GENERATION AND SIMULATION OF FLOW AND MASS TRANSFER IN VIRTUAL POROUS MEDIA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AZIZ MOHAMMED

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CHEMICAL ENGINEERING

JULY 2023

Approval of the thesis:

GENERATION AND SIMULATION OF FLOW AND MASS TRANSFER IN VIRTUAL POROUS MEDIA

submitted by **AZIZ MOHAMMED** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Chemical Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Pınar Çalık
Head of the Department, **Chemical Engineering** _____

Assoc. Prof. Dr. Harun Koku
Supervisor, **Chemical Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Naime Aslı Sezgi
Chemical Engineering, METU _____

Assoc. Prof. Dr. Harun Koku
Chemical Engineering, METU _____

Prof. Dr. Göknur Bayram
Chemical Engineering, METU _____

Prof. Dr. Görkem Külah
Chemical Engineering, METU _____

Assoc. Prof. Selis Önel Kayran
Chemical Engineering, Hacettepe University _____

Date: 25.07.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name : Aziz Mohammed

Signature :

ABSTRACT

GENERATION AND SIMULATION OF FLOW AND MASS TRANSFER IN VIRTUAL POROUS MEDIA

Mohammed, Aziz
Master of Science, Chemical Engineering
Supervisor : Assoc. Prof. Dr. Harun Koku

July 2023, 108 pages

The problem of fluid flow and mass transport in porous media emerges in various fields in engineering and science where high-surface-area materials are required. This ranges from food and soil sciences to chemical, environmental and petroleum engineering. Porous materials used in numerous applications are characterized by distinct morphology and physiochemical properties, depending on their main objective. For example, highly ordered porous structures with monomodal particle size distribution are used as solid-phase catalysts for gas reactions. On the other hand, the stationary phase in high performance liquid chromatography (HPLC) is made of hierarchically structured porous media with ideally bimodal pore size distribution which offer macropores for convective mass transport and mesopores for diffusion-limited mass transfer. In research studies concerning the development of chromatographic stationary phases, the traditional approach to come up with revolutionary designs, which would achieve low plate heights (or band dispersion) at high permeabilities, is based on an experimental trial-and-error procedure, where the stationary phase is physically synthesized and examined in the laboratory to evaluate its performance, leading to cumulative expenses in material/chemicals as well as time. Furthermore, due to uncontrolled disturbances in operating conditions,

it is often extremely difficult to completely single out certain parameters for optimization. The goal of this study is to develop a three-step strategy to model dispersion in virtually constructed geometries. The developed model is expected to, at least, serve as a directional guide to experimental research in stationary phase development by leveraging the recent advents in computational power available today. Because it is impossible to perform such a transformation in an absolute sense, some limitations and assumptions are present and discussed accordingly. The results of this work were found to be similar to other experimental and computational works in the literature, despite some minor discrepancies. Such discrepancies were addressed and justified mainly by attributing them to variations in the morphology of the porous media investigated by each study.

Keywords: Liquid chromatography, Dispersion, Stochastic simulations

ÖZ

SANAL GÖZENEKLİ ORTAMLARDA AKIŞ VE KÜTLE TRANSFERİNİN ÜRETİLMESİ VE SİMÜLASYONU

Mohammed, Aziz
Yüksek Lisans, Kimya Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Harun Koku

Temmuz 2023, 108 sayfa

Gözenekli ortamlarda sıvı akışı ve kütle taşınımı sorunu, yüksek yüzey alanlı malzemelerin gerektiği mühendislik ve bilim alanlarında çeşitli şekillerde ortaya çıkar. Bu, gıda ve toprak bilimlerinden kimya, çevre ve petrol mühendisliğine kadar çeşitli alanları kapsar. Birçok uygulamada kullanılan gözenekli malzemeler, temel amaca bağlı olarak farklı morfolojik ve fizikokimyasal özelliklere sahiptir. Örneğin, gaz reaksiyonları için katı faz katalizörleri olarak kullanılan düzenli gözenekli yapılar, tek modal parçacık boyutu dağılımına sahiptir. Diğer taraftan, yüksek performanslı sıvı kromatografisinde (HPLC) kullanılan durağan faz, konvektif kütle taşınması için makro gözenekler ve difüzyon sınırlı kütle transferi için mezo gözenekler sunan ideal olarak ikili modal gözenek boyutu dağılımına sahip hiyerarşik yapıya sahip gözenekli ortamlardan oluşur. Kromatografik durağan fazların geliştirilmesine yönelik araştırmalarda, yüksek geçirgenliklerde düşük plaka yüksekliklerine (veya bant yayılımına) sahip devrim niteliğinde tasarımların elde edilmesi için geleneksel bir yaklaşım izlenir. Bu yaklaşım, durağan fazın fiziksel olarak sentezlenip laboratuvarda performansının değerlendirildiği deneysel bir deneme yanılma prosedürüne dayanır ve malzeme/kimyasallar ve zaman açısından kümülatif maliyetlere neden olur. Ayrıca, işletme koşullarında kontrolsüz

değişiklikler olduğu için optimizasyon için belirli parametreleri tamamen ayırmak genellikle son derece zordur. Bu çalışmanın amacı, sanal olarak oluşturulan geometrilerde dağılımı modellemek için üç adımlı bir strateji geliştirmektir. Geliştirilen modelin, en azından günümüzde mevcut olan hesaplama gücünü kullanarak durağan faz geliştirme konusundaki deneysel araştırmalara yön gösteren bir rehber olarak hizmet etmesi beklenmektedir. Bu tür bir dönüşümü mutlak anlamda gerçekleştirmek mümkün olmadığından, bazı sınırlamalar ve varsayımlar bulunmakta ve buna göre tartışılmaktadır. Bu çalışmanın sonuçları, literatürdeki diğer deneysel ve hesaplamalı çalışmalarla uyumlu olmasına rağmen bazı küçük farklılıklar içermektedir. Bu farklılıklar, her bir çalışmanın incelediği gözenekli ortamın morfolojik özelliklerindeki farklılıklara bağlı olarak ele alınmış ve gerekçelendirilmiştir.

Anahtar Kelimeler: Sıvı kromatografisi, Dispersiyon, Stokastik Simülasyonlar

To family and friends,

ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to my supervisor, Assoc. Prof. Dr. Harun Koku. His invaluable guidance, expertise, and contant support have been instrumental in shaping this thesis. His insightful feedback and constant encouragement have pushed me to reach new heights in my research and academic pursuits.

