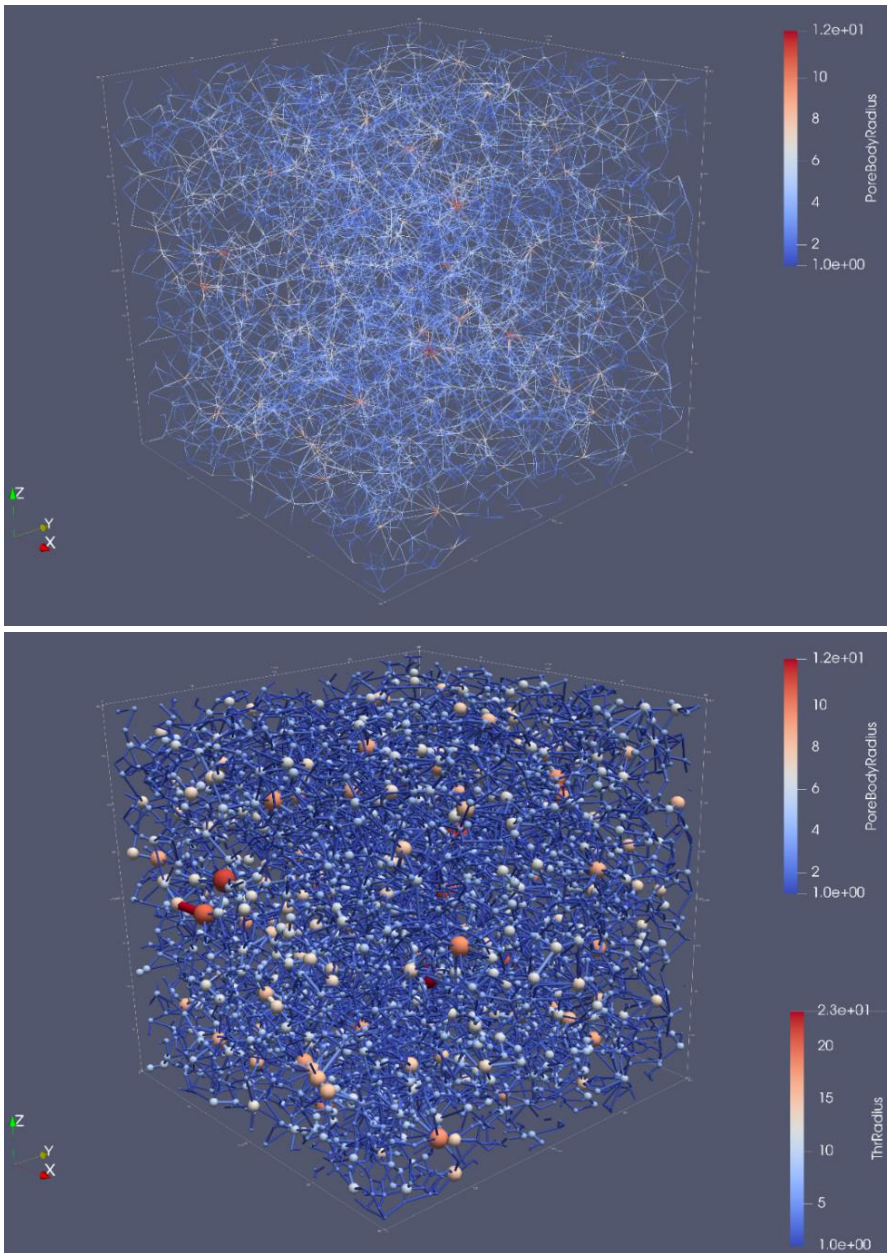


Figure 18. *Three-dimensional representation of the skeleton (top) and eventual pore network (bottom) of the Berea sandstone.*

Table 2. *Pore network properties for the 12 different samples generated by the medial axis (MA) algorithm, maximal ball (MB) algorithm and COMBO model.*

		Berea	**C1**	**C2**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**
Resolution (µm)		5.35	2.85	5.35	8.68	4.96	9.10	8.96	4.00	5.10	4.80	4.89	3.40
Size (voxels)		400^3	400^3	400^3	300^3	300^3	300^3	300^3	300^3	300^3	300^3	300^3	300^3
ϕ		19.6	23.3	16.8	14.1	24.6	16.9	17.1	21.1	24.0	25.0	34.0	22.2
	MA	19.5	21.3	14.1	14.1	24.6	16.6	16.5	21.1	24.0	25.0	34.0	22.1
	MB	19.6	23.2	16.8	14.1	24.6	16.8	17.1	21.1	24.0	25.0	34.0	22.2
	COMBO	18.4	26.9	14.7	14.2	22.5	16.4	15.7	19.8	25.2	23.0	29.4	21.9
nPB	*MA*	7611	4115	5590	1452	2906	9104	7199	659	1022	1759	2439	463
	MB	6298	4576	8508	1868	2021	8926	9556	518	597	1016	1324	604
	COMBO	11930	10146	15008	2493	4026	13763	12761	1278	1686	2407	3766	838
nPT	*MA*	14328	8893	11564	2455	5792	15833	11508	1243	2147	3448	5235	865
	MB	12545	6921	10336	3048	4942	15105	13322	900	1234	2741	4209	1054
	COMBO	16632	12070	15664	3176	6415	18621	14883	1535	2407	3818	5574	1137
$\overline{CN}$	*MA*	3.80	4.32	4.14	3.38	3.99	3.48	3.20	3.77	4.20	3.92	4.29	3.74
	MB	3.98	3.02	2.43	3.26	4.89	3.38	2.79	3.47	4.13	5.40	6.36	3.49
	COMBO	2.79	2.38	2.09	2.55	3.19	2.71	2.33	2.40	2.86	3.17	2.96	2.71
$\overline{R_{PB}}$ **(µm)**	*MA*	16.79	8.66	14.62	32.59	18.05	19.92	20.90	17.84	18.79	19.01	19.82	22.03
	MB	15.36	7.05	11.39	25.59	17.25	16.69	16.80	16.71	19.00	20.15	21.16	16.20
	COMBO	25.26	12.86	18.48	50.71	27.02	32.54	32.39	25.75	31.98	28.69	23.88	30.59
$\overline{R_{PT}}$ **(µm)**	*MA*	9.09	4.51	7.39	16.50	9.79	10.22	10.60	9.36	9.84	10.40	11.16	10.99
	MB	7.15	4.02	6.17	12.32	8.13	7.51	7.81	9.49	10.10	9.30	10.47	8.73
	COMBO	13.28	8.98	13.65	22.45	13.95	15.09	15.48	16.45	19.91	16.94	20.05	14.33
$\overline{L_{PT}}$ **(µm)**	*MA*	99	82.8	100.1	204.2	103.5	117	121	122.5	144.6	115.5	123	132.3
	MB	143.7	88.4	122.8	232.6	154.9	161.5	151.1	187	229.2	187.3	194.4	151.2
	COMBO	11.7	7.8	12.1	20.2	11.8	14.5	14.6	12.2	15.5	14.1	17.3	10.8

The COMBO model both generates more pore bodies and pore throats than the other two algorithms. It is important to note that more pore bodies are automatically linked to more pore throats. S9 has the least amount of pore bodies and pore throats, while the Berea sandstone has the most pore bodies and C2 the most pore throats. The average coordination number and throat length is lower for the COMBO model for every single sample while the pore body radii and pore throat radii are always larger. The difference in throat length between the COMBO model and the other two models is remarkable, as the difference is often a factor 10.

The average shape factor, average inscribed pore body radius and average inscribed pore throat radius are shown in Table 3. These pore network properties are represented separately as the data regarding the shape factor was not available for the MA and MB models, furthermore were the inscribed radii not used in the flow computations for the COMBO model. The pore body radii and pore throat radii in

Table 3. *Pore network properties of the 12 different samples generated our model (COMBO).*

	Berea	**C1**	**C2**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**
Avg. SF	0.032	0.027	0.030	0.033	0.029	0.037	0.037	0.025	0.024	0.025	0.021	0.031
Avg. R_{PBIS}	13.81	6.30	11.06	25.84	14.85	17.71	18.17	12.80	14.87	15.80	14.73	15.84
Avg. R_{PTIS}	10.23	5.97	9.90	17.65	10.52	13.27	13.50	10.96	12.80	11.48	12.10	10.96

Table 4. *Properties related to the merging algorithm applied in the COMBO model for the12 different samples. The dead-ends only represent the number of dead-end pores which are removed.*

	Berea	C1	C2	S1	S2	S3	S4	S5	S6	S7	S8	S9
Masters	8295	4662	7802	1416	2653	8379	7023	501	987	1469	1955	566
Slaves	8820	11313	11826	1844	3889	5906	5771	2075	2613	2935	5501	1200
Dead-ends	3635	5484	7206	1077	1373	5384	5738	777	789	938	1811	272
Multi Throats	1012	2942	3069	99	403	683	574	263	542	327	612	82

The values in Table 3 are much more similar to the radii used in the MA and MB algorithm, as these models also make use of the maximal inscribed sphere.

The amount of masters, slaves, dead-ends and throats connecting the same pair of pore bodies for each sample is shown in Table 4. The amount of masters is exactly equal to the total amount of pore bodies in Table 2 minus the number of dead-ends, as all slaves and dead-ends are removed. S3 and S4 are the only samples with a larger amount of masters than slaves, however this is compensated by the large number of dead-ends which are removed. The carbonate samples have the largest number of slaves, accordingly 111313 for C1 and 11826 for C2. The number of dead-ends and multi throats are also exceptionally high for these samples. From this data can be concluded that the initial amount of pore bodies, before any merger or removal, was the highest for the carbonate samples, as the masters, slaves and dead-ends together add up to the initial amount of pore bodies.

The frequency diagram of the shape factor is plotted for the Berea sandstone (Figure 20). The overall distribution is normal, with a peculiar peak at 0.061, representing a rectangle. This phenomenon is visible for all the shape factor frequency diagrams. The cumulative frequency of the throat radius for sandstone S3 is plotted in Figure 19. The radius of the maximum inscribed sphere and the inradius are both presented. The inscribed sphere radius has a stair-like cumulative frequency while the cumulative frequency of the inradius is much smoother. Another outstanding difference is the starting point of both curves, which for the inscribed sphere radius starts at 9.10 µm, which is equal to the resolution, and 4.55 µm for the inradius. The stair-like cumulative frequency and starting point equal to the resolution is observed for all samples but is most prominent for S3 and S4. The cumulative frequency of throats with an inscribed sphere radius smaller or equal to 3 voxels is for sandstone S3 97% while for the Berea sandstone this percentage is reduced to 87%. The vast majority of pore throats is smaller than three voxels when using the inscribed sphere and thus ambiguous, as voxels are discrete units.

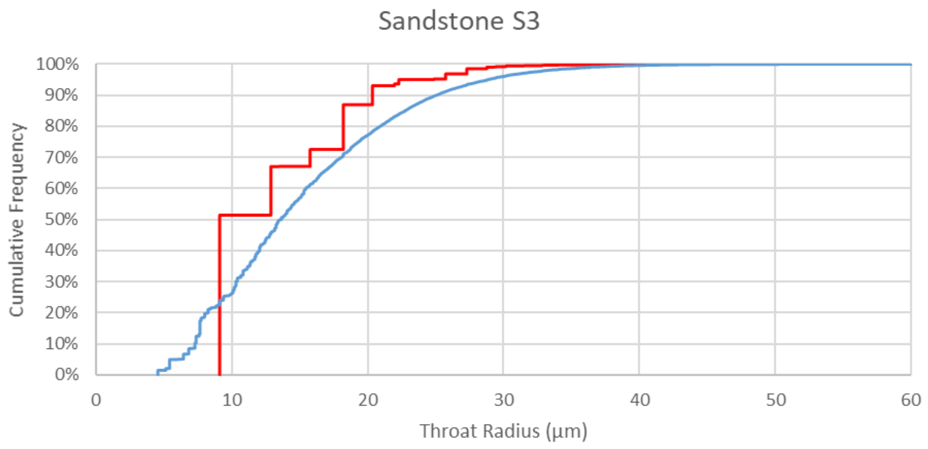


Figure 19. *Cumulative frequency diagram of the throat radius for sandstone S3. The red line represents the maximum inscribes sphere radius and the blue line represent the inradius.*

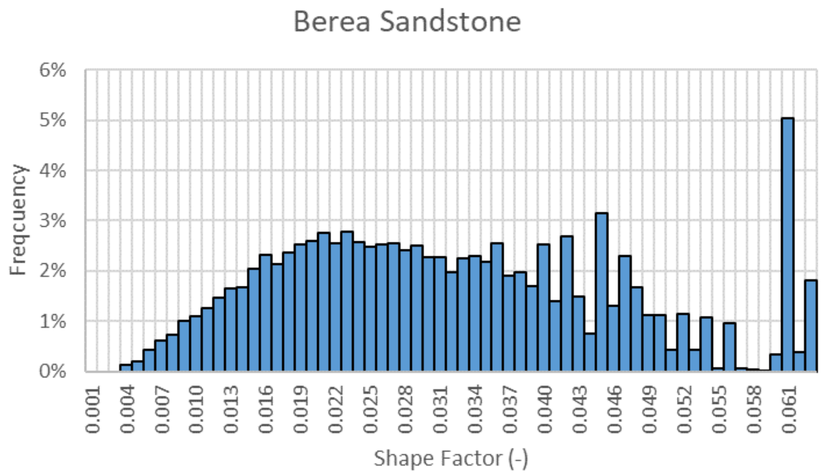


Figure 20. *Frequency diagram of the shape factor for the Berea sandstone with a bin of 0.001.*

This phenomenon can also be observed when looking at the average throat radius in Table 3, which is for almost all samples smaller than three times the resolution, except for S9. When looking at the average throat radius used by the MA and MB algorithm, S3 and S4 have values similar to the resolution. The cumulative frequency distribution for the pore throat radius in voxels is displayed in Figure 24. All samples have similar distributions, slightly varying in width. Again, S3 and S4 have a much narrower pore size distribution.

6.2 Absolute Permeability

The absolute permeability is computed in the x, y and z direction and then averaged for each sample. The results can be observed in Table 5, together with the absolute permeability values obtained by the MA algorithm, MB algorithm and the lattice Boltzmann simulation performed by Yi et al. (2009) (LBY) and Dong and Blunt (2009) (LBDB). The results obtained by the MA algorithm is most similar to the LBY simulation while the MB algorithm is most similar to the LBDB simulations. This is caused by using two different variations of the Lattice Boltzmann algorithm yielding completely different results, sometimes varying over a factor four (S3). The absolute permeability computed by the COMBO model will thus only be compared to the MA and MB algorithms. This precaution is taken due to the fact that the lattice Boltzmann simulations do not give conclusive results and because it is more rationale to compare the same type of models (PNMs).

Table 5. *Absolute permeability values obtained for each sample by different methods.*

	K_x	K_y	K_z	K_{avg}	K_{MA}	K_{MB}	K_{LBY}	K_{LBDB}
Berea	1796	1770	1585	1717	1790	1111	1777	1286
C1	1093	2216	871	1393	2009	556	1647	1102
C2	200	339	215	251	263	158	206	72
S1	2792	3078	2107	2659	3020	1486	2410	1678
S2	3957	3659	3073	3563	4685	3950	4779	3898
S3	1507	1923	1233	1554	1325	281	897	224
S4	1041	1165	721	976	823	169	528	259
S5	3790	5201	3932	4308	6384	5369	6386	4651
S6	11813	10380	7636	9943	15635	11282	15769	10974
S7	5151	6505	4666	5441	7515	7926	9068	6966
S8	9011	9591	7992	8865	15092	13932	15743	13169
S9	3035	4070	2282	3129	3267	3640	2790	2224

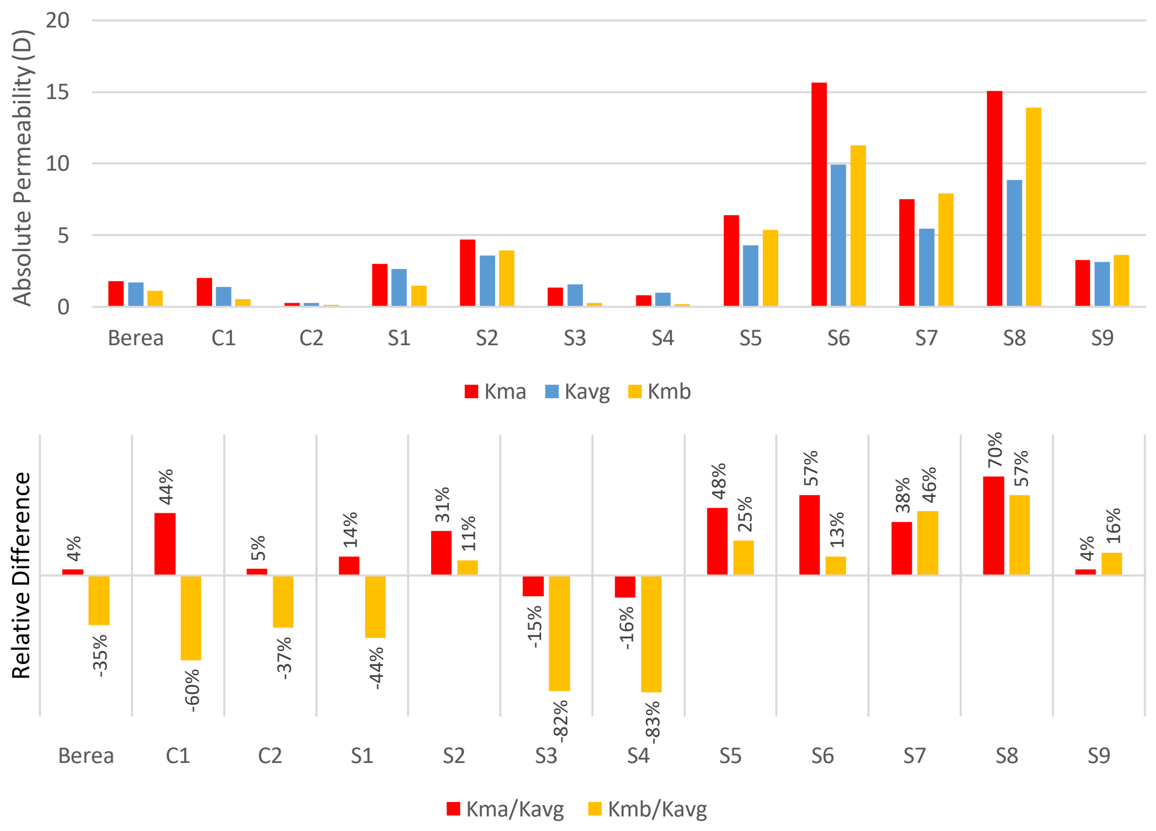


Figure 21. *Absolute permeability measurements (top) and the relative difference between these measurements (bottom) for the different algorithms.*

The difference in absolute permeability for the different flow directions is most noticeable for C1, C2 and S9, which have a relative difference between the minimum and maximum value of over 70%. On the other hand, the Berea and S5 sandstone are often consistent in all directions, with a variation of less than 20%.