I would also like to express my gratitude to the faculty members in the Chemical Engineering Department at METU, whose exceptional expertise, dedication, and passion for teaching have expanded my knowledge and enriched my understanding of advanced concepts. Their commitment to academic excellence and adeptness in conveying complex ideas have profoundly influenced my intellectual development.

In addition, I would like to extend my warmest thanks to my friends, Abdolrahim Alomaisy, Mohammed Bamatraf, Garallah Mohammed, Ala Alsuhile, and Rashad Abdullah. Their encouragement, shared experiences, and empathetic presence have been a constant source of strength and motivation, helping me overcome challenges and stay focused on my goals.

Finally, I convey my heartfelt appreciation to my family, who have been a constant pillar of support despite the physical distance. Their love, understanding, and belief in my abilities have fueled my determination to pursue my academic aspirations.

I would also like to acknowledge that this work was partially funded by the Scientific and Technological Research Council of Turkey under grant number TÜBİTAK 217M524.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

HETP	Height Equivalent to a Theoretical Plate
HPLC	High Performance Liquid Chromatography
LBM	Lattice Boltzmann Method
MPI	Message Passing Interface
RWPT	Random Walk Particle Tracking
STL	Standard Tessellation Language
SDE	Stochastic Differential Equation

LIST OF SYMBOLS

SYMBOLS

C_D	Unit conversion factor for diffusion and dispersion (m^2/s)
C_L	Unit conversion factor for length (m)
C_P	Unit conversion factor for pressure (Pa)
C_t	Unit conversion factor for time (s)
C_u	Unit conversion factor for velocity (m^2/s)
D_{ax}^*	Normalized dispersion coefficient
D_{ax}	Dispersion coefficient (m^2/s)
d_e	Equivalent particle size (m)
D_{ij}	Spatial distance between particle i and particle j (pixels)
D_m	Diffusion coefficient of the analyte in the mobile phase (m^2/s)
h	Reduced plate height
k	Permeability (m^2)
L	Column length (m)
M_{loss}	Total mass loss due to overlap of the growing particles and partially crossing the system bounds (voxels)
N_{mon}	Number of initial monomers (particles) in the sphere-based geometry construction simulation
Pe	Dimensionless Péclet number
$\mathcal{P}_s$	The solid-phase subdomain of the porous media
Re	Dimensionless Reynolds number
r_p	Radius of the spherical particle in the sphere-based geometry construction simulation
u_{avg}^x	Axial velocity averaged over the mobile phase (m/s)
u_f	Velocity field scaling factor
u_{max}	Axial maximum velocity (m/s)

u_x	Mobile phase velocity in the axial direction (m/s)
$V_{lattice}$	Volume of the 3D simulation box (lattice) in voxels
V_{solid}	Volume of the solid phase in the sphere-based geometry construction simulation (voxels)
W	The stochastic component of the diffusion term in the random walk equation
x	Axial coordinate and direction of the flow (m)
α_{ij}	Relative difference in coordinate values between coordinate i and coordinate j
ΔP	Applied pressure difference between entrance and exit planes (Pa)
ρ_{ex}	Extrinsic porosity or target porosity in the sphere-based geometry construction simulation
ρ_{int}	Extrinsic porosity or target porosity in the sphere-based geometry construction simulation
σ	Particle growth factor as a fraction of the particle radius
σ_t^2	Time-based variance of the tracers' exit time distribution (s^2)
σ_x^2	Position-based variance of the axial distribution of tracer positions (m^2)
ϵ	Total void fraction of the porous media
μ	Mobile phase dynamic viscosity ($Pa \cdot s$)
ν	Mobile phase kinematic viscosity (m^2/s)

CHAPTER 1

INTRODUCTION

1.1 Brief Description of the Chromatography Theory

Chromatography is an analytical technique that was first described by Mikhail Tsvet, the inventor of chromatography, in 1905 in the first ever publication in this field (Цвет, 1903). In the majority of chromatography applications, the separation is achieved by leveraging the differences in affinities of analyte molecules to a fixed stationary phase, which ideally creates distinct bands that travel through the stationary phase at different velocities. Most of the modern applications of analytical liquid chromatography are classified under HPLC (High Performance/Pressure Liquid Chromatography), where advanced instruments are used to maintain a pressure-driven flow of the mobile phase through the stationary phase.

The theory of chromatography is inherently complicated such that "chemists may tend to shy away from" (Neue, 1997). The basic working principle of a chromatography process can be explained by considering a small amount of analyte sample injected into the top of a column, where the first narrow band is formed. The analyte band travels through the column due to the imposed pressure gradient (as in HPLC), and the band broadens out, as depicted in Figure 1.1, where, hypothetically, a mixture of two components was injected, and as a result of having different affinities, the components travel at different rates (from left to right in Figure 1.1), leading to the formation of distinct bands that exit the column at different times;

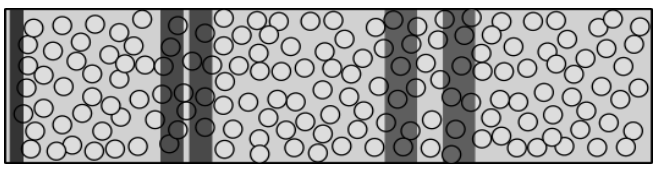


Figure 1.1. Schematic of band broadening and separation in chromatography.

hence the separation is achieved. Towards a quantitative description of the theory, the band can be conceived as the statistical distribution of analyte molecules along the axial direction. The second moment of the distribution, the variance σ^2, is directly proportional to the width of the band by definition. The slope of the plot of σ^2 with distance down the column is the formal definition of the chromatographic version of the Height Equivalent to Theoretical Plate (HETP), which is a constant given that the variance-distance relationship is linear, as is the case for most chromatographic stationary phases. The HETP is a fundamental parameter to quantify separation quality in chromatography: A low value of HETP means the speed of band-broadening is low, which in turns means that the distance between the bands that correspond to different components in the analyte sample is large, ultimately allowing for high-resolution separation. Other measures of separation quality in chromatography include resolution, plate count, column efficiency, and peak symmetry. These concepts are described in (Neue, 1997).