Figure 21 displays the absolute permeabilities and the relative difference of the different samples for the different algorithms. The results generated by the COMBO model sit in between the values computed by the MA and MB algorithm for the Berea sandstone, C1, C2 and S1. The COMBO model generates higher results for S3 and S4, while it computes lower values for all the other samples. In general, the MA algorithm tends to estimate a higher absolute permeability than the MB algorithm, except for S7 and S9. Peculiar is the fact that the COMBO model estimates a considerably lower absolute permeability for S7 and S8 compared to both algorithms.

6.3 Relative Permeability and Percolation Theory

The relative permeabilities obtained by PoreFlow are compared to three different relative permeability curves generated using CPA (25) and universal scaling (26). There are six key parameters governing the shape of this curve: the porosity (ϕ), the saturated permeability (K_s), the fractal dimensionality (D), the critical water content (θ_c), the cross-over water content (θ_x) and the calibration constant (K_0). The first two parameters are rock properties and are therefore similar for each curve. The other four parameters can vary for each curve, having unique effects on the shape. The unique parameter combinations for each sample can be observed in Table 6, the three different curves are labelled as PERC, FIT1 and FIT2. The values regarding PERC are calculated using the conventional percolation formulas while FIT1 and FIT2 are computed using a nonlinear least square optimization varying in critical water content. There are some remarkable observations when comparing the

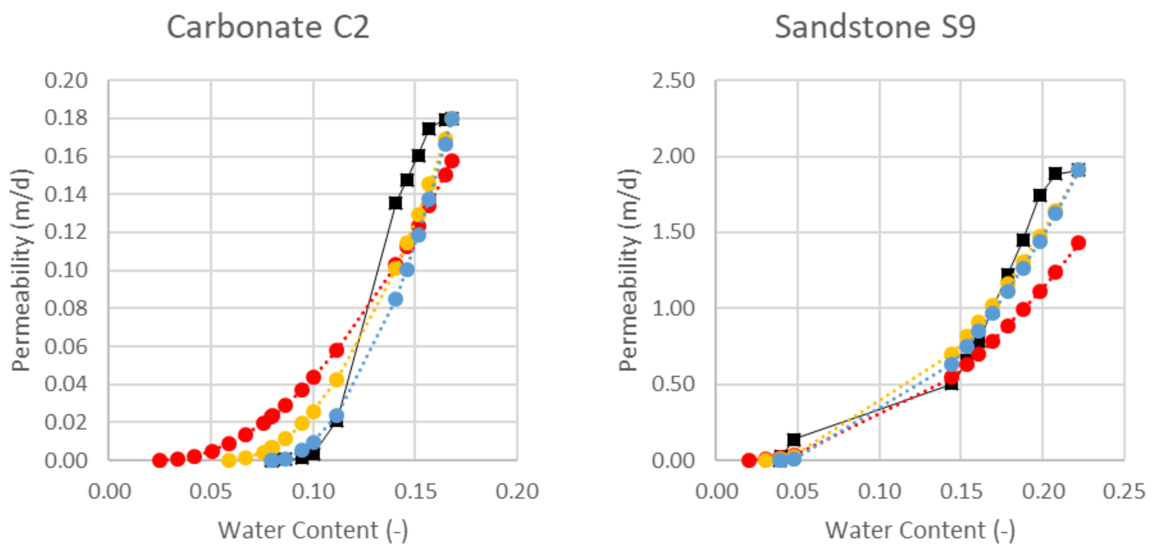


Figure 22. *The oddly shaped relative permeability curves for the carbonate C2 and sandstone S9. The black squares represent the results of PoreFlow, the red dashed line represent PERC, the yellow dashed line represents FIT1 and the blue dashed line represents FIT2.*

different parameter sets. First of all, the cross-over water content regarding PERC is for half of the samples (C2, S1, S5 – S7, S9) higher than the fully saturated conditions. Secondly, the critical water content and calibration constant is for FIT2 always higher than for PERC and FIT1. Lastly, when comparing the fractal dimensionality, it is notable that PERC has noticeable lower value for S1, S5 and S6. The fractal dimensionality should, in combination with the porosity, coincide with the cumulative frequency distribution of the pore throats (Figure 24). Unfortunately, the cumulative distribution is not exactly log-normal, making the method introduced by Hunt and Gee (2002) invalid.

The relative permeability curves can be observed in appendix A and B, in which the former is displayed on a normal scale and the latter is displayed on a log scale. Both scales are incorporated to accentuate the differences close to zero. The influence of each parameter on the shape of the curve can easily be observed but can also be conducted from the formulas. Universal scaling is only affected by the

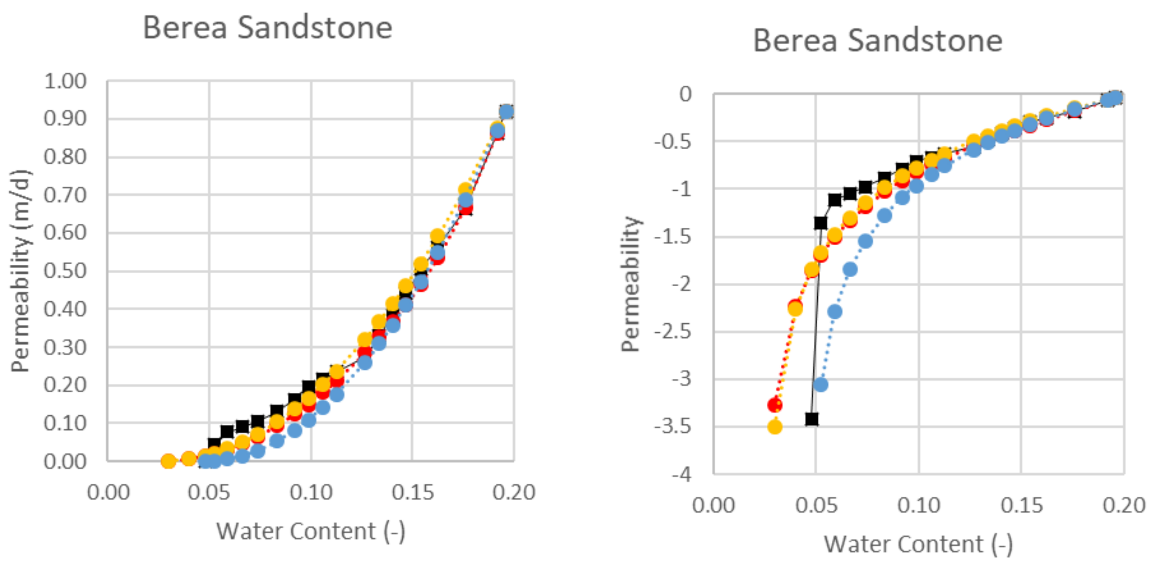


Figure 23. *The relative permeability curves for the Berea sandstone. The black squares represent the results of PoreFlow, the red dashed line represent PERC, the yellow dashed line represents FIT1 and the blue dashed line represents FIT2.*

Table 6. *Percolation parameters influencing the relative permeability curve.*

		Berea	C1	C2	S1	S2	S3	S4	S5	S6	S7	S8	S9
ϕ		0.196	0.233	0.168	0.141	0.246	0.169	0.171	0.211	0.240	0.250	0.340	0.222
K_s (m/d)		0.921	0.729	0.180	1.763	2.571	1.031	0.603	3.289	6.387	3.903	6.685	1.909
A/V (m^{-1})		0.446	0.314	0.431	0.435	0.398	0.693	0.685	0.240	0.231	0.312	0.316	0.237
D	*PERC*	2.819	2.713	2.717	2.614	2.780	2.849	2.883	2.512	2.525	2.681	2.801	2.560
	FIT1	2.760	2.756	2.835	2.885	2.674	2.846	2.878	2.739	2.781	2.752	2.645	2.725
	FIT2	2.784	2.782	2.863	2.889	2.734	2.838	2.880	2.803	2.768	2.746	2.729	2.739
Θ_c	*PERC*	0.026	0.021	0.025	0.025	0.024	0.032	0.032	0.019	0.018	0.021	0.021	0.018
	FIT1	0.027	0.068	0.059	0.036	0.003	0.026	0.045	0.025	0.017	0.002	0.005	0.025
	FIT2	0.048	0.090	0.080	0.037	0.057	0.040	0.053	0.046	0.062	0.070	0.098	0.039
Θ_x	*PERC*	0.144	0.227	0.244	0.385	0.166	0.131	0.105	0.519	0.477	0.256	0.131	0.426
	FIT1	0.196	0.233	0.168	0.110	0.246	0.127	0.121	0.211	0.160	0.166	0.247	0.222
	FIT2	0.196	0.233	0.168	0.109	0.239	0.147	0.128	0.174	0.215	0.240	0.261	0.222
K_0 (m/d)	*PERC*	28.0	16.3	7.7	61.8	43.5	49.7	19.9	53.6	95.7	74.6	19.0	34.5
	FIT1	32.2	26.8	15.1	141.4	43.4	44.5	29.1	95.1	106.9	54.4	55.0	49.2
	FIT2	42.0	35.9	23.2	141.4	71.7	60.0	35.0	113.7	197.1	119.7	100.0	56.8

calibration constant and critical water content while CPA is influenced by the porosity, absolute permeability, fractal dimensionality and, also, the critical water content. It is important to note that the dedicated parameters only influence their respective part of the curve. The cross-over water content acts as the transition point between universal scaling and CPA, the former is valid from the critical water content until the cross-over water content while CPA is valid from this transition point until the porosity. For universal scaling, a higher calibration constant would increase the cross-over permeability and would thus increase the slope. A higher critical water volume would also increase the slope, but this due to the fact that the water content range decreases. The shape of the curve is not directly altered as the power is always equal to two. For CPA, this is a bit more complicated due to a different exponent. As the fractal dimensionality increases, the slope increases and the shape changes slightly. The slope also steepens for a higher absolute permeability, increasing the vertical

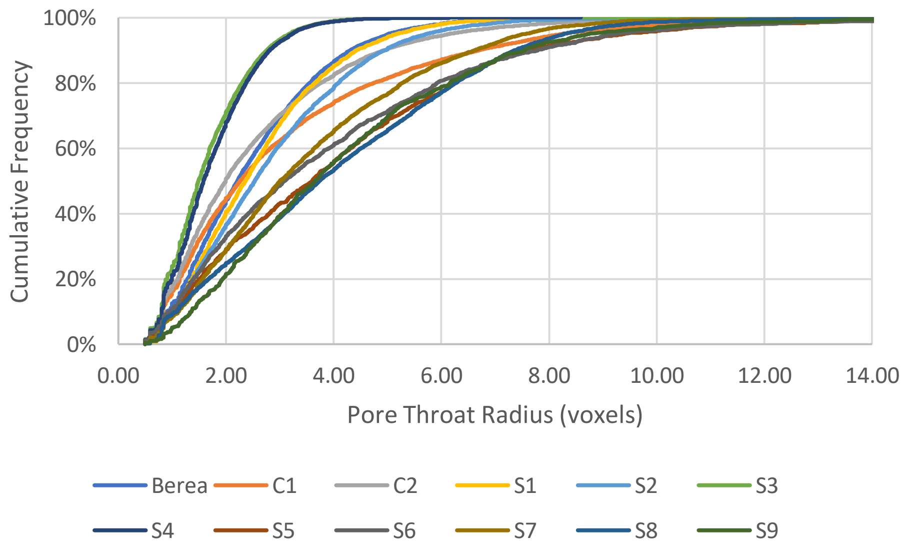


Figure 24. *Cumulative frequency distribution of the pore throat radius in voxels. S3 and S4 have clearly the narrowest pore size distribution.*

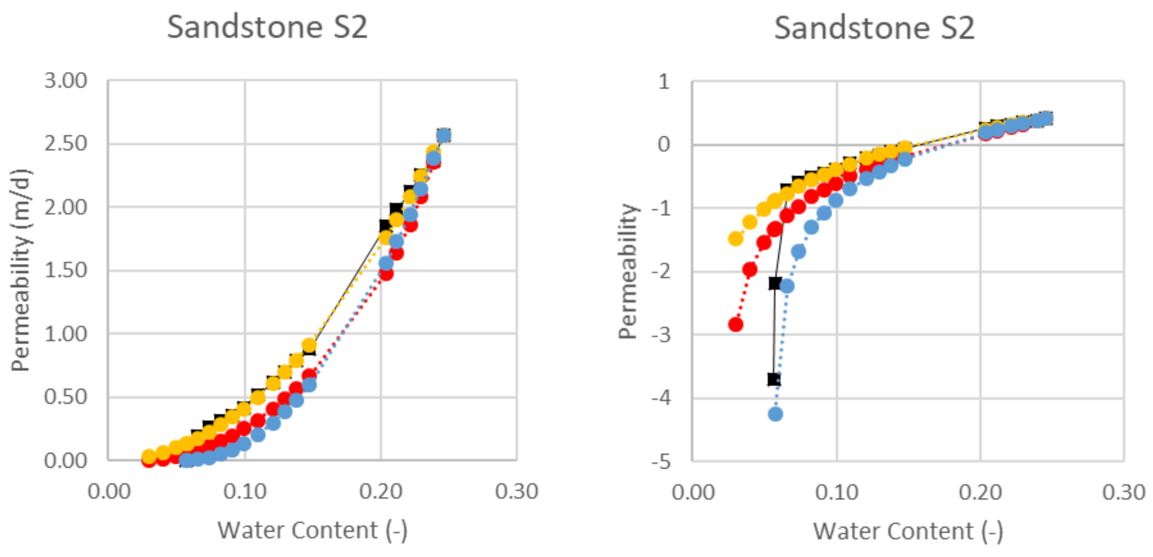


Figure 25. *The relative permeability curves for the sandstone S2. There is a sudden drop in permeability when the water content approaches 0.07. The black squares represent the results of PoreFlow, the red dashed line represent PERC, the yellow dashed line represents FIT1 and the blue dashed line represents FIT2. Both PERC and FIT2 underestimate the permeability.*

range of the curve. Decreasing the porosity, responsible for the upper limit of the water content, would increase the slope as the horizontal range decreases. Although the critical water content contributes to both universal scaling and CPA, it has a larger impact on universal scaling. This is due to the fact that the overall water content is lower and thus the proportion of the critical water content is larger. Before any comparisons are made between the curves generated by PoreFlow and the results computed using percolation theory, it is important to note that C2 and S9 are not representative, as the relative permeability curves have an unnatural shape (Figure 22). A normal relative permeability curve increases exponentially as the saturation increases, however, the relative permeability curves of these samples start to flatten as they approach full saturation. The results of PoreFlow are used as a benchmark, both for comparison and calibration, but the results will never match if the benchmark is not accurate.

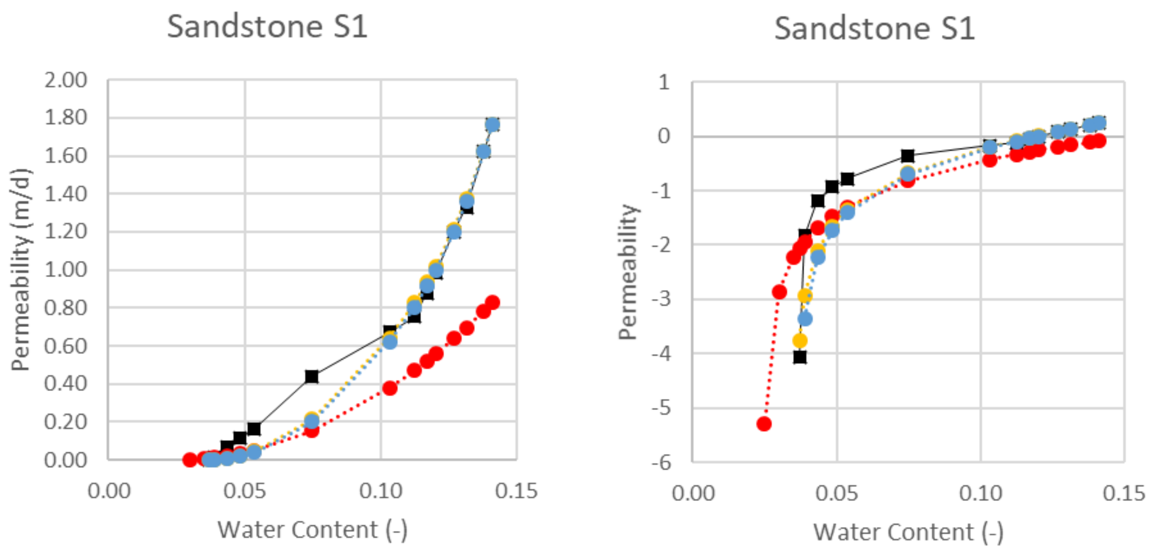


Figure 26. *The relative permeability curves for the sandstone S1. There are many saturation points measured at higher water contents, but fewer close to the critical water content. The black squares represent the results of PoreFlow, the red dashed line represent PERC, the yellow dashed line represents FIT1 and the blue dashed line represents FIT2. PERC heavily underestimates the relative permeability, also at higher saturations.*

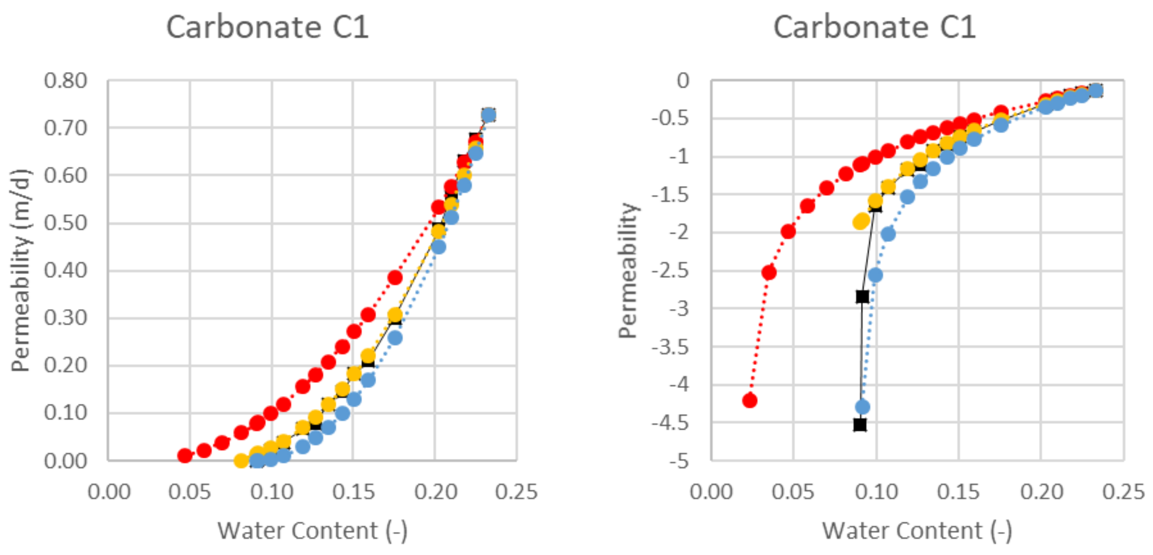


Figure 27. *The relative permeability curves for the carbonate C1. The black squares represent the results of PoreFlow, the red dashed line represent PERC, the yellow dashed line represents FIT1 and the blue dashed line represents FIT2. PERC heavily overestimates the permeability while FIT2 generates the most accurate results for this sample.*

Unfortunately, percolation theory (PERC) is not able to give consistent results for each sample. There are two main errors which can be observed: i) an absolute difference between the permeability values for a certain saturation and ii) a difference between the saturation values for which the permeability approaches zero. The first error is smaller for higher saturations but increases with decreasing saturation. The second error is always an underestimation of the critical water content. Ultimately, percolation theory is able to predict the relative permeability reasonably well for the Berea sandstone (Figure 23), S3, S4 and S7. For samples S1 (Figure 26), S2 (Figure 25), S5, S6 and S8 it underestimates the permeability while for sample C1 (Figure 27) it overestimates. The underestimation of S1, S5 and S6 is most noticeable as the permeability at full saturation is much lower than the saturated permeability. The overestimation of C1 is also very peculiar as this does not happen for any other sample or any other fitting technique.