1.1.1 Factors Controlling Separation Quality

The dynamics of chromatography are described in terms of the relationship between HETP and linear velocity representing the flow, the properties of the analyte sample, and the properties of the chromatographic column as the stationary phase. The most fundamental equation that relates HETP to the axial linear velocity, u_x, is the van Deemter equation (Van Deemter *et al.*, 1956)

$$\mathrm{HETP} = \mathrm{A} + \frac{\mathrm{B}}{u_x} + \mathrm{C}u_x \tag{1.1}$$

which is visualized in Figure 1.2 with arbitrary values of the fitting parameters A, B, and C. The three terms add up to constitute the van Deemter curve (black continuous line in Figure 1.2) that has an absolute minimum and behaves linearly at large linear velocities. This curve was also obtained by other models of plate height dependence on mobile phase linear velocity (Bristow & Knox, 1977; J Calvin

Giddings, 1965). The term linear velocity (u_x in Equation (1.1)) is dimensionless and is equivalent to the Péclet number defined by

$$Pe = \frac{u_{avg}^{x} d_e}{D_m} \tag{1.2}$$

where $u_{avg}^{x}$is the average axial velocity of the mobile phase, d_e is the equivalent particle size, and D_m is the bulk diffusion coefficient of the analyte in the mobile phase. The three terms appearing in the van Deemter equation represent three independent classes of factors controlling separation quality. The first term, the A term, is independent of the flow dynamics and represents the factors related to the stationary phase such as pore size distribution and structural nonuniformities. The B term represents axial molecular diffusion and it is inversely proportional to the velocity. The third term, the C term, comprises all mass transfer contributions and it is directly proportional to the velocity. Since the desired objective of a chromatography set-up is to obtain low HETP at high velocities, the C term has received extensive attention from researchers, and for the same reason, the B term is neglected in most practical applications. A detailed description of the effects and interactions of different factors in the HETP curve is available in (J. Caluin Giddings, 2017; Neue, 1997).

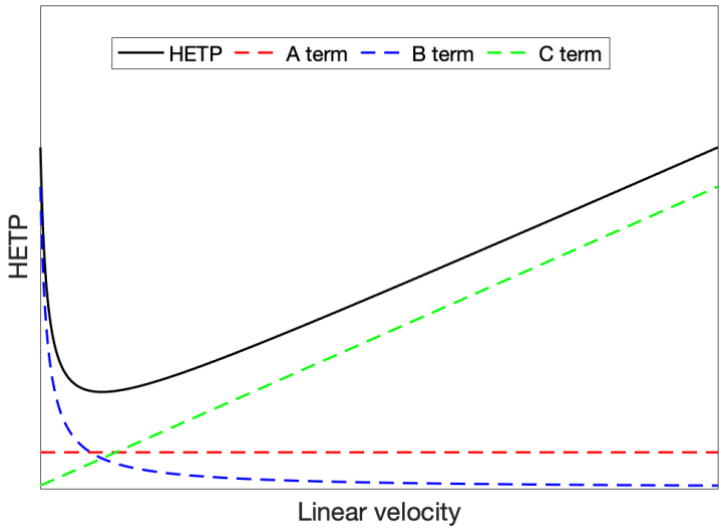


Figure 1.2. Visualization of the van Deemter equation.

Dispersion, another measure that is commonly used for evaluating the separation quality, is related to the reduced plate height, h, by

$$D_{ax}^{*} = \frac{h}{2} Pe \tag{1.3}$$

where D_{ax}^{*} is the dispersion coefficient normalized by molecular diffusion, h is the plate height normalized by a characteristic length, such as the equivalent particle size of the chromatographic bed, and Pe is the Péclet number defined in Equation (1.2).

1.2 Motivation and Objective of the Study

In most research studies in chromatography, the major goal is to achieve small dispersion coefficients (or HETP) at high mobile phase velocities. The traditional approach in the development of revolutionary separation materials that would achieve that goal is based on a trial-and-error procedure, where the stationary phase is physically synthesized and examined in the laboratory to evaluate its performance, leading to cumulative expenses in material/chemicals as well as time. Furthermore, due to uncontrolled disturbances in operating conditions, it is often extremely difficult to completely single out certain parameters for optimization.

This work attempts to achieve a transformation of this trial-and-error procedure from the physical domain to the virtual domain by leveraging the computational power available today, leading to a significant reduction in operational costs associated with laboratory experiments, as well as providing a study environment that is strictly controlled, except for the extremely rare events that might pop up from the quantum world. Because it is impossible to perform such a transformation in an absolute sense, some limitations are present and documented, and some assumptions were made as discussed in later chapters. The goal is to develop a three-step strategy to model dispersion in virtually constructed geometries, that should at least serve as a directional guide to experimental research in stationary phase development.

The first step is the construction of geometries with a high degree of adjustability of the macroscopic properties such as total porosity, particle size distribution, and particle shapes. The second stage is the simulation of flow in the constructed geometries using an appropriate numerical method; the Lattice Boltzmann method (LBM) was used. The third step is the coupling of the LBM velocity field with Brownian diffusion to simulate dispersion while accounting for the micro-structural details; this is achieved by implementing the Random Walk Particle Tracking (RWPT) method. The three steps are summarized in Figure 1.3.

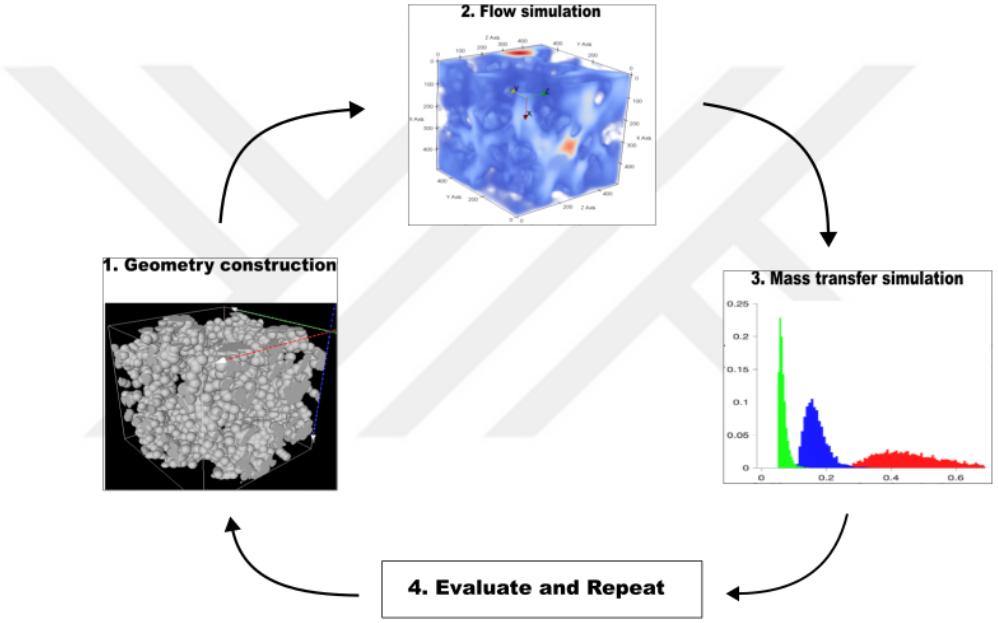


Figure 1.3. Summary of the objective of this study; a three-step strategy to model dispersion in the virtual space.

1.3 Organization of the Thesis

This thesis is organized as follows. In Chapter 2, a detailed coverage of relevant literature studies is provided as well as a dedicated body of text to cover the theoretical basis of the methods and approaches used in this work.

Chapter 3 provides a detailed description of the numerical methods and their implementation and is separated into three sections for each of the three stages shown in Figure 1.3. The methods are described in terms of mathematical equations exactly as implemented in the codes. The first section describes two algorithms to generate two types of virtual porous media: the pixel-based algorithm and the sphere-based algorithm. As the names suggest, the pixel-based algorithm creates geometries with hydraulic particle sizes in the length scale of pixels, and the sphere-based algorithm generates geometries with spherical particles connected as a single cluster with a much larger length scale compared to those generated by the pixel-based algorithm. The two resulting structures were merged to form a heterogeneous porous media with a bimodal pore size distribution. The second and third sections in Chapter 3 describe the implementation of the Lattice Boltzmann and the random walk methods, respectively.

Chapter 4 presents the results of this study and it is also divided into three sections for all of the three steps. First, the constructed geometries are presented and visualized. The tunability of the geometry-making model is discussed in such a way that different geometries with different macroscopic properties can be generated and analysed by following the same three-step strategy. The second and third sections present the results of flow and mass transfer simulations in terms of velocity distributions, 3D visualizations of the flow field, and dispersion vs. Péclet number plots. Both flow and mass transfer simulations were performed on two geometries and two boundary condition modes as shown in Table 1.1.

Table 1.1 The four system configurations simulated in this study.

System description	**ID**
Rigid spheres with bounce-back B.C.	1
Rigid spheres with periodic B.C.	2
Permeable spheres with bounce-back B.C.	3
Permeable Spheres with periodic B.C.	4

The first geometry is a bimodal structure of spherical particles, and the second geometry is the same geometry except that the spherical particles were made permeable as opposed to being impermeable initially. The modes for boundary conditions are bounce-back and periodic boundary conditions, concerning the behavior of the mobile phase at the lateral walls. The system boundaries that are perpendicular to the flow direction were always subject to a pressure difference boundary condition. Moreover, the results of this work were extensively compared and discussed with similar studies in the literature.

The conclusions, model limitations as well as recommendations for future works and possible workarounds are discussed in Chapter 5 and 6. Finally, algorithm flow charts, code scripts, and additional supporting information are given in the Appendices.

CHAPTER 2

LITERATURE SURVEY

2.1 Dispersion in Chromatography

In the previous chapter, the theory of chromatography was briefly discussed and the concept of plate height, HETP, was introduced as the slope of the variance of the axial position distribution of the analyte sample, σ_x^2, as it changes with distance down the column, as expressed in Equation (2.1).

$$H = \frac{d\sigma_x^2}{dx} \tag{2.1}$$

Throughout the body of this text, x is consistently used to denote the axial/longitudinal coordinate, or the direction of the flow. It is common in many separation processes to define a dimensionless (reduced) plate height, h, as the plate height normalized by a characteristic length. In chromatography, this characteristic length is the equivalent particle diameter of the porous bed, d_e. This relationship is given in Equation (2.2).

$$h = \frac{H}{d_e} \tag{2.2}$$

The axial dispersion coefficient D_{ax} in this context is related to the plate height by the mobile phase axial velocity,

$$D_{ax} = \frac{H}{2} u_x \tag{2.3}$$

Similarly, the dimensionless dispersion coefficient is defined by introducing the bulk diffusion coefficient of analyte molecules, D_m. Therefore, the dimensionless relationship between plate height and dispersion takes the form given in Equation (2.4).

$$D_{ax}^{*} = \frac{h}{2} Pe \tag{2.4}$$

The normalized dispersion D_{ax}^{*} can be expressed in terms of the position-based variance σ_x^2 as half the slope of the plot of variance with time, rather than the distance down the column.

$$D_{ax}^{*} = \frac{1}{2} \frac{d\sigma_x^2}{dt} \tag{2.5}$$

Practically, it is easier to estimate dispersion from the time-based variance σ_t^2 rather than the position-based variance σ_x^2 that appears in Equation (2.5). In the numerical simulations of this study, the dispersion coefficient was calculated by considering the variance of the exit time distribution of the random walk tracers, σ_t^2. First, an exit point was defined as the column length, and whenever a tracer particle was detected to have passed the set exit point, its exit time was recorded and stored. When all tracers have exited the column or the maximum number of iterations was reached, an exit time one-dimensional array is obtained, for which the variance, σ_t^2, is calculated. Dispersion is then calculated by Equation (2.6).

$$D_{ax}^{*} = \frac{1}{2} \frac{u_x L_c}{D_m} \frac{\sigma_t^2}{t_R^2} \tag{2.6}$$

where t_R is the average retention time, and L_c is the column length. The reduced plate height can then be calculated by

$$h = \frac{2D_{ax}^{*}}{Pe} \tag{2.7}$$

In similar numerical studies in the literature concerning mass transfer in porous media, Equation (2.6) was used to calculate the dispersion coefficient (Koku *et al.*, 2012b; Maier *et al.*, 2000a).