FIT1 tends to estimate the best results for higher saturations, it fails however, whenever the permeability drops suddenly (Figure 25). Overall, FIT1 performs considerably better than the results generated by percolation theory. It gives accurate results for the Berea sandstone, C1, S2 – S4 and S6 – S8. The estimation for S1 and S5 are less successful, these samples tend to have a lot of saturation points and thus calibration points at higher saturations but this decreases with decreasing saturation. Ultimately, FIT1 is superior when compensating for error 1.
FIT2 generates the best results at saturations close to the critical water content and often generates better results than percolation theory for higher saturations. It is the only method which approaches zero at the same water content as PoreFlow and is thus superior when compensating for error 2. This advantage brings the drawback of heavily underestimating the permeability when it starts to drop from full saturation, which results in many samples being underestimated by FIT2. The best results using this method are obtained for samples C1 (Figure 27), S3 and S4, the relative permeability of these samples also decreases more gradual (overall steeper slope). All the other samples are being underestimated.

7 Discussion

7.1 Pore Network Properties

Most differences in pore network properties regarding the topology are caused by the merging and removal procedures, while most geometry differences are caused by conscious physical improvements made when designing the COMBO model. The amount of pore bodies, amount of pore throats and the average coordination number are typical examples relating to the prior. Initially, the amount of pore bodies in the COMBO model is based on the number of junctions and dead-ends present in the skeleton, which is the same definition used in the MA algorithm. However, the MA algorithm reduces this number by cutting of dead-end throats with an ambiguous length. The COMBO model makes use of a sophisticated merging algorithm, also reducing the amount of pore bodies, but on a more physical basis. The MB algorithm has initially less pore bodies, as only the voxels with the largest inscribed sphere are appended. Besides, the MB algorithm utilises an ambiguous filter as well, to prevent a repetition of pore body voxels and pore throat voxels within a throat, drastically decreasing the amount of pore bodies.

An increase of pore bodies will be accompanied by an increase in pore throats, provided that the average coordination number does not change. However, the average coordination number decreases for the COMBO model. This causes the amount of pore throats to be slightly larger compared to the different algorithms. The reduction in average coordination number has to do with our merging and removal algorithm. When a throat is removed or a cluster is merged into a single pore body, the pore body will be conserved, even though its coordination number is two. This drastically decreases the average coordination number as the number of slaves and dead-ends is a considerable portion of the total amount of pore bodies. Besides, the majority of clusters consists of two or three pore bodies which are merged into one pore body. The pore bodies with a coordination number of two are kept in order to get a better representation of the pore space. Splitting up one throat into two parts gives more detail about the geometry and topology of the media. Furthermore, the pore bodies with a coordination number of zero are also preserved, which is done to conserve the porosity.

The average pore body radius, average pore throat radius and average pore throat length are influenced by the alterations made in the COMBO model to measure these entities. First of all, the pore throat radius, which is always higher for the COMBO model due to the use of the inradius. The underestimation of larger pores by the inscribed sphere is clearly explained in Figure 14. Unfortunately, there may also be some overestimation of the COMBO model since only nine scanning planes (Figure 12) are used, which limits the accuracy for larger pores. There may indeed be a perturbation in the pore surface which could cause the bottleneck to be smaller and thus throat radius to be lower. However, most often this is a local effect not influencing the flow and is expected to be negligible. The pore body radius is much larger due to the use of the volume equivalent radius instead of the maximum inscribed sphere. This is much more realistic as the pore bodies only contribute as main liquid storages, affecting also solute transport and mixing. The average pore throat length is much shorter due to i) the increase in pore throats and ii) because of the use of the aspect ratio. The total amount of throat length is not much different than the MA model, however, it is split up in more segments reducing the average throat length. The aspect ratio was introduced because there is no clear boundary between pore bodies and pore throats. Using Hagen-Poiseuille law and the aspect ratio was a liable solution to give a physical meaning to the throat length.

A reduction in network porosity is common, as pore bodies which are not attached to the main connection spanning between the inlet and outlet pores are often not considered. An increase in network porosity is however, very uncommon. This phenomenon is caused by pore throats for which its volume is not correctly divided. The volume of a pore body is calculated using the watershed algorithm after which half the volume of each attached throat is deducted. As the watershed algorithm in the COMBO model makes use of inscribed spheres as markers instead of points, it divides the pore space according to inscribed pore body radius. However, pore bodies with a relatively small inscribed sphere but a high coordination number will reduce drastically in size. This is due to two reasons, firstly the watershed algorithm underestimates the amount of volume which should be assigned to the pore body because of the small inscribed sphere. Secondly, volume decreases due to the large amount of connecting throats. Whenever a pore body volume becomes negative, the pore body radius is set to a minimum number (here equal to 1 voxel. This phenomenon causes volume to be generated, causing the network porosity to increase.

The large number of slaves, dead-ends and multi throats regarding the carbonates confirms the heterogeneity of the samples. Defining the difference between pore bodies and pore throats is even harder for these samples, which may cause difficulties when estimating the flow properties. Generating reliable results will even become more complex when incorporating relative permeability, multiphase flow and solute transport. The pore space needs to be mimicked in great detail to approximate these complex processes, but it is hard to include enough detail with a limited amount of pore bodies and pore throats. Due to the fact that the COMBO model handles more input parameters than the other algorithms, may results in achieving more accurate flow properties.

S3 and S4 clearly deal with resolution issues, which can be concluded from the average pore throat radius which is close to the resolution magnitude and by looking at the cumulative distribution of the pore throat radius, which has a stair-like shape. S3 and S4 also deal with a large amount of removed dead-end pores, as the pore body radius is often equal to the pore throat radius. The use of the inradius has decreased the amount of removed dead-ends, as the throat radius can become smaller than 1 voxel. Overall, there are five main advantages of using the inradius: i) the cumulative distribution is more realistically shaped as the radius is not limited by the discrete voxel length, ii) smaller throats will not be overestimated by the inscribed sphere, iii) larger pores will not be underestimated by the inscribed sphere, iv) skeletons which are not exactly in the middle of the pore space are no longer a problem when determining the geometrical features and v) a reduction in the amount of unnecessarily removed dead-end pores. These five improvements make the COMBO model more robust and better for dealing with resolution issues.

The peculiar peak in the frequency diagram of the shape factor is caused by the size of the bins. There are two common values, 0.0606601698888 and 0.0606601704427, which contribute to the same bin but cannot be linked to standard area and perimeter.

7.2 Absolute Permeability

The relatively high differences in absolute permeability for the different flow directions was expected for the carbonate samples, which are heterogeneous and anisotropic. It was not expected for a sandstone, which in general is more homogeneous. A cause for this difference in absolute permeability for S9 may be a preferential flow path, which cannot be excluded when looking at the binary images. However, all data presented in this paper is not coupled to an orientation, which makes it impossible to give an undeniable conclusion.

The properties of the COMBO model should explain why the absolute permeability is underestimated or overestimated compared to the MA and MB algorithms. Unfortunately, the result obtained are ambiguous and the input is intertwined with each other. This makes it impossible to verify whether the COMBO model is superior than the other algorithms and thus impossible to pin point what is the cause. The (absolute) permeability is governed by four parameters: the throat radius, the throat length, the shape factor and the connectivity of the throats. Larger throat radii, shorter throats, higher shape factors and a higher connectivity will all increase the (absolute) permeability, and vice versa will decrease it. These observations are generally accepted, but the trouble starts when comparing different models.

Initially, the connectivity is not altered by the COMBO model, as this is one of the key features of the skeletonization algorithm developed by Lee et al. (1994). The connectivity is dependent on the total amount of isolated objects, the number of redundant connections or loops and the completely enclosed cavities. In the COMBO model the connectivity is only reduced by removing some redundant connections due to multi throats. It is true that some completely enclosed cavities are removed as wel, but this is compensated by the fact that these will also be removed from the total amount of isolated objects. Hence, the connectivity is mostly altered for the carbonate samples, which have the largest amount of multi throats. The significance of this reduction in relation to the absolute permeability is unknown and should be studied in more detail.

Also, little is known about the significance of the geometrical parameters influencing the absolute permeability. From Poiseuille law in a cylindrical tube we can conclude that the radius is a power three more significant than the throat length, however, no conclusive relations exist for the shape factor. It is generally accepted that the shape and the value of the shape factor affect multiphase flow, but the influence on absolute permeability is often neglected. However, an obtuse triangle has a significant higher perimeter than a circle, consequently causing the friction to increase and the absolute permeability to drop.

The observations mentioned above make it impossible to label the capability of the COMBO model. Furthermore, the fact that the COMBO model has an entirely different topology than the other algorithms makes it even harder to compare the influence of the geometry on the absolute permeability. Lastly, the lack of a benchmark data makes it impossible to calibrate the COMBO model. There are several Lattice Boltzmann simulations and different pore network models, all generating different results. Before developing the COMBO model further, an in-depth sensitivity analysis should be done on the different input parameters. Also, an objective and generally accepted data set should be created.

Currently, the COMBO model is not a superior model, which was the goal when this entire research project started. However, the improvements made on the MA and MB algorithms are physically more feasible. Whether the COMBO model also produces a better pore network model for flow simulation is for now unknown.

7.3 Relative Permeability and Percolation Theory

The peculiar phenomenon of cross-over water contents being higher than the porosity results in percolation curves which are only computed using universal scaling and thus only influenced by the critical water content and calibration constant. Percolation theory heavily underestimates the relative permeability for these samples as a consequence. S7 is an exception to this statement, which is caused by the fact that the cross-over water content is only slightly higher than the porosity, which implies that universal scaling is indeed enough to predict the relative permeability. This exception is however a coincidence, as CPA should always give better estimations close to fully saturated conditions (Ghanbarian and Hunt, 2012). The reason why the critical water content and calibration constant is always higher for FIT2 is caused by the applied constraint. As PoreFlow indicates that the permeability approaches zero, it is feasible to apply this restriction to percolation theory as well. Whether this constraint is an improvement on the results, will be discussed later. The relatively low fractal dimensionality of S1, S5 and S6 implies a narrow pore size distribution for the pore space, however, this cannot be observed when looking at the cumulative frequencies of the pore throat radii. Another explanation could be that the entire pore space is used to determine the fractal dimensionality, instead of solely the throat volume. This would suggest that the radii of the pore bodies and pore throats are relatively more similar in size. However, this is denied by the pore network properties regarding the average pore body radius and average pore throat radius for the sample, which do not have a conspicuous larger difference than the other samples. The flaw is most likely caused by the box searching algorithm, which is markedly influenced by the input parameters (Fazzalari and Parkinson, 1996). Lastly, there is also a possibility of the samples not being fractal at all.

Most parameters influencing the relative permeability curves are straightforward to describe. Increasing the absolute permeability and keeping all other parameters constant will result in more water traveling through the same volume of pore space, increasing the slope of the relative permeability. The same is true for the calibration constant, which is basically the cross-over permeability and thus the upper limit of universal scaling. The absolute permeability and calibration constant are also dependent on each other, changing proportionally by a factor (30). Similar as the absolute permeability, would the slope of the relative permeability increase when the porosity is lowered, as the same amount of water needs to travel through a smaller pore space. This can also be applied to the critical water content, which decreases the number of pores contributing to a spanning cluster between the input and output pores when increased. Again, the same amount of water needs to travel through a smaller pore space, increasing the slope of the relative permeability. An increase in fractal dimensionality and thus a broader pore size distribution, also increases the slope op the relative permeability. Larger pores have a relatively small total volume, as there are relatively few of them, but contribute significantly to the permeability. Increasing the pore size distribution will incorporate these larger pores for percolation theory as well. This will result in a drastic decrease in permeability as these larger pores will drain first, while having a relatively small impact on the saturation.

The biggest problem when estimating relative permeability using percolation theory (PERC) is related to the cross-over saturation. Wrongly estimating this value can cause the relative permeability to be heavily underestimated. The cross-over saturation is dependent on three different parameters, the porosity, the critical water content and the fractal dimensionality. The porosity and critical water content are both not the cause for failure, as the porosity is correct, and the critical water volume is relatively small compared to the cross-over water content. However, an overestimation of the cross-over water content is caused by an underestimation of the fractal dimensionality and vice versa. The prior statement is true for S1, S5 and S6, while the latter statement is true for S2 and S8. Remarkable is the fact that these five samples all underestimate the relative permeability, but for opposing reasons, emphasising the importance of correctly predicting this parameter. The overestimation of the relative permeability related to C1 is caused by an underestimated critical water content. This parameter is dependent on the ratio between the pore surface area and pore volume, which is remarkably low for carbonate C1. Heterogenous samples are more irregularly shaped and an increased ratio between the area and volume would be expected, however this is not the case. There are three possible explanation causing this error: i) trapping is not correctly incorporated, ii) the empirical formula provided by Moldrup (2001) is not applicable or iii) there is a flaw in de marching algorithm by Lorensen and Cline (1997), not correctly providing the surface area. The first explanation is most likely the cause, as trapping is only somewhat accounted for by not taking into account the smallest pores when applying percolation theory. However, trapping is also applicable for larger pores. The significance of this error should be studied in further detail. Moreover, there is the general problem of underestimating the critical water content, which is the case for all samples tested. The question arises if the differences between 0.01 m/d or 0.001 m/d has any significant influence on the purpose of the research. In general, it is more valuable to estimate the higher relative permeabilities correctly for higher saturations. FIT1 should be the objective percolation curve if the prior statement is indeed applicable for the corresponding research. FIT2 is more valuable if the research is benefitted by exactly establishing at which water content the permeability approaches zero. Whichever research objective is applicable, percolation theory (PERC) should at least match one of them to become an addition to any research.

Overall, the estimations of percolation theory (PERC) would drastically improve if there was a better way of estimating the fractal dimensionality causing the cross-over water content to be more accurate. An exact estimation of the critical water content is for many research purposes not of great importance; besides, it has no major impact on the other parameters governing the relative permeability. However, three possible explanation for underestimating the critical water content are mentioned and could thus easily be investigated. Incorporating the improvement mentioned above would also improve the value of the calibration constant, which is dependent on the five key parameters related to percolation theory. Ultimately, universal scaling and critical path analysis hold a promising technique, although the current issues need to be resolved to exploit its full potential. It is a great tool to give an extra insight into the relation between the relative permeability and the water content.

8 Conclusion

This research introduces a new pore network structure model, the COMBO model, based on the fundamentals of medial axis thinning and the maximal ball algorithm. The skeletonization algorithm developed by Lee et al. (1994) is incorporated to extract a topological superior skeleton. Afterwards a merging algorithm is utilised based on the principles introduced by Dong and Blunt (2009), unifying overlapping pore bodies into single representative pore bodies. The merging algorithm is outstanding in preserving a geometrical correct pore space, while removing unrealistic resistances within pore bodies due to artificially created pore throats. Furthermore, the pore network model incorporates four characteristic improvement regarding the pore throat radius, pore throat length, pore body radius and shape factor. Especially the introduction of the inradius provides a great potential, as it is a scale independent method not limited by the position of the skeleton, also improving on the amount of otherwise unnecessarily removed pore throats. Using the inradius increases the robustness of the model, better dealing with resolution issues. Ultimately, the COMBO model is a combination of proven concepts combined with physically feasible improvements.

The COMBO model is tested on 12 different samples, generating results regarding the pore network properties, absolute permeability and relative permeability. The accuracy of the developed pore network model is not yet determined, as reliable benchmark data is absent, however, this will be the first objective when continuing this research. Calibrating the model and performing a sensitivity analysis on the connectivity, throat radius, throat length and shape factor would be a major contribution to this research area, as there are still many uncertainties. The relative permeability data is compared to percolation theory, providing new insights into the predictive capabilities of this analytical tool. Overall, percolation theory lacks accuracy due to the poorly estimated cross-over water content which is mainly affected by the fractal dimensionality of the sample. Resolving this issue in combination with other minor issues concerning the critical water content would drastically improve on the overall performance of the percolation theory.

References

Al-Kharusi, A. S., & Blunt, M. J. (2007). Network extraction from sandstone and carbonate pore space images. *Journal of Petroleum Science and Engineering*, *56*(4), 219-231.

Ambegaokar, V., Halperin, B. I., & Langer, J. S. (1971). Hopping conductivity in disordered systems. *Physical review B*, *4*(8), 2612.

Arganda-Carreras, I., Fernandez-Gonzalez, R., Munoz-Barrutia, A. & Ortiz-De-Solorzano, C.(2010). 3D reconstruction of histological sections: Application to mammary gland tissue, *Microscopy Research and Technique*, 73(11), 1019–1029.

Arya, L. M., & Paris, J. F. (1981). A physicoempirical model to predict the soil moisture characteristic from particle-size distribution and bulk density data 1. *Soil Science Society of America Journal*, *45*(6), 1023-1030.

Bakke, S., & Øren, P. E. (1997). 3-D pore-scale modelling of sandstones and flow simulations in the pore networks. *Spe Journal*, *2*(02), 136-149.

Balberg, I. (1987). Recent developments in continuum percolation. *Philosophical Magazine B*, *56*(6), 991-1003.

Baldwin, C. A., Sederman, A. J., Mantle, M. D., Alexander, P., & Gladden, L. F. (1996). Determination and characterization of the structure of a pore space from 3D volume images. *Journal of colloid and interface science*, *181*(1), 79-92.

Bird, N. R. A., Perrier, E., & Rieu, M. (2000). The water retention function for model of soil structure with pore and solid fractal distribution. *European of Soil Science*, *51*, 55-56.

Blunt, M. J. (2001). Flow in porous media—pore-network models and multiphase flow. *Current opinion in colloid & interface science*, *6*(3), 197-207.

Blunt, M. J., Bijeljic, B., Dong, H., Gharbi, O., Iglauer, S., Mostaghimi, P., ... & Pentland, C. (2013). Pore-scale imaging and modelling. *Advances in Water Resources*, *51*, 197-216.

Bruggeman, D. A. G. (1935). The calculation of various physical constants of heterogeneous substances. I. The dielectric constants and conductivities of mixtures composed of isotropic substances. *Annals of Physics*, *416*, 636-791.

Bryant, S. L., King, P. R., & Mellor, D. W. (1993). Network model evaluation of permeability and spatial correlation in a real random sphere packing. *Transport in Porous Media*, *11*(1), 53-70.

Bryant, S. L., Mellor, D. W., & Cade, C. A. (1993). Physically representative network models of transport in porous media. *AIChE Journal*, *39*(3), 387-396.