2.1.1 Asymptotic Behavior of Dispersion

In most studies, especially numerical simulations, of mass transfer in chromatographic stationary phases, dispersion is often estimated in its transient (time-dependent) state, which means that it is dependent on the choice of the column length or the exit point to record particles' exit times. To analyze the steady-state, or asymptotic, dispersion, dedicated studies are usually required for this purpose, concerning the long-time solution of the problem (Hulin, 1994; Koch & Brady, 1987). The term asymptotic dispersion refers to a state where band broadening does not effectively increase with distance down the column, or equivalently with the amount of time the analyte particles spend in the column.

To reach this state, the analyte molecules (or the majority of the band) need to sample all preferential paths in such a way that they become sufficiently *familiar* with the structural complexities present in the geometry, and when this occurs, the width of the band becomes relatively constant with time; therefore, the more nonuniformities present in the system, the more difficult (i.e. time-consuming) it becomes to reach asymptotic behavior. Another way to think about this, in a more abstract sense, is by comprehending the band broadening phenomenon, at relatively high Péclet numbers, to be the consequence of the band absorbing the microscopic heterogeneities in the system, such that the more structural complexities left for the band to discover, the higher the potential for the band to grow wider.

As an illustration of this concept from a practical perspective, Figure 2.1 shows the results of an experimental study to evaluate the asymptotic behavior of dispersion on random packings of porous and nonporous spheres (Kandhai *et al.*, 2002). The experiments were performed at the same Péclet number and showed that asymptotic dispersion for porous spheres was reached after a much longer time than that for nonporous spheres. In a random sphere packing with nonporous or impermeable particles, dispersion is mainly dominated by the velocity field fluctuations in the medium and/or by the no-slip boundary condition at the fluid-solid interface (Charlaix *et al.*, 1987; Koch & Brady, 1985). However, in the case of porous

particles, there is an additional contribution to dispersion due to the existence of stagnant zones in the intraparticle pore space (J Calvin Giddings *et al.*, 1965; Sahimi, 2011; Salles *et al.*, 1993). This adds a significant amount of structural complexity that increases dispersion and enhances non-asymptotic behavior, as evident by the experimental results shown in Figure 2.1.

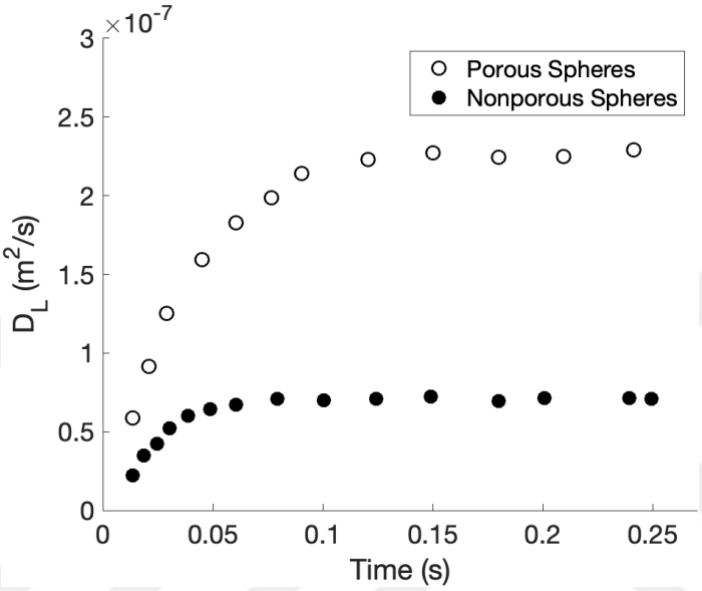


Figure 2.1. Experimental results on transient and asymptotic longitudinal dispersion in random sphere packings, regenerated from (Kandhai *et al.*, 2002).

As discussed so far, the dispersion phenomenon in chromatography has received rigorous attention both from experimental and modeling-based research studies. Many of these studies are focused on the development of revolutionary materials for the stationary phase with the ideal stationary phase being that which achieves low dispersion, or plate height, at high flow rates, or permeability. Since the focus of this work is to develop a numerical framework for generating hypothetical stationary phase geometries with controlled properties and to model and evaluate their performance in terms of dispersion coefficient, the upcoming section provides a brief review on chromatographic stationary phases. For recent trends and breakthroughs in monolithic columns for chromatography, a comprehensive review is given in (Svec & Lv, 2015).

2.2 Chromatographic Stationary Phases

The problem of fluid flow and mass transport in porous media emerges in various fields of engineering and science where high-surface-area materials are required. This ranges from food and soil sciences to chemical, environmental, and petroleum engineering (Bear, 1972; Gelhar *et al.*, n.d.; Guiochon *et al.*, 2006; Sahimi, 1993; Zahedzadeh *et al.*, 2010). Porous materials used in numerous applications are characterized by distinct morphology and physiochemical properties, depending on their main objective.

For example, highly ordered porous structures with monomodal particle size distribution are used as solid-phase catalysts for gas reactions. On the other hand, the stationary phase in high-performance liquid chromatography (HPLC) is made of hierarchically structured porous media with ideally bimodal pore size distribution which offers macropores for convective mass transport and mesopores (or micropores) for diffusion-limited mass transfer (Hlushkou, Bruns, & Tallarek, 2010). In HPLC, the amount of applied pressure to drive the mobile phase through the stationary phase is always limited by operational costs and/or mechanical stability of the chromatographic equipment; for example, a pressure of 10,000 bar marks the upper bound to what can be used in any practical chromatographic setup (Guiochon, 2007a). Therefore, depending on the desired application, a compromise must be made between the time of analysis and the quality of the separation. The limits of this trade-off between permeability and efficiency, considering maximum operating pressures, are well defined (J Calvin Giddings, 1965; J H_ Knox & Saleem, 1969).

The use of elevated temperatures to enhance mobile phase viscosity and, hence, achieve the same resolution at higher flowrates has been investigated with considerable progress (X. Wang, Barber, *et al.*, 2006; X. Wang, Stoll, *et al.*, 2006; Yang *et al.*, 2005). In addition to the use of high operating temperatures, other approaches and alternatives to increase the analysis speed have been studied in the literature (Guiochon, 2006). In the early periods in the history of chromatography,

packed beds of fine particles were used as the stationary phase (Senchenkova, 2003; TSWETT, 1954). What is interesting about packed beds is that their porosities come always near 0.4, no matter the size of the particles, provided that their sphericity is close to unity, and that their particle size distribution is also narrow (Bird *et al.*, 1960). Therefore, to considerably enhance the permeability of a packed bed, the number of packing defects should be increased, which in turn reduces the efficiency of the separation (Skudas *et al.*, 2009). Monolithic (i.e. continuous) beds provide just the solution to this problem.

A monolithic bed, namely; a monolith is realized as a continuous solid-phase block with large macropores to allow for undisrupted bulk flow (Koku *et al.*, 2011). The key feature of monolithic beds is that the solid phase is continuous with large macropores and small highly restricted mesopores, which offer relatively higher permeabilities compared to packed beds, as well as access to a larger surface area of the intra-skeleton pore space. Monolithic column efficiency is then controlled by adjusting the sizes of both the macropores and the mesopores, which opens a whole new scope of design prospects of stationary phases. The process of empirical optimization relies on the synthesis and testing of many combinations of inter- and intra-porosities, size distributions, and retention patterns, provided there is enough control during the preparation of the monoliths.

Monoliths are classified into two categories: Silica-based (Minakuchi *et al.*, 1997, 1998; Tanaka *et al.*, 2002) and organic polymer monoliths (Peters *et al.*, 1997; Svec & Frechet, 1996; Svec & Fréchet, 1992; Q. C. Wang *et al.*, 1994; Xie *et al.*, 1999), where the differences between the two types are related directly to their respective preparation procedure (Nakanishi & Soga, 1991, 1992b, 1992a; Sinner & Buchmeiser, 2000). Silica-based monoliths allow for creating hierarchically structured pore space for which all skeleton parameters can be independently adjusted (Leinweber & Tallarek, 2003), and they are used in partitioning chromatography for substances of low to medium molecular mass (Cabrera *et al.*, 2000; Lubda *et al.*, 2001; Tanaka *et al.*, 2000), while polymeric monoliths are mostly applied in the purification of biomacromolecules like plasmid-DNA and proteins

(Josić & Štrancar, 1999; Mihelič *et al.*, 2000; Vodopivec *et al.*, 2000). For both silica-based and polymeric monoliths, one restriction is their proper attachment to the wall of the column, resulting in a recommended upper limit for the operating pressure: 1 – 5 MPa for polymeric monoliths and up to 20 MPa for the silica-based monoliths (Leinweber & Tallarek, 2003).

2.3 Preparation of Silica Monoliths

The synthesis procedure of continuous silica-based porous media with a bimodal PSD (Pore Size Distribution) was first introduced by Nakanishi and Soga in the early 1990s when they came up with the preparation steps that allowed for independent control of the two pore size distributions; corresponding to the intra and inter-skeleton porous domains (Nakanishi & Soga, 1991, 1992a, 1992b). The Nakanishi and Soga procedure was later patented in the US, Japan, and Germany (Lubda & Mueller, 2007; Nakanishi & Soga, 1997). The procedure starts with the preparation of a starting solution of silane (or a mixture containing silane) as the reagent, and a porogen (inert pore-generating compound such as PEG, polyethylene glycol). Then the mixture is subjected to a hydrolysis exothermic reaction which leads to the formation of a precipitate, a monolithic insoluble mass with large pores. The concentration of silane and the porogen compound during the preparation of the starting mixture is the controlling factor that determines the pore size distributions. A detailed description of the production procedure of silica monoliths is given in (Guiochon, 2007b).

2.4 Virtual Construction of Porous Media

The problem of reconstruction of porous media lies in the development the porous lattice structure virtually while preserving the statistical properties of the physical one, mainly for numerical simulation purposes. Yeong and Torquato made use of two-point correlation functions to reconstruct 1-D and 2-D random media based on

minimizing the difference concerning a reference function (Yeong & Torquato, 1998). This method has not been investigated for the reconstruction of 3-D random media because it is inherently computationally expensive. For analyzing the effect of coordination number distribution on transport, the pore-network model is commonly used to regenerate real sandstone samples (Raoof & Majid Hassanizadeh, 2010). A pseudo-crystallization approach was followed to generate virtual porous media that take a void fraction and particle size distribution as input parameters (Rahmanian & Kantzas, 2018).

2.5 Modeling of Transport in Monolithic Stationary Phases

To evaluate the performance of chromatographic monoliths, numerous studies have been conducted targeting enhanced flow permeability and column efficiency. A fundamental equation that accurately captures the experimental observation between the applied pressure drop per unit length, $\frac{\Delta P}{L}$, and the mobile-phase area-averaged velocity u_{avg}^{x} is the Darcy equation expressed as

$$u_{avg}^{x} = \frac{k}{\mu}\frac{\Delta P}{L} \tag{2.8}$$

where k is the hydraulic permeability and μ is the dynamic viscosity. Commercial silica monoliths commonly have an absolute permeability in the order of 0.01 μm^2 (Guiochon, 2007b). On the other hand, for polymeric monoliths typical absolute permeability values may vary widely depending on the broad range of possibilities for synthesis formulations (Moravcová *et al.*, 2004).

The most common assessment parameter for monolithic column efficiency is the height equivalent to a theoretical plate (HETP), which is the combined result of lateral or axial mobile-phase diffusion, interfacial adsorption phenomena of solute molecules, and flow pattern-induced dispersion (J Calvin Giddings *et al.*, 1965; Horvath & Lin, 1976; John H Knox, 1977; Van Deemter *et al.*, 1956). The

fundamental plate height equation developed by van Deemter is expressed as (Van Deemter *et al.*, 1956)

$$h = A + \frac{B}{Pe} + C \cdot Pe \tag{2.9}$$

for which h is the reduced plate height, and Pe is the Péclet number

$$Pe = \frac{u_{avg}^{x} d_e}{D_m} \tag{2.10}$$

where D_m is the molecular mobile-phase diffusion coefficient, u_{avg}^{x} is the superficial average velocity of the mobile phase, and d_e is the characteristic equivalent particle size. The constants A, B and C weigh, respectively, the contribution of convective transport, molecular diffusion, and interfacial adsorption of the solute. Experimental curves of the van Deemter equation for silica and polymer monoliths show that the contribution of the C-term (corresponding to interface adsorption-desorption kineticss) to the overall plate height is small, and hence neglected, compared to the mechanical dispersion term A. The van Deemter equation shown above and its extensions such as the Knox equation (Bristow & Knox, 1977) remain useful in the qualitative evaluation of monolith performance, especially when the aim is to compare different kinds of columns or different solute materials. Since the parameters A, B and C are laid out in the van Deemter equation as fitting parameters, its predictive capability stays limited. More powerful approaches include the coupling model devised by Giddings (J Calvin Giddings *et al.*, 1965) which is of a microscopic scale and accounts for the underlying phenomena of adsorption, convection, and mass transfer. But such models are based on several simplifying assumptions about the system geometry and flow field and comprise parameters that must be assumed or estimated by experiment (Koku *et al.*, 2012b).