Bryant, S., & Blunt, M. (1992). Prediction of relative permeability in simple porous media. *Physical Review A*, *46*(4), 2004.

Bultreys, T., Van Hoorebeke, L., & Cnudde, V. (2015). Multi-scale, micro-computed tomography-based pore network models to simulate drainage in heterogeneous rocks. *Advances in Water Resources*, *78*, 36-49.

Bultreys, T., De Boever, W., & Cnudde, V. (2016). Imaging and image-based fluid transport modeling at the pore scale in geological materials: A practical introduction to the current state-of-the-art. *Earth-Science Reviews*, *155*, 93-128.

Bultreys, T., Lin, Q., Gao, Y., Raeini, A. Q., AlRatrout, A., Bijeljic, B., & Blunt, M. J. (2018). Validation of model predictions of pore-scale fluid distributions during two-phase flow. *Physical Review E*, *97*(5), 053104.

Dong, H., & Blunt, M. J. (2009). Pore-network extraction from micro-computerized-tomography images. *Physical review E*, *80*(3), 036307.

Fazzalari, N. L., & Parkinson, I. H. (1996). Fractal dimension and architecture of trabecular bone. *The Journal of pathology*, *178*(1), 100-105.

Filgueira, R. R., Pachepsky, Y. A., Fournier, L. L., Sarli, G. O., & Aragon, A. (1999). Comparison of fractal dimensions estimated from aggregate mass-size distribution and water retention scaling. *Soil science*, *164*(4), 217-223.

Ghanbarian, B., Hunt, A. G., Ewing, R. P., & Skinner, T. E. (2014). Universal scaling of the formation factor in porous media derived by combining percolation and effective medium theories. *Geophysical Research Letters*, *41*(11), 3884-3890.

Ghanbarian, B., Sahimi, M., & Daigle, H. (2016). Modeling relative permeability of water in soil: Application of effective-medium approximation and percolation theory. *Water Resources Research*, *52*(7), 5025-5040.

Ghanbarian-Alavijeh, B., & Hunt, A. G. (2012). Unsaturated hydraulic conductivity in porous media: Percolation theory. *Geoderma*, *187*, 77-84.

Hadwiger, H. (1957). Über eibereiche mit gemeinsamer treffgeraden. *Portugaliae mathematica*, *16*(1), 23-29.

Hao, L. & Cheng, P.(2010). Pore-scale simulations on relative permeabilities of porous media by lattice Boltzmann method. *International Journal of Heat and Mass Transfer.,* 53 (9) (2010), 1908-1913.

Homann, H. (2007). Implementation of a 3D thinning algorithm. *Insight Journal*, *421*.

Hunt, A. G. (2001). Applications of percolation theory to porous media with distributed local conductances. *Advances in Water Resources*, *24*(3-4), 279-307.

Hunt, A. G. (2005). Continuum percolation theory for transport properties in porous media. *Philosophical Magazine*, *85*(29), 3409-3434.

Hunt, A. G., & Gee, G. W. (2002a). Water-retention of fractal soil models using continuum percolation theory. *Vadose Zone Journal*, *1*(2), 252-260.

Hunt, A. G., & Gee, G. W. (2002b). Application of critical path analysis to fractal porous media: Comparison with examples from the Hanford site. *Advances in Water Resources*, *25*(2), 129-146.

Hunt, A., & Ewing, R. (2009). *Percolation Theory for Flow in Porous Media* (Vol. 771). Springer Science & Business Media.

Ioannidis, M. A., & Chatzis, I. (2000). On the geometry and topology of 3D stochastic porous media. *Journal of colloid and interface science*, *229*(2), 323-334.

Jiang, Z., Van Dijke, M. I. J., Sorbie, K. S., & Couples, G. D. (2013). Representation of multiscale heterogeneity via multiscale pore networks. *Water resources research*, *49*(9), 5437-5449.

Jiang, Z., Wu, K., Couples, G., Van Dijke, M. I. J., Sorbie, K. S., & Ma, J. (2007). Efficient extraction of networks from three-dimensional porous media. *Water Resources Research*, *43*(12).

Joekar Niasar, V., Hassanizadeh, S. M., Pyrak-Nolte, L. J., & Berentsen, C. (2009). Simulating drainage and imbibition experiments in a high-porosity micromodel using an unstructured pore network model. *Water resources research*, *45*(2).

Joekar-Niasar, V., & Hassanizadeh, S. M. (2011). Effect of fluids properties on non-equilibrium capillarity effects: Dynamic pore-network modeling. *International Journal of Multiphase Flow*, *37*(2), 198-214.

Joekar-Niasar, V., & Hassanizadeh, S. M. (2012). Analysis of fundamentals of two-phase flow in porous media using dynamic pore-network models: A review. *Critical reviews in environmental science and technology*, *42*(18), 1895-1976.

Joekar-Niasar, V., Prodanović, M., Wildenschild, D., & Hassanizadeh, S. M. (2010). Network model investigation of interfacial area, capillary pressure and saturation relationships in granular porous media. *Water Resources Research*, *46*(6).

Kirkpatrick, S. (1971). Classical transport in disordered media: scaling and effective-medium theories. *Physical Review Letters*, *27*(25), 1722.

Kirkpatrick, S. (1973). Percolation and conduction. *Reviews of modern physics*, *45*(4), 574.

Landauer, R. (1978, April). Electrical conductivity in inhomogeneous media. In *AIP Conference Proceedings* (Vol. 40, No. 1, pp. 2-45). AIP.

Lee, T.C., Kashyap R.L. & Chu, C.N. (1994). Building Skeleton Models via 3-D Medial Surface Axis Thinning Algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6), 462-478.

Lenormand, R., Zarcone, C., & Sarr, A. (1983). Mechanisms of the displacement of one fluid by another in a network of capillary ducts. *Journal of Fluid Mechanics*, *135*, 337-353.

Lerdahl, T. R., Oren, P. E., & Bakke, S. (2000, January). A predictive network model for three-phase flow in porous media. In *SPE/DOE Improved Oil Recovery Symposium*. Society of Petroleum Engineers.

Levine, S. A. M. U. E. L., & Cuthiell, D. L. (1986). Relative Permeabilitties In Two-Phase Flow Through Porous Media: An Application of Effective Medium Theory. *Journal of Canadian Petroleum Technology*, *25*(05).

Liang, Z., Ioannidis, M. A., & Chatzis, I. (2000). Geometric and topological analysis of three-dimensional porous media: pore space partitioning based on morphological skeletonization. *Journal of colloid and interface science*, *221*(1), 13-24.

Lindquist, W. B., Lee, S. M., Coker, D. A., Jones, K. W., & Spanne, P. (1996). Medial axis analysis of void structure in three-dimensional tomographic images of porous media. *Journal of Geophysical Research: Solid Earth*, *101*(B4), 8297-8310.

Lindquist, W. B., Venkatarangan, A., Dunsmuir, J., & Wong, T. F. (2000). Pore and throat size distributions measured from synchrotron X-ray tomographic images of Fontainebleau sandstones. *Journal of Geophysical Research: Solid Earth*, *105*(B9), 21509-21527.

Lorensen, W. E., & Cline, H. E. (1987, August). Marching cubes: A high resolution 3D surface construction algorithm. In *ACM siggraph computer graphics* (Vol. 21, No. 4, pp. 163-169). ACM.

Mason, G., & Morrow, N. R. (1991). Capillary behavior of a perfectly wetting liquid in irregular triangular tubes. *Journal of Colloid and Interface Science*, *141*(1), 262-274.

Meakin, P., & Tartakovsky, A. M. (2009). Modeling and simulation of pore-scale multiphase fluid flow and reactive transport in fractured and porous media. *Reviews of Geophysics*, *47*(3).

Moldrup, P., Olesen, T., Komatsu, T., Schjønning, P., & Rolston, D. E. (2001). Tortuosity, diffusivity, and permeability in the soil liquid and gaseous phases. *Soil Science Society of America Journal*, *65*(3), 613-623.

Ollion, J., Cochennec, J., Loll, F, Escudé, C & Boudier, T (2013). TANGO: A Generic Tool for High-throughput 3D Image Analysis for Studying Nuclear Organization. *Bioinformatics*, 29(14):1840-1.

Øren, P. E., & Bakke, S. (2002). Process based reconstruction of sandstones and prediction of transport properties. *Transport in porous media*, *46*(2-3), 311-343.

Øren, P. E., & Bakke, S. (2003). Reconstruction of Berea sandstone and pore-scale modelling of wettability effects. *Journal of Petroleum Science and Engineering*, *39*(3-4), 177-199.

Oren, P. E., Bakke, S., & Arntzen, O. J. (1997, January). Extending predictive capabilities to network models. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers.

Prodanović, M., Lindquist, W. B., & Seright, R. S. (2006). Porous structure and fluid partitioning in polyethylene cores from 3D X-ray microtomographic imaging. *Journal of Colloid and Interface Science*, *298*(1), 282-297.

Prodanović, M., Mehmani, A., & Sheppard, A. P. (2015). Imaged-based multiscale network modelling of microporosity in carbonates. *Geological Society, London, Special Publications*, *406*(1), 95-113.

Ramstad, T., Øren, P. E., & Bakke, S. (2010). Simulation of two-phase flow in reservoir rocks using a lattice Boltzmann method. *Spe Journal*, *15*(04), 917-927.

Ramstad, T., Idowu, N., Nardi, C., & Øren, P. E. (2012). Relative permeability calculations from two-phase flow simulations directly on digital images of porous rocks. *Transport in Porous Media*, *94*(2), 487-504.

Raoof, A., & Hassanizadeh, S. M. (2010). A new method for generating pore-network models of porous media. *Transport in porous media*, *81*(3), 391-407.

Raoof, A., & Hassanizadeh, S. M. (2012). A new formulation for pore-network modeling of two-phase flow. *Water Resources Research*, *48*(1).

Raoof, A., Nick, H. M., Hassanizadeh, S. M., & Spiers, C. J. (2013). PoreFlow: A complex pore-network model for simulation of reactive transport in variably saturated porous media. *Computers & Geosciences*, *61*, 160-174.

Rieu, M., & Sposito, G. (1991). Fractal fragmentation, soil porosity, and soil water properties: I. Theory. *Soil Science Society of America Journal*, *55*(5), 1231-1238.

Sahimi, M. (2003). *Heterogeneous Materials I: Linear transport and optical properties* (Vol. 22). Springer Science & Business Media.

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., ... & Tinevez, J. Y. (2012). Fiji: an open-source platform for biological-image analysis. *Nature methods*, *9*(7), 676.

Silin, D., & Patzek, T. (2006). Pore space morphology analysis using maximal inscribed spheres. *Physica A: Statistical mechanics and its applications*, *371*(2), 336-360.

Tartakovsky, A. M., Meakin, P., Scheibe, T. D., & West, R. M. E. (2007). Simulations of reactive transport and precipitation with smoothed particle hydrodynamics. *Journal of Computational Physics*, *222*(2), 654-672.

Thovert, J. F., Salles, J., & Adler, P. M. (1993). Computerized characterization of the geometry of real porous media: their discretization, analysis and interpretation. *Journal of microscopy*, *170*(1), 65-79.

Turcotte, D. L. (1986). Fractals and fragmentation. *Journal of Geophysical Research: Solid Earth*, *91*(B2), 1921-1926.

Van Genuchten, M. T. (1980). A closed-form equation for predicting the hydraulic conductivity of unsaturated soils 1. *Soil science society of America journal*, *44*(5), 892-898.

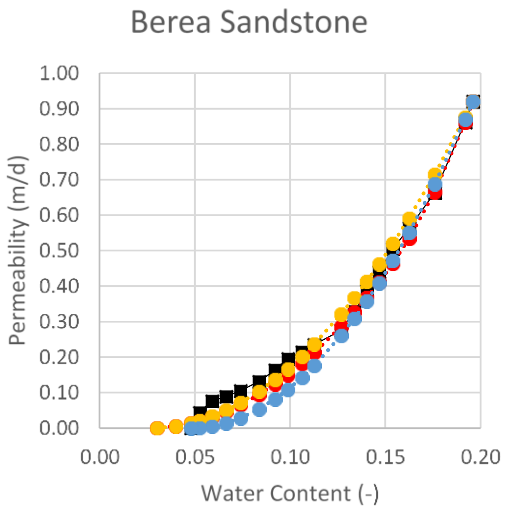
Vogel, H. J., & Roth, K. (2001). Quantitative morphology and network representation of soil pore structure. *Advances in water resources*, *24*(3-4), 233-242.

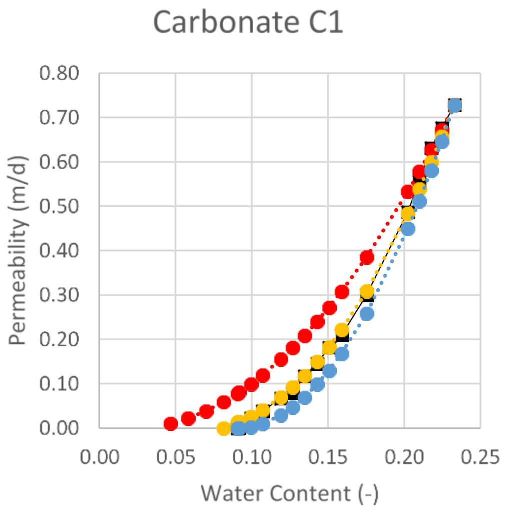
Xiong, Q., Baychev, T. G., & Jivkov, A. P. (2016). Review of pore network modelling of porous media: experimental characterisations, network constructions and applications to reactive transport. *Journal of contaminant hydrology*, *192*, 101-117.

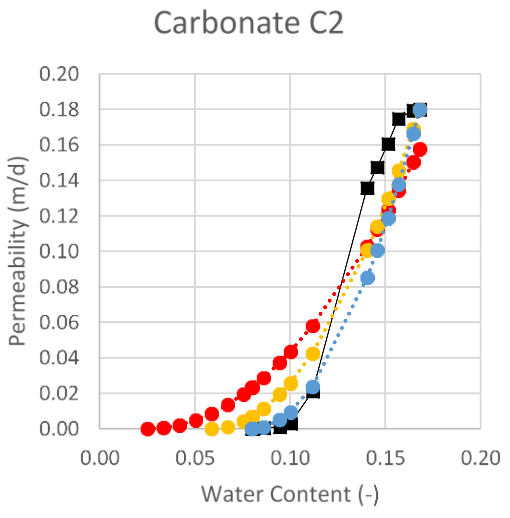
Yi, Z., Lin, M., Jiang, W., Zhang, Z., Li, H., & Gao, J. (2017). Pore network extraction from pore space images of various porous media systems. *Water Resources Research*, *53*(4), 3424-3445.

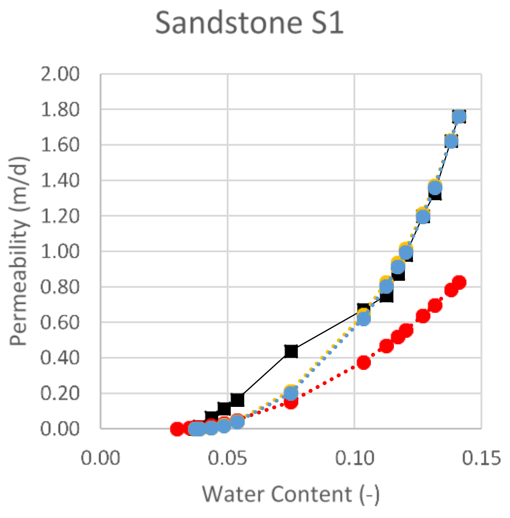
Zhang, M., Xu, K., He, Y., Jivkov, A.P (2014) Pore-scale modelling of 3D moisture distribution and critical saturation in cementitious materials. *Construction and Building Materials,* 64, pp. 222-230.

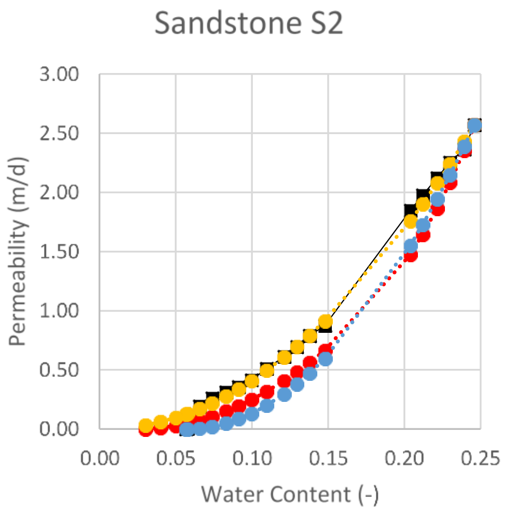
Appendix A

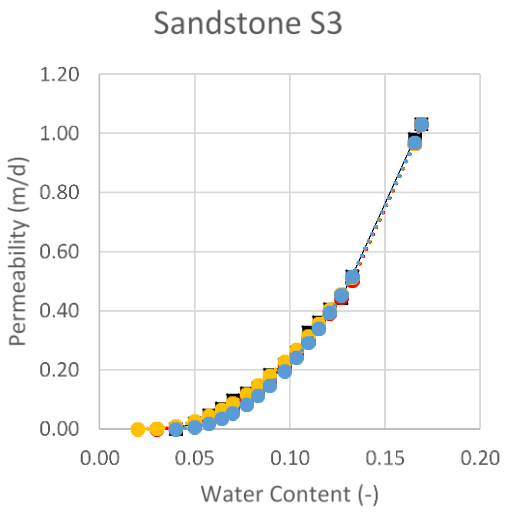

Berea Sandstone
Permeability (m/d)
Water Content (-)
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
0.05
0.10
0.15
0.20

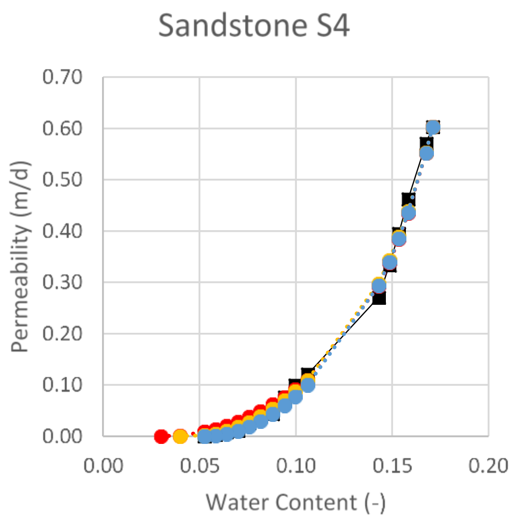

Carbonate C1
Permeability (m/d)
Water Content (-)
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
0.05
0.10
0.15
0.20
0.25

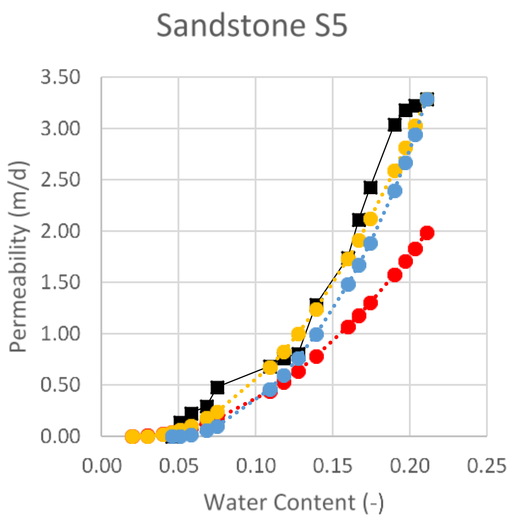

Carbonate C2
Permeability (m/d)
Water Content (-)
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0.00
0.05
0.10
0.15
0.20