Another approach for modelling mass transport in porous media is the general rate model (GRM) which provides the simultaneous solution of the PDEs that describe the fraction of the mobile phase that percolates through the bed and the fraction that

constitutes the liquid hold-up in the mesoporous space. Moreover, the GRM accounts for mass transfer kinetics of adsorption-desorption in the fluid-solid interface.

Meyer and Liapis followed a more rigorous approach as they included some amount of geometrical information combined with the GRM in their modeling of the microscopic interactions of flow with monolith geometry, by introducing a representative network of pores as a cubic lattice (Liapis *et al.*, 1999; Meyers & Liapis, 1999). A general rate model with a simplified representation of the geometry of a unit skeleton was applied by Miyabe and coworkers to reach accurate plate height expressions and used experimental results to estimate other remaining parameters (Miyabe *et al.*, 2003; Miyabe & Guiochon, 2002), however, such methods that start from a complete set of transport equations like the general rate model remain unsolvable analytically, and their corresponding numerical solution techniques introduce even more complexity (Czok & Guiochon, 1990; Guiochon *et al.*, 2006).

To make these rigorous approaches realizable certain simplifications are usually made such as introducing a constant axial dispersion coefficient to be obtained experimentally and neglecting the effects of some physical phenomena in the governing transport equations. Although the Darcy equation is fairly adequate at predicting the pressure drop effect on material flowrate, one can hardly rely on the attempts to relate permeability to morphology like porosity and mean particle size (Mihelič *et al.*, 2005; Vervoort *et al.*, 2005) since they often are based on empirical data such as the Kozeny-Carmen equation, which is at first derived for random-close packed beds of nearly spherical particles with a narrow pore size distribution and porosity of about 0.4 (Carman, 1937).

2.6 Particle-based Modeling of Transport in Porous Media

When dealing with the complex geometries of chromatographic monoliths (and porous media in general) starting from the microstructural details not only promises

to produce an accurate description of the macroscopic variables but also accounts for the effect of the geometry on phenomena that are medium-specific. For example, solute entrapment leads to an inverse relation of exit peak area with the flow rate in polymeric monoliths with virus pulse injection (Trilisky & Lenhoff, 2009).

The Lattice Boltzmann (LB) method is a numerical technique for modelling transport phenomena in complex porous structures at low Reynolds numbers (Chen & Doolen, 1998; He & Luo, 1997; Junk & Klar, 2000; Succi, 2001). Previously, the technique has been used to solve for the microstructure-based velocity field in sandstone, asphalt, and silicon carbide matrices (Bernsdorf *et al.*, 2000; Kutay *et al.*, 2006; Manwart *et al.*, 2002). Moreover, LB-based solutions of flow and mass transfer in porous and non-porous spheres have been compared, with excellent agreement, to results obtained from NMR velocimetry (Kandhai, 2002; Manz *et al.*, 1999).

The random walk method is based on the theory of Brownian motion (Einstein, 1905; Langevin, 1908). The method accounts for the details of the microstructure and provides an approach that is intuitive to model mass transfer in irregular geometries, which can be coupled with the appropriate flow solution such as that obtained by the Lattice Boltzmann method. It is important to note that this approach follows a Markovian nature where the particle's position depends only on its position in the previous time step, and not on the past trajectory before the previous step. The random walk approach has been applied to model mass transfer in several studies (Hlushkou, Bruns, Ho, *et al.*, 2010; Kandhai, 2002; Koku *et al.*, 2012a; Langford *et al.*, 2006; Maier *et al.*, 2000b, 2003; Salles *et al.*, 1993; Tallarek *et al.*, 2019).

The basis of the random walk method is a stochastic differential equation (SDE) with the particle's position being the dependent variable

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{dt}} = \mathrm{a}(\mathbf{x}, \mathrm{t}) + \mathrm{b}(\mathbf{x}, \mathrm{t})\ \mathrm{dG} \tag{2.11}$$

where $\mathbf{x}$ is the particle's position vector, a and b are the advection (drift) and diffusion terms, respectively, and dG is a time-dependent stochastic function based on the Brownian motion. For a numerical simulation with a discrete space, Equation

(2.11) is discretized to obtain the appropriate integration of the SDE. The discretized form of the Equation (2.11) has been formulated (Gardiner, 1985; Risken & Risken, 1996) and is given in (2.12).

$$\mathbf{x}_t = \mathbf{x}_{t-1} + a \cdot \Delta t + b \cdot \Delta G \tag{2.12}$$

Here $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ indicate the coordinate position values at the current and previous time steps. Solving Equation (2.12) for a large number of particles where the particle positions denoted by the dependent variable $\mathbf{x}_t$ recovers the probabilistic solution of the Fokker-Planck Equation (FBE) with an ensemble, which is equivalent to solving the diffusion-convection equation (Gardiner, 1985; Ito *et al.*, 1951; Kinzelbach & Uffink, 1991; LaBolle *et al.*, 1996; Thomson & Montgomery, 1994).