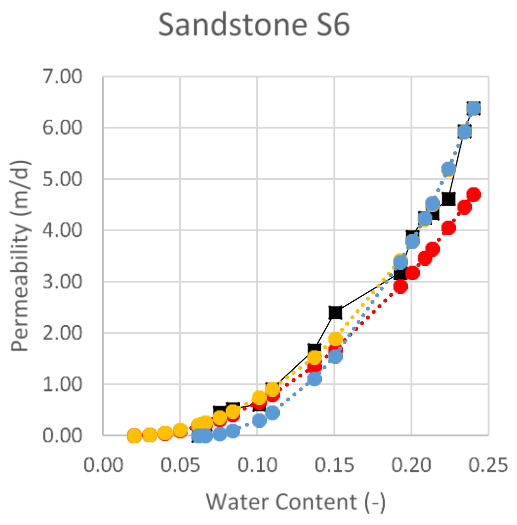

Sandstone S1
Permeability (m/d)
Water Content (-)
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.00
0.05
0.10
0.15

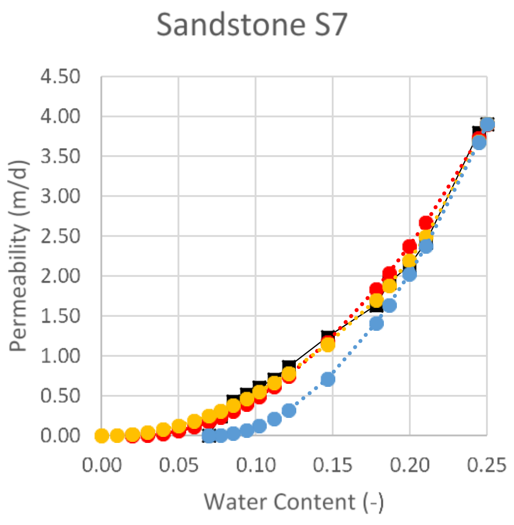

Sandstone S2
Permeability (m/d)
Water Content (-)
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.00
0.05
0.10
0.15
0.20
0.25

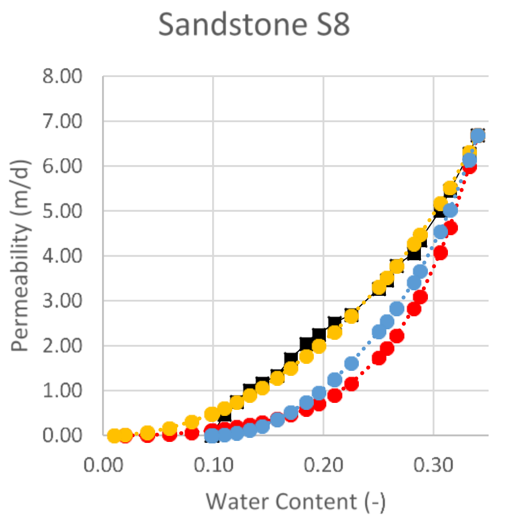

Sandstone S3
Permeability (m/d)
Water Content (-)
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.00
0.05
0.10
0.15
0.20











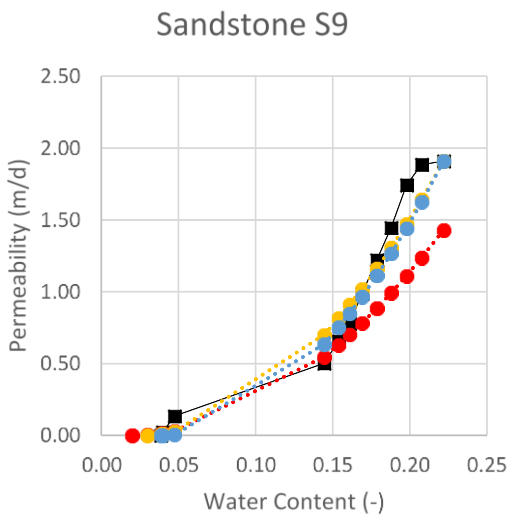
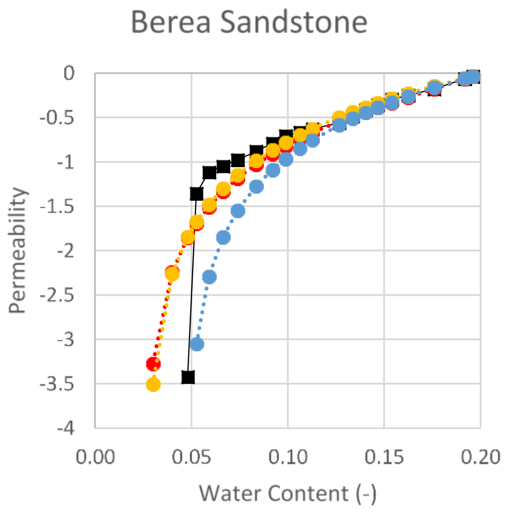
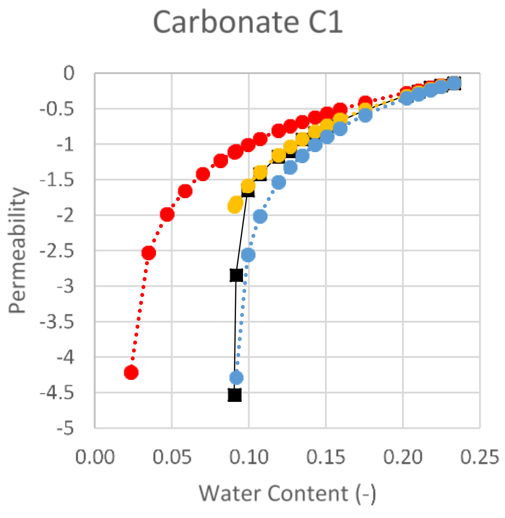
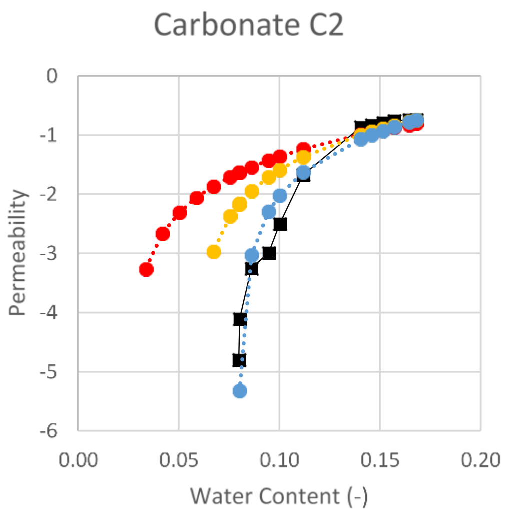


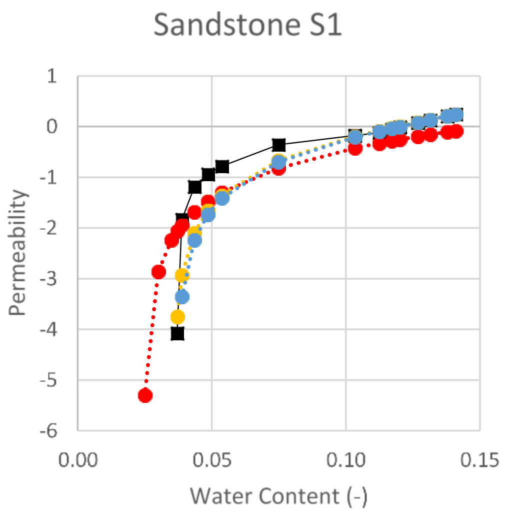
Figure A1 – A12. *Relative permeability versus water content for the twelve different samples. The black squares represent the results of PoreFlow, the red dashed line represent PERC, the yellow dashed line represents FIT1 and the blue dashed line represents FIT2.*

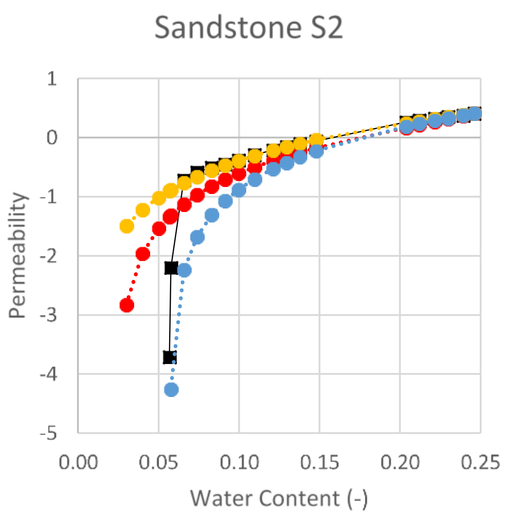
Appendix B

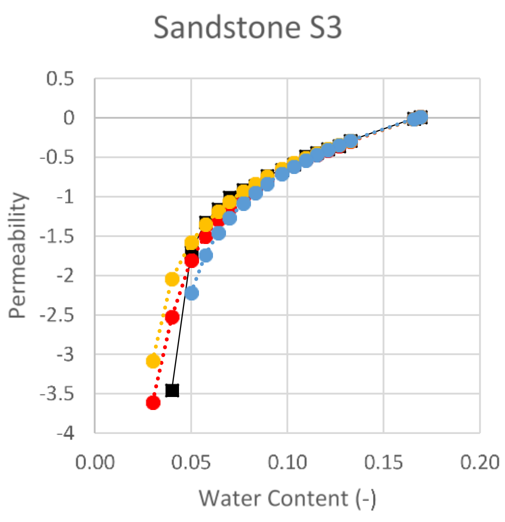

Berea Sandstone
Permeability
Water Content (-)


Carbonate C1
Permeability
Water Content (-)


Carbonate C2
Permeability
Water Content (-)


Sandstone S1
Permeability
Water Content (-)


Sandstone S2
Permeability
Water Content (-)


Sandstone S3
Permeability
Water Content (-)

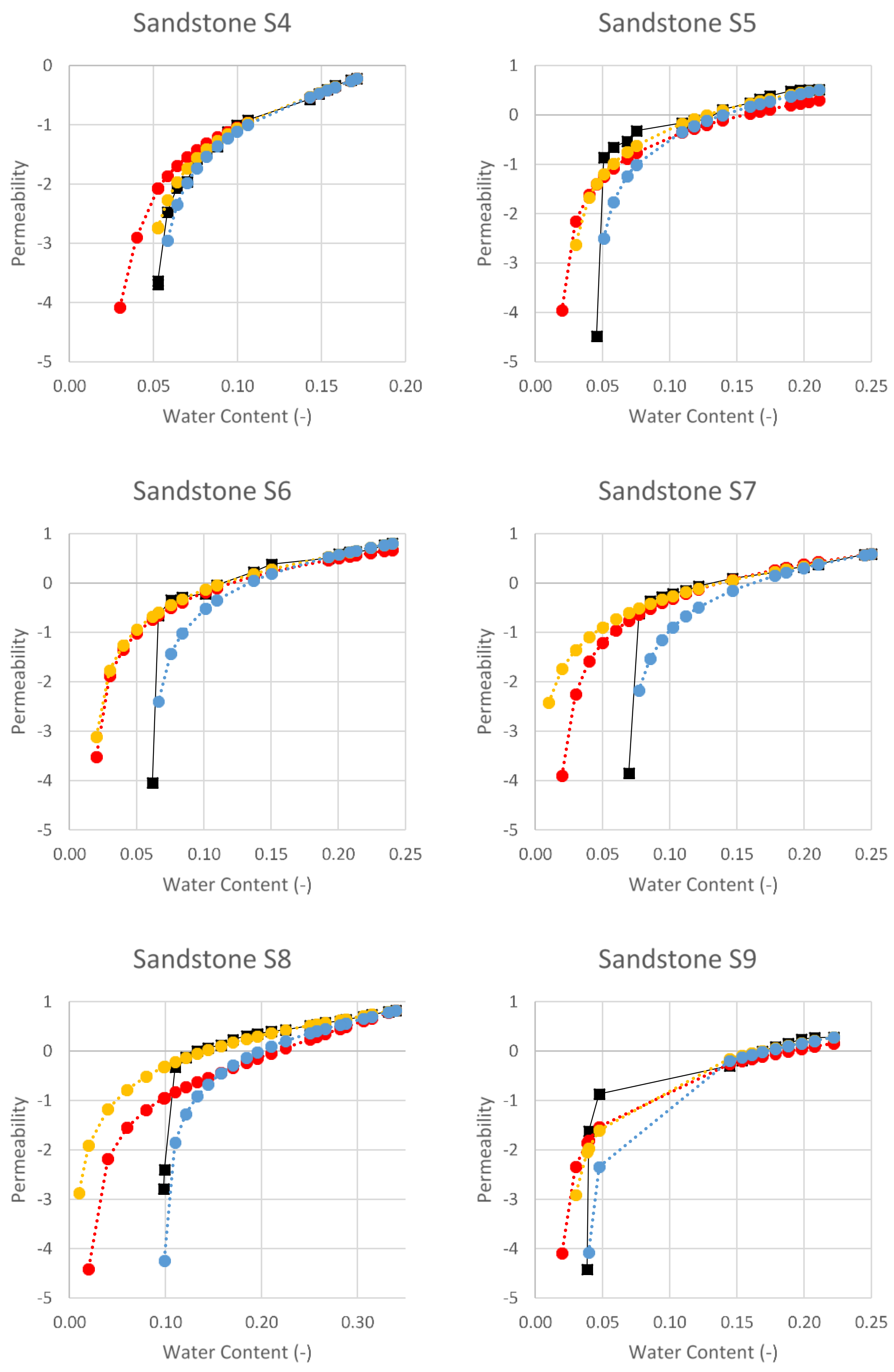


Figure B1 – B12. *Relative permeability on a logarithmic scale versus water content for the twelve different samples. The black squares represent the results of PoreFlow, the red dashed line represent PERC, the yellow dashed line represents FIT1 and the blue dashed line represents FIT2.*

Appendix C

FIJI (Fiji Is Just ImageJ) – Skeletonizing Cookbook

Preparing Your Samples Extraordinaire

The following plugins should be installed before proceeding with this manual.

- Help → Update.. → Ok

Click on "Manage update sites" and check the boxes in front of "3D ImageJ Suite", "ImageScience" and "BoneJ experimental".

Note: It is recommended to save you image stack after each step.

Step 1: Importing

Drag the folder with the greyscale pictures into FIJI. Click on yes **without** checking any boxes.

Step 2: Cropping

Select the area which you want to analyse, use different selection tools for different shapes. When the desired area has been selected follow the steps below.

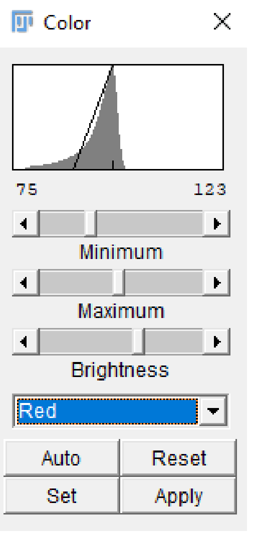
- Right click on the image → Duplicate → Check the box in front of "duplicate stack" → OK

When using a circular area the background can be removed after step 5 (The binarizing process).

Step 3: Colour Balancing

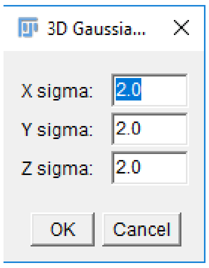
- Image → Adjust → Color Balance

Change the minimum and maximum by moving the sliders. Click on apply when satisfied, then click on yes.



Step 4: Smoothing

- Process → Filters → Gaussian Blur 3D → Put all sigma's on 2.0 → OK


3D Gaussia...

X sigma: 2.0

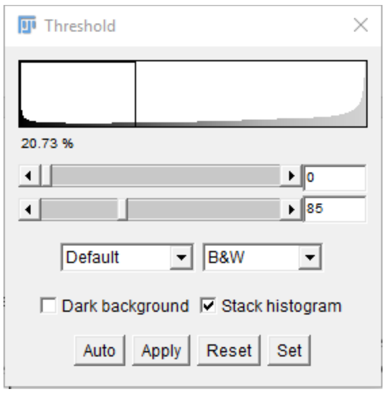
Y sigma: 2.0

Z sigma: 2.0

Step 5: Binarizing

- Image → Adjust → Threshold

Make sure to check the box in front of "Stack Histrogram". Adjust the lower slider and make sure that the percentage beneath the histogram is similar to the porosity of your sample. Click on apply when satisfied.


Threshold

20.73 %

0

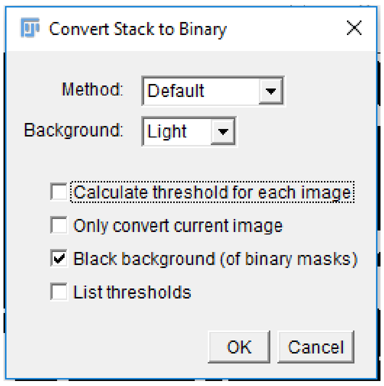
85

Default

B&W

☐ Dark background ☑ Stack histogram

A new window should pop up. Change the method to "Default", the background colour to "Light" and check the box in front of "Black background (of binary masks) ". Then click on OK.


Convert Stack to Binary

Method: Default

Background: Light

☐ Calculate threshold for each image

☐ Only convert current image

☑ Black background (of binary masks)

☐ List thresholds

Step 6: Removing Floating Solids

Plugins → 3D → 3D Fill Holes

Step 7: Checking The Porosity using the histogram

Analyze → Histogram

The porosity can be checked by looking at the histogram. The porosity should be equal to the amount of pore voxels, with value of 255, divided by the total amount of voxels. If these numbers do not match you should go back to step 5.

Step 8: Checking The Porosity and Connectivity using BoneJ2

Plugins → BoneJ → Fraction → Volume Fraction

The porosity will be calculated and shown in a table.

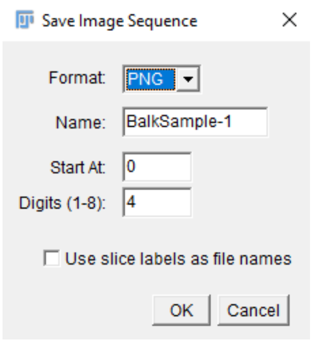
Plugins → BoneJ → Connectivity

The Euler number will be calculated. It is important to note that both parameters are not always readily available and should thus be obtained beforehand.

Step 9: Saving Binary Images

File → Save As → Image sequence

Change the file format to “PNG” and click on OK. Then select the file location at which you want to save the binary images.


Save Image Sequence

Format: PNG

Name: BalkSample-1

Start At: 0

Digits (1-8): 4

☐ Use slice labels as file names

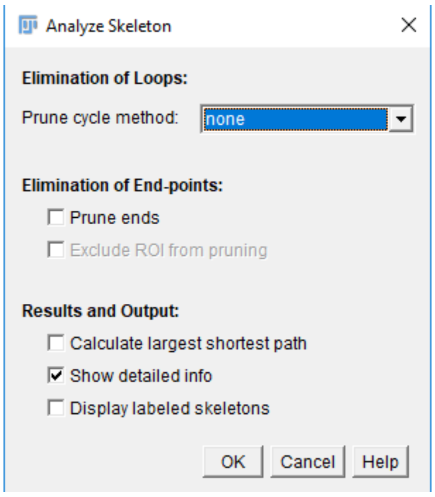
Step 10: Skeletonizing

Plugins → Skeletonize → Skeletonize (2D/3D)

Step 11: Analysing Skeleton

Analyze → Skeleton → Analyze Skeleton (2D/3D)

A new window should pop up, click on OK. The Branch information will pop up in a new window, save this file at a desired file location for later use.


Analyze Skeleton

Elimination of Loops:

Prune cycle method: none

Elimination of End-points:

☐ Prune ends

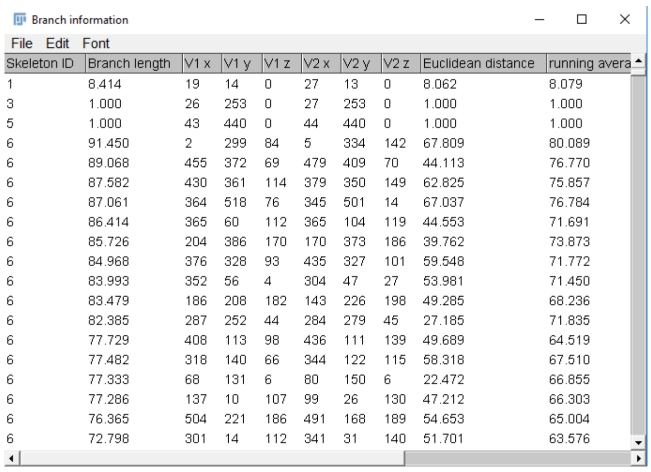
☐ Exclude ROI from pruning

Results and Output:

☐ Calculate largest shortest path

☑ Show detailed info

☐ Display labeled skeletons



Skeleton ID	Branch length	V1 x	V1 y	V1 z	V2 x	V2 y	V2 z	Euclidean distance	running avera
1	8.414	19	14	0	27	13	0	8.062	8.079
3	1.000	26	253	0	27	253	0	1.000	1.000
5	1.000	43	440	0	44	440	0	1.000	1.000
6	91.450	2	299	84	5	334	142	67.809	80.089
6	89.068	455	372	69	479	409	70	44.113	76.770
6	87.582	430	361	114	379	350	149	62.825	75.857
6	87.061	364	518	76	345	501	14	67.037	76.784
6	86.414	365	60	112	365	104	119	44.553	71.691
6	85.726	204	386	170	170	373	186	39.762	73.873
6	84.968	376	328	93	435	327	101	59.548	71.772
6	83.993	352	56	4	304	47	27	53.981	71.450
6	83.479	186	208	182	143	226	198	49.285	68.236
6	82.385	287	252	44	284	279	45	27.185	71.835
6	77.729	408	113	98	436	111	139	49.689	64.519
6	77.482	318	140	66	344	122	115	58.318	67.510
6	77.333	68	131	6	80	150	6	22.472	66.855
6	77.286	137	10	107	99	26	130	47.212	66.303
6	76.365	504	221	186	491	168	189	54.653	65.004
6	72.798	301	14	112	341	31	140	51.701	63.576

Step 12: Extracting The Skeleton

File → New → Script

A java console should appear. Change the language to BeanShell and open and run the BranchExtractor script. Save the log file at a desired file location for later use.

The FIJI procedure is now finished. Use the results as input for the different Python scripts. Enjoy your sample!

Appendix D

All scripts are written in python and should thus also be read and executed with a capable interpreter.

```
# PoreBodyDataExtracotr.py - Martijn van Leer & Floris Denekamp
# Uses Branch information.csv file and binary files (/Binaries) to obtain:
# - Pore Body locations
# - Distance Maps (distance between pore voxel and closest solid voxel)
# - Inscribed Sphere Radii of pore bodies and throats
# - Merging algorithm is applied for overlapping pore bodies
# - This file creates input files for NeighbourKernelArrayExtractor.py and ThroatDataExtractor.py

#Import toolkits
import numpy
import csv
from scipy import ndimage, misc
from PIL import Image
import os
import math
from collections import defaultdict
import time
import multiprocessing as mp
import datetime

Choose whether Maximal Ball algorithm has to be run
while True:
        CalcMB = raw_input('do you want to calculate the MB? y/n')
        if CalcMB == 'y' or CalcMB == 'n':
                break
        else:
                'Nope.'
print datetime.datetime.now(), '\n'

#Start timer
Start = time.time()

#Create empty lists for connected Pore Body coordinates by X, Y and Z
PB1X = []
PB1Y = []
PB1Z = []
PB2X = []
PB2Y = []
PB2Z = []

#Create empty lists for Coordinates of Pore Bodies
CorPB1 = []
CorPB2 = []
PBIndex1 = []
PBIndex2 = []

#Create empty lists for Tortuous Length
```

```
TorL = []
EucL = []
TorR = []

#Create empty lists for Pore Locations (all skeleton voxels)
PLoc = []

#Create empty lists for all Pore Body coordinates by X, Y and Z
PBX = []
PBY = []
PBZ = []
PBR = []

#Defining function to find clusters from pairs of overlapping pore bodies
# e.g. [[1,2],[1,3],[4,5],[4,6]] becomes [[1,2,3],[4,5,6]]
def connected_components(lists):
   neighbors = defaultdict(set)
   seen = set()
   for each in lists:
      for item in each:
         neighbors[item].update(each)
   def component(node, neighbors=neighbors, seen=seen, see=seen.add):
      nodes = set([node])
      next_node = nodes.pop
      while nodes:
         node = next_node()
         see(node)
         nodes |= neighbors[node] - seen
         yield node
   for node in neighbors:
      if node not in seen:
         yield sorted(component(node))

#Rename all binary files so it does not matter what the names are beforehand as long as the files are
in the right folder (/Binaries)
picnum1 = 1
for filename in sorted(os.listdir("Binaries")):
         dst = 'Binaries/BinSam' + str(picnum1).zfill(4) + '.png'
         src = 'Binaries/' + filename
         if not os.path.exists(dst):
                  os.rename(src,dst)
         picnum1 += 1

#Read csvfile of skeletonanalysis(/Data/Branch information.cvs) and append values to variables.
with open('Data/Branch information.csv', 'rb') as csvfile:
   reader = csv.reader(csvfile, delimiter=',')
   for row in reader:
      TorL.append(row[1])
      PB1X.append(row[2])
      PB1Y.append(row[3])
      PB1Z.append(row[4])
```

```
        PB2X.append(row[5])
        PB2Y.append(row[6])
        PB2Z.append(row[7])
        EucL.append(row[8])
csvfile.close()
print 'CSV file imported.'
print datetime.datetime.now(), '\n'

#Read binary images and create 3d array of it
nr = 1
while nr < picnum1:
        f = ('Binaries/BinSam'+str(nr).zfill(4)+'.png')
        row = ndimage.imread(f,flatten=1)
        PLoc.append(row)
        nr = nr + 1
print 'Binary images imported.'
print datetime.datetime.now(), '\n'

#Create 3D array with  all sides as solids
#Create a distance map for every pore voxel with euclidean distance to closest grain voxel
print 'Creating distance maps...'
SmallBinSam = numpy.array(PLoc)
SmallBinSam[SmallBinSam > 0] = 1
BinSam = numpy.zeros((SmallBinSam.shape[0]+2,SmallBinSam.shape[1]+2,SmallBinSam.shape[2]+2))
BinSam[1:-1,1:-1,1:-1] = SmallBinSam
SamDist = ndimage.distance_transform_edt(BinSam)
numpy.save('Data/BinarizedSampleArray', BinSam)   #Save the Binary arrary
print 'Binarized 3D arrays of the sample and skeleton are saved.'
numpy.save('Data/SampleDistanceArray', SamDist)   # Save the distance map
print 'Distance map created.'
print datetime.datetime.now(), '\n'

#Make strings of all three coordinates of the pore body locations, compensating for the artificial
solid sides of the sample
#create array to make list of all pore bodies
PBCorCheck1 = numpy.zeros_like(BinSam)
for p in range(1,len(PB1X)):
        PB1X[p] =int(PB1X[p]) + 1
        PB1Y[p] =int(PB1Y[p]) + 1
        PB1Z[p] =int(PB1Z[p]) + 1
        PB2X[p] =int(PB2X[p]) + 1
        PB2Y[p] =int(PB2Y[p]) + 1
        PB2Z[p] =int(PB2Z[p]) + 1
        CorPB1.append(str(PB1X[p]) + '\t' + str(PB1Y[p]) + '\t' + str(PB1Z[p]))
        CorPB2.append(str(PB2X[p]) + '\t' + str(PB2Y[p]) + '\t' + str(PB2Z[p]))
        PBCorCheck1[int(PB1Z[p]),int(PB1Y[p]),int(PB1X[p])] = 1

#Make porebody list unique
AllPB = list(set(CorPB1))
for p in range(1,len(PB2X)):
    if PBCorCheck1[int(PB2Z[p]),int(PB2Y[p]),int(PB2X[p])] == 0:
        AllPB.append(str(PB2X[p]) + '\t' + str(PB2Y[p]) + '\t' + str(PB2Z[p]))
        PBCorCheck1[int(PB2Z[p]),int(PB2Y[p]),int(PB2X[p])] = 1

#Find inscribed sphere radius of all pore body locations using the distance map
counter = 0
print 'All pore bodies found. Finding inscribed spheres.'
print datetime.datetime.now(), '\n'
PBCorCheckUnique = numpy.zeros_like(BinSam)
for q in range(0,len(AllPB)):
    PBX, PBY, PBZ = AllPB[q].split('\t')
    PBCorCheckUnique[int(PBZ), int(PBY), int(PBX)] = q
    PBR.append(SamDist[int(PBZ), int(PBY), int(PBX)])

PBRsum = []
print 'Inscribed spheres found. Creating lists by index...'
print datetime.datetime.now(), '\n'
#Create list of throats using connected pore body indexes
for p in range(len(CorPB2)):
    X1, Y1, Z1 = CorPB1[p].split('\t')
    X2, Y2, Z2 = CorPB2[p].split('\t')
    PBIndex1.append(int(PBCorCheckUnique[int(Z1),int(Y1),int(X1)]))
    PBIndex2.append(int(PBCorCheckUnique[int(Z2),int(Y2),int(X2)]))
    PBRsum.append(PBR[PBIndex1[p]] + PBR[PBIndex2[p]])

#calculate tortuosity
for p in range(1,len(TorL)):
        if float(EucL[p]) == 0:
                TorR.append(1.0)
        elif float(EucL[p]) > float(TorL[p]):
                TorR.append(1.0)
        else:
                TorR.append(float(TorL[p])/float(EucL[p]))

AllBraCor = []
OLDBraCor = []
Temp = []
#read pore throat skeleton locations
with open('Data/log.txt', 'r') as f:
    for line in f:
        inner_list = [elt.strip() for elt in line.split('endbranch')]
        if len(inner_list[0]) == 3:
            X,Y,Z = inner_list[0].split(',')
            ls = [X,Y,Z]
            AllBraCor.append(ls)
        else:
            AllBraCor.append(inner_list[0])
#create list of lists of throat skeleton locations
for p in range(0,len(AllBraCor)):
    if len(AllBraCor[p]) == 0:
        OLDBraCor.append(Temp)
        Temp = []
```

```
  else:
    X,Y,Z = AllBraCor[p].split(',')
    ls = [int(float(Z))+1,int(float(Y))+1,int(float(X))+1]
    Temp.append(ls)

#match skeleton branches with pore bodies
print 'Matching branches with pore bodies..'
print datetime.datetime.now(), '\n'
BraCor = []
PBcombo = []
Seen = [0] * len(PBIndex1)
for p in range(len(PBIndex1)):
			x1,y1,z1 = map(int,AllPB[PBIndex1[p]].split('\t'))
			x2,y2,z2 = map(int,AllPB[PBIndex2[p]].split('\t'))
			PBcombo.append([[z1,y1,x1],[z2,y2,x2]])
			for q in range(len(OLDBraCor)):
				if PBcombo[p][0] == OLDBraCor[q][0] and PBcombo[p][1] == OLDBraCor[q][-
1] and Seen[q] == 0:
					BraCor.append(OLDBraCor[q])
					Seen[q] = 1
					break

print 'Start Maximal Ball Pore Body Merge Protocol (MBPBMP)...'
print datetime.datetime.now(), '\n'
##The merge algorithm is started. It makes sure no overlapping pore bodies exists in the sample.
Overlap = [None]*len(CorPB2)
Iteration = 1
Cluster = [[]]
Cleanresults = []

#Find overlapping pore bodies
if CalcMB == 'y':
	for p in range(len(AllPB)):
		X1, Y1, Z1 = AllPB[p].split('\t')
		for q in range(len(AllPB)):
			X2, Y2, Z2 = AllPB[q].split('\t')
			CalcEucL = ((float(X1) - float(X2))**2 + (float(Y1) - float(Y2))**2 + (float(Z1) -
float(Z2))**2)
			if (PBR[p] + PBR[q])**2 >= float(CalcEucL) and p != q and [p,q] not in
Cleanresults and [q,p] not in Cleanresults:
				Cleanresults.append([p,q])
	with open('Data/cleanresults.txt', 'w') as cr:
		for p in range(len(Cleanresults)):
			cr.write(str(Cleanresults[p][0]) + '\t' + str(Cleanresults[p][1]) + '\n')
else:
	with open('Data/cleanresults.txt', 'r') as cr:
			Cleanresultsstr = cr.readlines()
	for p in Cleanresultsstr:
		Cleanresults.append(map(int,(p.split('\t'))))

print 'Overlapping pore bodies found. Running maximal ball algorithm..'
print datetime.datetime.now(), '\n'
Cluster = list(connected_components(Cleanresults))
total = 0

#using maximal ball in clusters
for p in range(len(Cluster)):
	Master = []
	Slave = []
	CR = []
	loc = []
	for x in Cluster[p]:
		CR.append([PBR[x],x])
	CR.sort(reverse = True)
	for t in CR:
		x,y,z = AllPB[t[1]].split('\t')
		loc.append([int(x),int(y),int(z)])
	for q in range(len(Cluster[p])):
		Slaafje = False
		for r in range(len(Master)):
			if (CR[q][0] + CR[Master[r][1]][0])**2 >= ((loc[Master[r][1]][0] - loc[q][0])**2
+ (loc[Master[r][1]][1] - loc[Master[r][1]][1])**2 + (loc[Master[r][1]][2] - loc[q][2])**2) and
Master[r][1] != q:
				Slave.append([CR[q][1],Master[r][0],q])
				Slaafje = True
				break
		if Slaafje == False:
			amount = len(Slave)
			for r in range(amount):
				if (CR[q][0] + CR[Slave[r][2]][0])**2 >= (loc[Slave[r][2]][0] -
loc[q][0])**2 + (loc[Slave[r][2]][1] - loc[q][1])**2 + (loc[Slave[r][2]][2] - loc[q][2])**2 and Slave[r][2]
!= q:
					Slave.append([CR[q][1],Slave[r][1],q])
					Slaafje = True
					break
		if Slaafje == False:
			if [CR[q][1],q] not in Master:
				Master.append([CR[q][1],q])
	for q in Slave:
		for p  in range(len(PBIndex1)):
			if PBIndex1[p] == q[0]:
				PBIndex1[p] = q[1]
			if PBIndex2[p] == q[0]:
				PBIndex2[p] = q[1]
	End = time.time()
	total += len(Slave)

print 'Masters and slaves defined. Amount of Slaves:', total

#The x, y and z coordinates, throat radius and aspect length.
ThrX = []
ThrY= []
```

```
ThrZ = []
ThrRadius = []
LengthRatio = []
ThrLength1 = []
ThrLength2 = []

#Finding throat locations with smallest radius
for p in range(len(BraCor)):
	RatioPointList = []
	MinBraDis1 = 10000
	if len(BraCor[p]) > 3:
		for q in range(1,len(BraCor[p])-1):
			if SamDist[BraCor[p][q][0],BraCor[p][q][1],BraCor[p][q][2]] < MinBraDis1:
				MinBraDis1 =
SamDist[BraCor[p][q][0],BraCor[p][q][1],BraCor[p][q][2]]
		for q in range(1,len(BraCor[p])-1):
			if SamDist[BraCor[p][q][0],BraCor[p][q][1],BraCor[p][q][2]] == MinBraDis1:
				RatioPointList.append(q)
		RatioPoint = RatioPointList[int((len(RatioPointList)-1) * 0.5)]
		LengthRatio.append((float(RatioPoint)+1.0)/(len(BraCor[p])+1))
		ThrX.append(BraCor[p][RatioPoint][2])
		ThrY.append(BraCor[p][RatioPoint][1])
		ThrZ.append(BraCor[p][RatioPoint][0])
		ThrRadius.append(MinBraDis1)
		EucL1 = math.sqrt((BraCor[p][0][2]- BraCor[p][-1][2])**2+(BraCor[p][0][1]-
BraCor[p][-1][1])**2+(BraCor[p][0][0]- BraCor[p][-1][0])**2)
		TorL = TorR[p] * EucL1
		X1, Y1, Z1 = map(int,AllPB[PBIndex1[p]].split('\t'))
		X2, Y2, Z2 = map(int,AllPB[PBIndex2[p]].split('\t'))
		MergeLength1 = math.sqrt((BraCor[p][0][2]- X1)**2+(BraCor[p][0][1]-
Y1)**2+(BraCor[p][0][0]- Z1)**2)
		MergeLength2 = math.sqrt((X2- BraCor[p][-1][2])**2+(Y2- BraCor[p][-1][1])**2+(Z2-
BraCor[p][-1][0])**2)
		ThrLength1.append(LengthRatio[p] * float(TorL) - PBR[PBIndex1[p]] +
MergeLength1)
		ThrLength2.append((1-LengthRatio[p]) * float(TorL) - PBR[PBIndex2[p]] +
MergeLength2)
		if ThrLength1[p] <= 0 or ThrLength2[p] <= 0:
			ThrLength1[p] = 0.5
			ThrLength2[p] = 0.5
	else:
		LengthRatio.append(0.5)
		ThrX.append(BraCor[p][1][2])
		ThrY.append(BraCor[p][1][1])
		ThrZ.append(BraCor[p][1][0])
		ThrRadius.append(SamDist[BraCor[p][1][0],BraCor[p][1][1],BraCor[p][1][2]])
		ThrLength1.append(0.4)
		ThrLength2.append(0.4)

print 'Midpoints of the throats determined.'
print datetime.datetime.now(), '\n'
```

```
#Finding dead-ends that need to be removed
CorNum = [0] * len(AllPB)
CorNumLoc = [0] * len(AllPB)
for p in range(len(ThrX)):
        CorNum[PBIndex1[p]] += 1
        CorNumLoc[PBIndex1[p]] += p
        CorNum[PBIndex2[p]] += 1
        CorNumLoc[PBIndex2[p]] += p

DeadEndcounter =0
for p in range(len(CorNum)):
        if CorNum[p] == 1:
                if PBR[p] <= ThrRadius[CorNumLoc[p]]:
                        DeadEndcounter +=1
                        if PBIndex1[CorNumLoc[p]] == p:
                                PBIndex1[CorNumLoc[p]] = PBIndex2[CorNumLoc[p]]
                        else:
                                PBIndex2[CorNumLoc[p]] = PBIndex1[CorNumLoc[p]]
print 'Amount of dead-ends removed:', DeadEndcounter

LocPB1 = []
LocPB2 = []
#Create new unique list of pore bodies, replacing the removed PB.
for p in PBIndex1:
  LocPB1.append(AllPB[p])
for p in PBIndex2:
  LocPB2.append(AllPB[p])
AllPBFinal = list(set(LocPB1))
n = 1
PBCorCheck2 = numpy.zeros_like(BinSam)
for p in range(len(AllPBFinal)):
  x,y,z = AllPBFinal[p].split('\t')
  PBCorCheck2[int(z),int(y),int(x)] = n
  n += 1
for p in range(len(LocPB2)):
  x,y,z = LocPB2[p].split('\t')
  if PBCorCheck2[int(z),int(y),int(x)] == 0:
    PBCorCheck2[int(z),int(y),int(x)] = n
    AllPBFinal.append(str(x) + '\t' + str(y) + '\t' + str(z))
    n+=1
for p in range(len(PBIndex1)):
  x1,y1,z1 = LocPB1[p].split('\t')
  x2,y2,z2 = LocPB2[p].split('\t')
  PBIndex1[p] = int(PBCorCheck2[int(z1),int(y1),int(x1)]-1)
  PBIndex2[p] = int(PBCorCheck2[int(z2),int(y2),int(x2)]-1)

#Creating array of PB indexes and removing merged PB.
print 'Removing merged pore bodies...'
print datetime.datetime.now(), '\n'
ThrXFinal = []
```

```
ThrYFinal = []
ThrZFinal = []
PBIndex1Final = []
PBIndex2Final = []
ThrRadiusFinal = []
TorRFinal = []
LengthRatioFinal = []
ThrLengthReal1 = []
ThrLengthReal2 = []
TortuousLengthReal = []

for p in range(len(ThrX)):
	if PBIndex1[p] != PBIndex2[p]:
		ThrXFinal.append(ThrX[p])
		ThrYFinal.append(ThrY[p])
		ThrZFinal.append(ThrZ[p])
		ThrRadiusFinal.append(ThrRadius[p])
		PBIndex1Final.append(int(PBIndex1[p]))
		PBIndex2Final.append(int(PBIndex2[p]))
		TorRFinal.append(TorR[p])
		LengthRatioFinal.append(LengthRatio[p])
		ThrLengthReal1.append(ThrLength1[p])
		ThrLengthReal2.append(ThrLength2[p])

#create new list of PBR
PBR = []
for p in AllPBFinal:
	x,y,z = map(int,p.split('\t'))
	PBR.append(SamDist[z,y,x])

#savety check for throat length
for p in range(len(ThrLengthReal1)):
	if ThrLengthReal1[p] <= 0:
		ThrLengthReal1[p] = 0.5
	if ThrLengthReal2[p] <= 0:
		ThrLengthReal2[p] = 0.5

print 'Writing data to file...'
print datetime.datetime.now(), '\n'

#writing data to files
with open ('Data/ThroatLocations.txt', 'w') as TLOC:
	with open('Data/ThroatRadius.txt', 'w') as TR:
		with open('Data/ThroatsIndex.txt', 'w') as TI:
			with open ('Data/RealThroatLengthOne.txt', 'w') as RTLO:
				with open ('Data/RealThroatLengthTwo.txt', 'w') as RTLT:
					with open ('Data/LengthRatio.txt', 'w') as LR:
						for p in range(len(ThrRadiusFinal)):
				TLOC.write(str(ThrXFinal[p]) + '\t' + str(ThrYFinal[p]) + '\t' +
str(ThrZFinal[p]) + '\n')
							TR.write(str(ThrRadiusFinal[p]) + '\n')
```

```
                                                TI.write(str(PBIndex1Final[p]) + '\t' +
str(PBIndex2Final[p]) + '\n')
                                                RTLO.write(str(ThrLengthReal1[p]) + '\n')
                                                RTLT.write(str(ThrLengthReal2[p]) + '\n')
                                                LR.write(str(LengthRatioFinal[p]) + '\n')

with open ('Data/PoreBodyCoordinates.txt', 'w') as PBC:
        with open ('Data/PoreBodyRadius.txt', 'w') as PB:
                for p in range(len(AllPBFinal)):
                        PBC.write(str(AllPBFinal[p]) + '\n')
                        PB.write(str(PBR[p]) + '\n')

End = time.time()

print 'Finished. Total runtime:', round(End - Start), 's.'
print datetime.datetime.now(), '\n'
#Seal of Approval if everything went alright.
print '           .---.                     '
print '          /o   o\                    '
print '       __(=  "  =)__                 '
print '        //\`-=-`/\\                  '
print '           )   (_                    '
print '          /      `==-._              '
print '         /      \     ``==.          '
print '        /  /   \  \        `=..--._  '
print '    ___/  /     \  \___      _,  ,  \          '
print '    `-----`"""""`------`""` \  \__/          '
print '                            `-`          '
print '             Seal of Approval                 '
```

```
#NeighborCounter.py - Martijn van Leer & Floris Denekamp - PoreBodyDataExtractor has to be run
before this.
# create xy, yz & xz planes and count the neighbors in those directions.
#These could be added up to find the perimeter in ThroatDataExtractor.py

#import toolkits
import numpy
from scipy import signal
from os import mkdir, path
from scipy import ndimage, misc
from PIL import Image
import datetime

print datetime.datetime.now(), '\n'

#Load the binary sample and make it inverse
BinSam = numpy.load('Data/BinarizedSampleArray.npy')
BinSam = 1 - BinSam
BinSam = BinSam.astype(int)

#a 3x3x3 kernel is created 9 times to define different planes through a cube by defining directions
from the centre in which neighbours should be counted. The values in which neighbours should be
counted are assigned value 1.0. The arrays are saved as NB#.npy
Kernel1 = numpy.zeros((3,3,3), dtype = float)#1
Kernel1[1,0,1] = 1.0
Kernel1[1,1,0] = 1.0
Kernel1[1,1,2] = 1.0
Kernel1[1,2,1] = 1.0
NB1 = signal.convolve(BinSam,Kernel1, mode = 'same')
numpy.save('Data/NB1.npy', NB1)
del NB1
print 'Kernel 1/9 created.'
print datetime.datetime.now(), '\n'

Kernel2 = numpy.zeros((3,3,3), dtype = float)#2
Kernel2[0,1,1] = 1.0
Kernel2[1,1,0] = 1.0
Kernel2[1,1,2] = 1.0
Kernel2[2,1,1] = 1.0
NB2 = signal.convolve(BinSam,Kernel2, mode = 'same')
numpy.save('Data/NB2.npy', NB2)
del NB2
print 'Kernel 2/9 created.'
print datetime.datetime.now(), '\n'

Kernel3 = numpy.zeros((3,3,3), dtype = float)#3
Kernel3[0,1,1] = 1.0
Kernel3[1,0,1] = 1.0
Kernel3[1,2,1] = 1.0
Kernel3[2,1,1] = 1.0
NB3 = signal.convolve(BinSam,Kernel3, mode = 'same')
```

```
numpy.save('Data/NB3.npy', NB3)
del NB3
print 'Kernel 3/9 created.'
print datetime.datetime.now(), '\n'

Kernel4 = numpy.zeros((3,3,3), dtype = float)#4
Kernel4[1,0,2] = 1.0
Kernel4[1,2,0] = 1.0
Kernel4[2,1,1] = 1.4142136
Kernel4[0,1,1] = 1.4142136
NB4 = signal.convolve(BinSam,Kernel4, mode = 'same')
numpy.save('Data/NB4.npy', NB4)
del NB4
print 'Kernel 4/9 created.'
print datetime.datetime.now(), '\n'

Kernel5 = numpy.zeros((3,3,3), dtype = float)
Kernel5[0,1,2] = 1.4142136
Kernel5[2,1,0] = 1.4142136
Kernel5[1,2,1] = 1.0
Kernel5[1,0,1] = 1.0
NB5 = signal.convolve(BinSam,Kernel5, mode = 'same')
numpy.save('Data/NB5.npy', NB5)
del NB5
print 'Kernel 5/9 created.'
print datetime.datetime.now(), '\n'

Kernel6 = numpy.zeros((3,3,3), dtype = float)
Kernel6[2,0,1] = 1.4142136
Kernel6[0,2,1] = 1.4142136
Kernel6[1,1,2] = 1.0
Kernel6[1,1,0] = 1.0
NB6 = signal.convolve(BinSam,Kernel6, mode = 'same')
numpy.save('Data/NB6.npy', NB6)
del NB6
print 'Kernel 6/9 created.'
print datetime.datetime.now(), '\n'

Kernel7 = numpy.zeros((3,3,3), dtype = float)
Kernel7[1,2,2] = 1.0
Kernel7[1,0,0] = 1.0
Kernel7[2,1,1] = 1.4142136
Kernel7[0,1,1] = 1.4142136
NB7 = signal.convolve(BinSam,Kernel7, mode = 'same')
numpy.save('Data/NB7.npy', NB7)
del NB7
print 'Kernel 7/9 created.'
print datetime.datetime.now(), '\n'

Kernel8 = numpy.zeros((3,3,3), dtype = float)
Kernel8[2,1,2] = 1.0
```

```
Kernel8[0,1,0] = 1.0
Kernel8[1,2,1] = 1.4142136
Kernel8[1,0,1] = 1.4142136
NB8 = signal.convolve(BinSam,Kernel8, mode = 'same')
numpy.save('Data/NB8.npy', NB8)
del NB8
print 'Kernel 8/9 created.'
print datetime.datetime.now(), '\n'

Kernel9 = numpy.zeros((3,3,3), dtype = float)
Kernel9[2,2,1] = 1.0
Kernel9[0,0,1] = 1.0
Kernel9[1,1,2] = 1.4142136
Kernel9[1,1,0] = 1.4142136
NB9 = signal.convolve(BinSam,Kernel9, mode = 'same')
numpy.save('Data/NB9.npy', NB9)
del NB9
print 'Kernel 9/9 created.'
print datetime.datetime.now(), '\n'
#Seal of Approval shows up if everything went ok.
print 'Neighbour maps created. Finished.'
print '          .---.                       '
print '         /o   o\                      '
print '      __(=  "  =)__                   '
print '       //\`-=-`/\\                    '
print '          )   (_                      '
print '         /      `==-._                '
print '        /      \    ``==.            '
print '       / /  \ \       `=..--._       '
print '    ___/ /   \ \___     _, ,  \      '
print '   `-----`""""""`------`''''`\  \__/     '
print '                     `-`              '
print '           Seal of Approval           '
```

```
#ThroatRadiusFinder.py - Martijn van Leer & Floris Denekamp - PoreBodyDataExtractor.py and
NeighbourKernelArrayExtractor.py have to be run for this script to work.
#This script:
# - obtains Pore volumes using watershed
# - obtains smallest locations of throats
# - obtains Shape factors on throat locations
# - obtains Volumes of throats
# - Merges the throtas which connect the same pore bodies

#import toolkits
import numpy
import csv
import os
import math
import time
from scipy import ndimage, misc
from skimage.morphology import watershed
from collections import defaultdict
import multiprocessing as mp
import datetime

while True:
	CalcSH = raw_input("Do you want to calculate the Shapefactor?(y/n)")
	if CalcSH == 'y' or CalcSH == 'n':
		break
	print 'That is not a correct input.'

#Define same function as in PoreBodyDataExtractor
def connected_components(lists):
  neighbors = defaultdict(set)
  seen = set()
  for each in lists:
    for item in each:
      neighbors[item].update(each)
  def component(node, neighbors=neighbors, seen=seen, see=seen.add):
    nodes = set([node])
    next_node = nodes.pop
    while nodes:
      node = next_node()
      see(node)
      nodes |= neighbors[node] - seen
      yield node
  for node in neighbors:
    if node not in seen:
      yield sorted(component(node))

#Start the timer and define function to show the elapsed time
Start = time.time()
print datetime.datetime.now(), '\n'
def Showtime(Start):
  End = time.time()
```

```
    Elapsed  = round(End - Start,0)
    print  'It took ', Elapsed,'s to run so far. '

#Pore body 1 is connected to pore body 2, connected by a throat
ThrIndex1 =[]
ThrIndex2 =[]

#Pore body radii
PBR = []

#Pore body x, y and z coordinates
PBX = []
PBY = []
PBZ = []
CorPB1 = []
CorPB2 = []
AllPB = []

#Throat parameters
ThrX = []
ThrY = []
ThrZ = []
ThrRadius = []
ThrArea = []
ThrPeri = []
RealThroatLength1 = []
RealThroatLength2 = []
LengthRatio = []

print 'Importing text data...'

#Read data from the files created in PoreBodyDataExtractor.py
with open('Data/ThroatsIndex.txt', 'rb') as csvfile:
    reader = csv.reader(csvfile, delimiter = '\t')
    for row in reader:
        ThrIndex1.append(int(row[0]))
        ThrIndex2.append(int(row[1]))

with open('Data/PoreBodyCoordinates.txt', 'rb') as csvfile:
    reader = csv.reader(csvfile, delimiter = '\n')
    for row in reader:
         AllPB.append((row[0]))

with open('Data/RealThroatLengthOne.txt', 'rb') as csvfile:
    reader = csv.reader(csvfile, delimiter = '\n')
    for row in reader:
         RealThroatLength1.append(float(row[0]))

with open('Data/RealThroatLengthTwo.txt', 'rb') as csvfile:
    reader = csv.reader(csvfile, delimiter = '\n')
    for row in reader:
        RealThroatLength2.append(float(row[0]))

with open('Data/LengthRatio.txt', 'rb') as csvfile:
    reader = csv.reader(csvfile, delimiter = '\n')
    for row in reader:
        LengthRatio.append(float(row[0]))

ThrLoc = []
with open('Data/ThroatLocations.txt', 'rb') as TL:
            reader = csv.reader(TL, delimiter = '\n')
            for row in reader:
        ThrLoc.append((row[0]))

with open('Data/ThroatRadius.txt', 'rb') as TR:
            reader = csv.reader(TR, delimiter = '\n')
            for row in reader:
        ThrRadius.append(float(row[0]))

with open('Data/PoreBodyRadius.txt', 'rb') as PB:
            reader = csv.reader(PB, delimiter = '\n')
            for row in reader:
        PBR.append(float(row[0]))

for p in ThrLoc:
            x,y,z = map(int,p.split('\t'))
            ThrX.append(x)
            ThrY.append(y)
            ThrZ.append(z)

RealThroatLength = []
for p in range(len(RealThroatLength1)):
            RealThroatLength.append(RealThroatLength1[p] + RealThroatLength2[p])

BinSam = numpy.load('Data/BinarizedSampleArray.npy')

print 'Starting watershed on pore bodies...'
print 'Creating watershed markers...'
#watershed algorithm to obtain pore volumes
Markers = numpy.zeros_like(BinSam).astype(numpy.int32)
BinMarkers = numpy.zeros_like(BinSam).astype(numpy.int32)
counter = 0
#create markers over the volume of the pore bodies using the inscribed sphere, so the volumes are
almost never smaller than the volume of the inscribed sphere.
for p in range(len(AllPB)):
            x,y,z = AllPB[p].split('\t')
            x = int(x)
            y = int(y)
            z = int(z)
            for a in range(-int(PBR[p])+1, int(PBR[p])):
                        for b in range(-int(PBR[p])+1, int(PBR[p])):
                                    for c in range(-int(PBR[p])+1, int(PBR[p])):
                                        if (a**2 + b**2 + c**2) < int(PBR[p])**2 and BinSam[z - c, y - b, x - a]
== 1:
                                                if Markers[z - c, y - b, x - a] != 0:
                                                        counter += 1
                                                Markers[z - c, y - b, x - a] = p + 1

print 'Overlapping watershed marker voxels:',counter #This can be caused by discretization error
from cubic voxels to spheres.
print 'Starting watershed algorithm...',
Showtime(Start)
print datetime.datetime.now(), '\n'

#Creating binary markers, markers will be equal to zero.
BinMarkers[Markers > 0] = 1
BinMarkers = 1 - BinMarkers

#Creating a distance map for the binarized markers
DistBinMarkers = ndimage.morphology.distance_transform_edt(BinMarkers)
#dtype = float64

#Creating the watershed segments
print 'Segmentation completed.',
Showtime(Start)
print datetime.datetime.now(), '\n'

Segments =  watershed(DistBinMarkers, Markers, mask = BinSam)

print 'Watershed algorithm completed.'
print 'Calculating volumes...'

#add up the volumes of the watershed output to obtain the volume per pore, and assign values for
boundary pores if the segment touches the side of the domain.
def Volumef(a):
        Boundaries = [0] * len(AllPB)
        SegmentVolume = [0] * len(AllPB)
        for b in range(0,Segments.shape[1]):
                for c in range(0,Segments.shape[2]):
                        if Segments[a,b,c] != 0:
                                SegmentVolume[int(Segments[a,b,c])-1]
=SegmentVolume[int(Segments[a,b,c])-1] + 1
                        #Find in & outlet boundary pores
                        if a == 1:
                                Boundaries[int(Segments[a,b,c])-1] = 1
                        elif a == Segments.shape[0]-2:
                                Boundaries[int(Segments[a,b,c])-1] = 2
        return [SegmentVolume, Boundaries]
pool = mp.Pool(processes=10) #amount of cores used to calculate volumes
TempResults = pool.map(Volumef, range(0,Segments.shape[0]))
pool.close()
pool.join()
#create seperate lists for boundaries and pore body volumes
```

```
SegmentVolume, Boundaries = zip(*TempResults)
SegmentVolume = [sum(i) for i in zip(*SegmentVolume)]
Boundaries = [max(i) for i in zip(*Boundaries)]
print 'Pore body volumes calculated.'

CalculatedVolume = []
for p in range(len(SegmentVolume)):
        x,y,z = AllPB[p].split('\t')
        x = int(x)
        y = int(y)
        z = int(z)
        CalculatedVolume.append((4.0/3.0)*(PBR[p])**3.0*math.pi)
        if SegmentVolume[p] == 0:
                SegmentVolume[p] = 1.0

print 'Total amount of throats: ', len(ThrX)

print 'Finding shapefactors...',
Showtime(Start)
print datetime.datetime.now(), '\n'
#count pore cells in the plane with the smallest area, and calculate shapefactors
def SFf(p):
        Area = []
        Perimeterlist = []
        for Dir in range(1,10):
        SF = 0
        Perimeter = 0
        d = 0
        e = 0
        ThroatPointChecker = numpy.zeros_like(BinSam).astype(numpy.int32)
        PorePoints = []
            HorPoints = []
            VerPoints = []
        VerPointsUpdater = []
            HorPointsUpdater = []
            NBMap = numpy.load('Data/NB'+str(Dir) +'.npy')
            if ThrIndex1[p] != ThrIndex2[p]:
              if Dir ==1:
                xUpdater = 'd'
                yUpdater = 'e'
                zUpdater = '0'
                AreaUnit = 1.0
              if Dir ==2:
                xUpdater = 'd'
                yUpdater = '0'
                zUpdater = 'e'
                AreaUnit = 1.0
              if Dir==3:
                    xUpdater = '0'
                yUpdater = 'd'
                zUpdater = 'e'
                AreaUnit = 1.0
            if Dir ==4:
                xUpdater = 'd'
                yUpdater = '-d'
                zUpdater = 'e'
                AreaUnit = 1.41421356237
            if Dir ==5:
                xUpdater = 'd'
                yUpdater = 'e'
                zUpdater = '-d'
                AreaUnit = 1.41421356237
            if Dir ==6:
                xUpdater = 'e'
                yUpdater = 'd'
                zUpdater = '-d'
                AreaUnit = 1.41421356237
            if Dir ==7:
                xUpdater = 'd'
                yUpdater = 'd'
                zUpdater = 'e'
                AreaUnit = 1.41421356237
            if Dir ==8:
                xUpdater = 'd'
                yUpdater = 'e'
                zUpdater = 'd'
                AreaUnit = 1.41421356237
            if Dir ==9:
                xUpdater = 'e'
                yUpdater = 'd'
                zUpdater = 'd'
                AreaUnit = 1.41421356237

            PorePoints.append([ThrZ[p],ThrY[p],ThrX[p]])
            ThroatPointChecker[PorePoints[0][0],PorePoints[0][1],PorePoints[0][2]] =1
            HorPoints.append([ThrZ[p],ThrY[p],ThrX[p]])
            VerPoints.append([ThrZ[p],ThrY[p],ThrX[p]])
            while True:
                    for point in HorPoints:
                        while True:
                            d +=1
                            if d == 1 and ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] == 1:
                                d = 0
                                break
                            if BinSam[point[0]+ eval(zUpdater),point[1]+ eval(yUpdater),point[2]+
eval(xUpdater)] == 1:
                                if ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] == 0:
                                    PorePoints.append([point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)])
```

```
                    VerPointsUpdater.append([point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)])
                  ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] += 1
                elif BinSam[point[0]+ eval(zUpdater),point[1]+ eval(yUpdater),point[2]+
eval(xUpdater)] == 0:
                  d = 0
                  break

            for point in HorPoints:
              while True:
                d -=1
                if d == -1 and ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] == 1:
                  d = 0
                  break
                if BinSam[point[0]+ eval(zUpdater),point[1]+ eval(yUpdater),point[2]+
eval(xUpdater)] == 1:
                  if ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] == 0:
                    PorePoints.append([point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)])
                    VerPointsUpdater.append([point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)])
                  ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] += 1
                elif BinSam[point[0]+ eval(zUpdater),point[1]+ eval(yUpdater),point[2]+
eval(xUpdater)] == 0:
                  d = 0
                  break

            for point in VerPoints:
              while True:
                e +=1
                if e == 1 and ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] == 1:
                  e = 0
                  break
                if BinSam[point[0]+ eval(zUpdater),point[1]+ eval(yUpdater),point[2]+
eval(xUpdater)] == 1:
                  if ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] == 0:
                    PorePoints.append([point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)])
                    HorPointsUpdater.append([point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)])
                  ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] += 1
                elif BinSam[point[0]+ eval(zUpdater),point[1]+ eval(yUpdater),point[2]+
eval(xUpdater)] == 0:
                  e = 0
```

```
                                break

                        for point in VerPoints:
                            while True:
                                e -=1
                                if e == -1 and ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] == 1:
                                    e = 0
                                    break
                                if BinSam[point[0]+ eval(zUpdater),point[1]+ eval(yUpdater),point[2]+
eval(xUpdater)] == 1:
                                    if ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] == 0:
                                        PorePoints.append([point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)])
                                        HorPointsUpdater.append([point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)])
                                    ThroatPointChecker[point[0]+ eval(zUpdater),point[1]+
eval(yUpdater),point[2]+ eval(xUpdater)] += 1
                                elif BinSam[point[0]+ eval(zUpdater),point[1]+ eval(yUpdater),point[2]+
eval(xUpdater)] == 0:
                                    e = 0
                                    break

                        if (len(HorPointsUpdater) == 0  and len(VerPointsUpdater) == 0):
                                break
                        if len(Area) != 0:
                                if len(PorePoints) * AreaUnit > min(Area):
                                        break
                        HorPoints = []
                        for point in HorPointsUpdater:
                            if ThroatPointChecker[point[0],point[1],point[2]] == 1:
                                HorPoints.append(point)
                        VerPoints = []
                        for point in VerPointsUpdater:
                            if ThroatPointChecker[point[0],point[1],point[2]] == 1:
                                VerPoints.append(point)
                        HorPointsUpdater = []
                        VerPointsUpdater = []
                    for point in PorePoints:
                        Perimeter += NBMap[point[0],point[1],point[2]]
                    Area.append(len(PorePoints) * AreaUnit)
                    Perimeterlist.append(Perimeter)
                else:
                    Area.append(-1)
                    Perimeterlist.append(-1)
                    SF = -1
        ID = Area.index(min(Area))
        if p in range(0,len(ThrX),int(len(ThrX)/10)):
                print round(float(p)/len(ThrX)*100), '%finished'
                Showtime(Start)
        return [p, Area[ID], Perimeterlist[ID]]

if CalcSH == 'y':
        pool = mp.Pool(processes=10)
        results = pool.map(SFf, range(len(ThrX)))
        pool.close()
        pool.join()
        Showtime(Start)
        print datetime.datetime.now(), '\n'

        print 'All shapefactors were found.',
        Showtime(Start)
print datetime.datetime.now(), '\n'

if CalcSH == 'y':
        tempresults = []
        for q in results:
                tempresults.append(q)
        tempresults.sort()

for p in range(len(ThrIndex1)):
        if CalcSH == 'y':
                ThrArea.append(tempresults[p][1])
                ThrPeri.append(tempresults[p][2])

#find pores that are connected with multiple throats
print 'Finding Multi-connected pores...',
Showtime(Start)
print datetime.datetime.now(), '\n'

ThrIndices = []
for p in range(len(ThrIndex1)):
   ThrIndices.append([ThrIndex1[p],ThrIndex2[p]])
   ThrIndices[p].sort()


MultiConnectedIndices = []
seen = []
Counter = 0
for p in range(len(ThrIndices)):
        for k in range(len(ThrIndices)):
                if ThrIndices[p] == ThrIndices[k] and p != k:
                        if[p,k] not in MultiConnectedIndices and [k,p] not in MultiConnectedIndices:
                                MultiConnectedIndices.append([p,k])
print 'total amount of multi-connected pores:' ,len(MultiConnectedIndices)
print 'Multi-connected pores found.',
Showtime(Start)
print datetime.datetime.now(), '\n'
print 'Removing multi-connected pores...'
MultiPores = list(connected_components(MultiConnectedIndices))
```

```
#add up the area, perimeter and radius of the throats, and take the average length to merge the
pore bodies.
Removables = []
for q in range(len(MultiPores)):
        minRealThroatLength = RealThroatLength[MultiPores[q][0]]
        minRealThroatLength1 = RealThroatLength1[MultiPores[q][0]]
        minRealThroatLength2 = RealThroatLength2[MultiPores[q][0]]
        for r in range(1,len(MultiPores[q])):
                if CalcSH == 'y':
                        ThrArea[MultiPores[q][0]] += ThrArea[MultiPores[q][r]]
                        ThrPeri[MultiPores[q][0]] += ThrPeri[MultiPores[q][r]]
                if minRealThroatLength > RealThroatLength[MultiPores[q][r]]:
                        minRealThroatLength = RealThroatLength[MultiPores[q][r]]
                        minRealThroatLength1 = RealThroatLength1[MultiPores[q][r]]
                        minRealThroatLength2 = RealThroatLength2[MultiPores[q][r]]
                        ThrX[MultiPores[q][0]] = ThrX[MultiPores[q][r]]
                        ThrY[MultiPores[q][0]] = ThrZ[MultiPores[q][r]]
                        ThrZ[MultiPores[q][0]] = ThrZ[MultiPores[q][r]]
                ThrRadius[MultiPores[q][0]] += ThrRadius[MultiPores[q][r]]
                Removables.append(MultiPores[q][r])
        RealThroatLength[MultiPores[q][0]] = minRealThroatLength
        RealThroatLength1[MultiPores[q][0]] = minRealThroatLength1
        RealThroatLength2[MultiPores[q][0]] = minRealThroatLength2
Removables.sort()

#Remove multiconnected pores
ConfOne = []
ConfTwo = []
ThroatArea = []
ThroatPerimeter = []
ShapeFactor = []
ThroatRadius = []
LengthRatioFinal = []
RealThroatLengthFinal = []
RealThroatLength1Final = []
RealThroatLength2Final = []
ThrIndex1Final = []
ThrIndex2Final = []
ThrXFinal = []
ThrYFinal = []
ThrZFinal = []
CorNum = [0] * len(AllPB)
for p in range(len(ThrIndex1)):
   for q in range(len(Removables)):
      if p == Removables[q]:
         break
      elif p != Removables[q] and q == len(Removables)-1:
         ConfOne.append(ThrIndex1[p])
         ConfTwo.append(ThrIndex2[p])
         if CalcSH == 'y':
```

```
    ThroatArea.append(ThrArea[p])
    ThroatPerimeter.append(ThrPeri[p])
    ShapeFactor.append(ThrArea[p]/ThrPeri[p]**2.0)
   ThroatRadius.append(ThrRadius[p])
   CorNum[ThrIndex1[p]] += 1
   CorNum[ThrIndex2[p]] += 1
   LengthRatioFinal.append(LengthRatio[p])
   RealThroatLengthFinal.append(RealThroatLength[p])
   RealThroatLength1Final.append(RealThroatLength1[p])
   RealThroatLength2Final.append(RealThroatLength2[p])
   ThrIndex1Final.append(ThrIndex1[p])
   ThrIndex2Final.append(ThrIndex2[p])
   ThrXFinal.append(ThrX[p])
   ThrYFinal.append(ThrY[p])
   ThrZFinal.append(ThrZ[p])

if CalcSH == 'n':
    with open('Data/AREA.txt', 'r') as m:
        with open('Data/PERIMETER.txt', 'r') as n:
            with open('Data/SH.txt', 'r') as o:
                AreaList = m.readlines()
                PerimeterList = n.readlines()
                SHList = o.readlines()
    for p in range(len(AreaList)):
        ThroatArea.append(float(AreaList[p]))
        ThroatPerimeter.append(float(PerimeterList[p]))
        ShapeFactor.append(float(SHList[p]))

#Calculate the inradius using the shapefactor
ThroatInRadius = []
for p in range(len(ThroatRadius)):
    if ShapeFactor[p] < 0.048113:
        ThroatInRadius.append(float(ThroatArea[p]*2)/ThroatPerimeter[p])
    elif ShapeFactor[p] > 0.0625:
        ThroatInRadius.append(math.sqrt(float(ThroatArea[p])/math.pi))
    else:
        ThroatInRadius.append(0.5*math.sqrt(float(ThroatArea[p])))
ThroatCombo = []
for p in range(len(ThroatInRadius)):
    if ThroatRadius[p] > ThroatInRadius[p] and ThroatRadius[p] > 3.0:
        ThroatCombo.append(ThroatRadius[p])
    else:
        ThroatCombo.append(ThroatInRadius[p])

with open('Data/SH.txt','w') as TSF:
    with open('Data/PIPER.txt','w') as TR:
        with open  ('Data/AREA.txt', 'w') as TRA:
            with open  ('Data/PERIMETER.txt', 'w') as TRP:
                with open('Data/ThroatInradius.txt','w') as TIR:
                    for p in range(len(ConfOne)):
                        TSF.write(str(ShapeFactor[p]) + '\n')
```

```
                                                        TRA.write(str(ThroatArea[p]) + '\n')
                                                        TRP.write(str(ThroatPerimeter[p]) + '\n')

                                                        TR.write(str(ThroatCombo[p]) + '\n')
                                                        TIR.write(str(ThroatInRadius[p]) + '\n')


#Calculating pore body radius for aspect ratio
PBRSegment = []
for p in SegmentVolume:
        PBRSegment.append(((3.0*p)/(4.0*math.pi))**(1.0/3.0))

ThroatLength1 = []
ThroatLength2 = []
ThroatLength = []
for p in range(len(ThrIndex1Final)):
        L1 = RealThroatLength1Final[p]
        L2 = RealThroatLength2Final[p]
        if L1 <= 0:
                print L1
                L1 = 0.5
        if L2 <= 0:
                print L2
                L2 = 0.5
        k1 = PBRSegment[ThrIndex1Final[p]]/ThroatInRadius[p]
        k2 = PBRSegment[ThrIndex2Final[p]]/ThroatInRadius[p]
        if k1 < 1:
                k1 = 1.0
        if k2 < 1:
                k2 = 1.0
        Length1 = L1 * (1+k1+k1**2)/(3*k1**3)
        Length2 = L2 * (1+k2+k2**2)/(3*k2**3)
        ThroatLength1.append(Length1)
        ThroatLength2.append(Length2)
        ThroatLength.append(Length1+Length2)

#assign value to throats using the length and the inscribed sphere radius. This volume is subtracted
from the pore bodies it connects to
TotalVolume = sum(SegmentVolume)

for p in range(len(ThroatRadius)):
        SegmentVolume[ThrIndex1Final[p]] -= ThroatCombo[p]**2 * math.pi*ThroatLength[p]
        SegmentVolume[ThrIndex2Final[p]] -= ThroatCombo[p]**2 * math.pi*ThroatLength[p]

#Calculating PORER for PoreFlow
PBRSegment = []
counter = 0
for p in range(len(SegmentVolume)):
        if SegmentVolume[p] <= 0:
                SegmentVolume[p] = 1.0
                counter +=1
        PBRSegment.append(((3.0*SegmentVolume[p])/(4.0*math.pi))**(1.0/3.0))


print '\nTotal volume:', TotalVolume, '\nPore body volume:', sum(SegmentVolume), '\nThroat
volume:', TotalVolume-sum(SegmentVolume), '\nRatio pore body volume to total volume:',
sum(SegmentVolume)/TotalVolume , '\nSegmented volumes smaller than 0 =', counter, '\n'

print 'Multi-connected pores removed.'
print 'Writing output files...',
Showtime(Start)
print datetime.datetime.now(), '\n'
#write output files that could be used as input in PoreFlow
with open('Data/PORERinscr.txt', 'w') as PBRF:
        with open('Data/PORER.txt', 'w') as PORER:
                with open('Data/PORE_LOC.txt', 'w') as PBL:
                        with open('Data/CoordinationNumbers.txt','w') as CN:
                                with open('Data/PORE_INLET.txt', 'w') as INOUT:
                                        for p in range(len(PBR)):
                                                PORER.write(str(PBRSegment[p]) + '\n')
                                                CN.write(str(CorNum[p]) + '\n')
                                                INOUT.write(str(Boundaries[p]) + '\n')
                                                PBRF.write(str(PBR[p]) + '\n')
                                                PBL.write(str(AllPB[p]) + '\n')

with open('Data/CONF.txt', 'w') as Conf:
        with open('Data/PIPEL.txt'  ,'w') as TL:
                with open('Data/PIPERinscr.txt','w') as inscr:
                        for p in range(len(ConfOne)):
                                Conf.write(str(ConfOne[p]+1) + '\t' + str(ConfTwo[p]+1) + '\t1\n')
                                TL.write(str(ThroatLength[p]) + '\n')
                                inscr.write(str(ThroatRadius[p]) + '\n')

#Create vtk files with pore body throats
print'Creating vtk file...'
SkVTK = open('Data/Skeleton.vtk', 'w')
SkVTK.write('# vtk DataFile Version 2.0\nSkeletonoutput\nASCII\n\nDATASET
UNSTRUCTURED_GRID\n')
SkVTK.write('POINTS ' + str(len(AllPB)) +  ' float\n')
for p in AllPB:
   SkVTK.write(str(p)+'\n')

SkVTK.write('\nCELLS ' + str(len(ConfOne)) + ' ' +str(len(ConfOne)*3) + '\n')
for p in range (0, len(ConfOne)):
   SkVTK.write('2\t' + str(ConfOne[p]) + '    \t' + str(ConfTwo[p]) + '\n')

SkVTK.write('\nCELL_TYPES ' + str(len(ConfOne)) + '\n')
for p in range (0,len(ConfOne)):
   SkVTK.write('3\n')

SkVTK.write('\nPOINT_DATA '+ str(len(PBR)) +'\nSCALARS PoreBodyRadius float\nLOOKUP_TABLE
default \n')
```

```
for p in range (len(PBR)):
   SkVTK.write(str(PBR[p]) + '\n')

SkVTK.write('\nSCALARS VolumeEquivalentRadius float\nLOOKUP_TABLE default \n')
for p in range (len(PBRSegment)):
   SkVTK.write(str(PBRSegment[p]) + '\n')

SkVTK.write('\nSCALARS Boundaries float\nLOOKUP_TABLE default \n')
for p in range (len(Boundaries)):
   SkVTK.write(str(Boundaries[p]) + '\n')

SkVTK.write('\nSCALARS CoordinationNumbers float\nLOOKUP_TABLE default \n')
for p in range (len(CorNum)):
   SkVTK.write(str(CorNum[p]) + '\n')

SkVTK.write('\nCELL_DATA '+ str(len(ThroatRadius)) + '\nSCALARS ThrRadius float\nLOOKUP_TABLE
default \n')
for p in range (0,len(ThroatRadius)):
   SkVTK.write(str(ThroatRadius[p]) + '\n')

if CalcSH == 'y':
		SkVTK.write('\nSCALARS Shapefactor float\nLOOKUP_TABLE default \n')
		for p in range (0,len(ShapeFactor)):
			SkVTK.write(str(ShapeFactor[p]) + '\n')

		SkVTK.write('\nSCALARS ThroatArea float\nLOOKUP_TABLE default \n')
		for p in range (0,len(ThroatArea)):
			SkVTK.write(str(ThroatArea[p]) + '\n')

		SkVTK.write('\nSCALARS ThroatPerimeter float\nLOOKUP_TABLE default \n')
		for p in range (0,len(ThroatPerimeter)):
			SkVTK.write(str(ThroatPerimeter[p]) + '\n')
SkVTK.close()

End = time.time()
Elapsed = End - Start
print 'Finished. it took '+ str(round(Elapsed)) + 's to run'
print datetime.datetime.now(), '\n'
#Show Seal of Approval if everything went ok.
print '            .---.                        '
print '           /o   o\                       '
print '        __(=  "  =)__                    '
print '         //\`-=-`/\\\                    '
print '            )   (_                       '
print '           /      `==-._                 '
print '          /      \     ``==.             '
print '         /  /   \  \        `=..--._     '
print '     ___/  /     \  \___     _,  ,  \    '
print '    `-----`""""""""""`------`""""`\  \__/    '
print '                            `-`          '
print '             Seal of Approval            '
```