The recent advances in computational power enabled pore-scale detailed analysis of complex geometries, and simulation methods that utilize this power based on detailed structural information are now verified and documented (Hlushkou, Bruns, Ho, *et al.*, 2010; Langford *et al.*, 2006; Maier *et al.*, 1998; Salles *et al.*, 1993). Koku and Lenhoff implemented a direct numerical simulation of flow and dispersion in a 3-dimensional reconstructed sample of a commercial polymeric monolith (Koku *et al.*, 2011, 2012b). They obtained the velocity field using the Lattice Boltzmann method which has proven to be more appropriate for modeling flow in irregular structures at low Reynolds number (Chen & Doolen, 1998; McNamara & Zanetti, 1988; Succi, 2001). For the simulation of dispersion, they used the random walk particle tracking method whereby a large number of particles (point particle and finite-size particles) are set in random diffusive motion within the reconstructed media and subjected to the LB-obtained flow field. The trajectories of the individual particles were subsequently analyzed for the estimation of dispersion. The random walk method, which is based on Einstein's mathematical description of Brownian motion, has been applied to solve a broad range of convection-diffusion problems (Kandhai, 2002; Maier *et al.*, 1998, 2000b, 2003; Salles *et al.*, 1993).

In this work, a three-step iterative numerical framework is introduced; to first synthesize porous geometries with a bimodal particle size distribution. Second, the

flow is simulated using the Lattice Boltzmann method with applied pressure difference as the driving force, and third, the dispersion was estimated by emulating the diffusion-advection displacement of a large number of point tracers via a random walk particle tracking implementation. The simulation results were benchmarked extensively against analytical models, similar numerical studies, and experiments reported in the literature.

CHAPTER 3

METHODS

In this chapter, the numerical methods, models and algorithms that were used in this work are described. Section 3.1 describes two algorithms for the construction of virtual porous media that attempt to mimic the physical synthesis procedures of silica and polymer monoliths by emulation of the diffusion of monomers and cluster formations with immediate bonding upon collision, namely; by diffusion-limited aggregation (Kolb, 1996). Section 3.2 illustrates how the Lattice Boltzmann method was implemented for flow simulation by using an open-source library with certain modifications. Additional helper functions and scripts were developed locally to aid with the necessary processing of the input and output files. In section 3.3, the coupling of the velocity field with diffusion to simulate dispersion and mass transfer, via the random walk particle tracking method, is explained in terms of the MATLAB implementation and the applied boundary conditions. All simulations were ran on an HP Z2 Tower G4 workstation with a 3.4GHz 6-core Intel processor equipped with a 32 GB of memory. The codes were written and compiled in MATLAB R2020b, except for the Lattice Boltzmann simulations which were performed using the PALABOS library written natively in C++. The algorithm flowcharts are given in Appendices B.1 through B.3. The codes are given in Appendices C.1 through C.4.

3.1 Construction of Virtual Porous Media

In this section, the two methods that were used for constructing virtual porous media are described. The first algorithm attempts to generate rod-like virtual porous geometries with a uniform pore size distribution to approximate a highly ordered porous media with monomodal particle size distribution. This method is based on random displacement of pixels in 3D space, hence it is referred to as the ‘pixel-based

algorithm' in this text. The second approach aims to emulate the sol-gel method where initial particles with spherical shapes are allowed to grow in size and diffuse in space, ultimately forming a 3D network of solid clusters which resembles non-uniform porous structures. This method is referred to as the 'sphere-based algorithm' in this text, as its legacy name. The constructed geometries were characterized in terms of porosity, equivalent particle size, and pore size distributions using locally written codes.

3.1.1 Pixel-based Algorithm

The mechanism starts with the random placement of logical ones and zeros, corresponding to the solid and void elements (voxels), respectively, in an initial 3D cubic matrix of a finite size. This algorithm was developed in our research group before it was used in this work (Koku, 2011). The ratio of the number of voxels filled with zeros to the entire size of the matrix is the desired porosity of the final geometry. The algorithm executes many iterations leading to the successive random displacement of the solid sites (logical ones) and their attachment to each other to form larger clumps, all of which are labeled and recorded as distinct entities by the algorithm. Convergence is achieved when all solid sites are connected to construct a single block, which corresponds to a final monolith-like porous geometry.

3.1.1.1 Input and Output

The input parameters to this algorithm, in addition to an empty output directory name, are:

1. Number of iterations. This is the maximum number of time steps the simulation is allowed to execute if convergence is not reached.
2. Lattice size. This is the length of one edge of the 3D simulation box, which makes the total number of elements of the lattice equal to the cube of the lattice size.

3. Porosity. This is the fraction of sites with logical zeros to the number of sites of the entire lattice. This quantity is conserved during the simulation as the algorithm does not introduce components of mass consumption or generation.
4. Connectivity specifier. This parameter is passed into a built-in MATLAB function (namely; `bwlabeln()`) which is used to identify connected groups of voxels and determine final convergence.

The connectivity of voxels in 3D is established according to a connectivity specifier of 6, which is the number of faces a cube has. Therefore, two voxels are declared connected if they share at least one face. This is the lowest value for this argument. Values of 18 (6 faces + 12 edges) or 26 (6 faces + 12 edges + 8 corners) can also be chosen depending on the purpose of the analysis. For example, for estimating a cluster analysis-based particle size distribution, higher connectivity values should be used to relax the connectivity criterion since cluster elements can be connected to each other by sides, faces or corners.

The output of the algorithm is a series of binary (black and white) images, such that when they are stacked together, they make up the constructed geometry which can be rendered in 3D for visualization, or compressed to a single data (.dat) file to be fed to the LBM flow simulation solver.

3.1.1.2 Estimation of Equivalent Particle Size

The characterization of the constructed media was done by numerical inspection of its statistical distribution of void or solid elements. The absolute porosity constitutes the number ratio of solid voxels to the entire cubic volume of the initial cartesian 3D matrix. The particle size was estimated by a numerical cluster analysis, and verified by analytical models based on the Darcy permeability. The particle size distribution was estimated in two different ways. The first method is by numerically conducting a cluster analysis that enumerates the groups of voxels that are connected through

edges or faces only and not vortices. An assumption was made that each cluster is a perfect cube and therefore the corresponding particle size is the cubic root of the cluster volume of an equivalent cube. The mean of the resulting cluster size distribution was then taken to be the equivalent particle size of the entire medium. The second method was analytically solving the combined Hagen-Poiseuille law and Darcy's law in Equation (3.1)

$$d_e = \sqrt{\frac{150(1-\epsilon)^2 k}{\epsilon^3}} \tag{3.1}$$

where d_e is the equivalent particle size, k is the permeability, and ϵ is the porosity. Both approaches gave similar results. The second approach is well-established and common in similar studies. Therefore, it was used as the primary approach to estimate the equivalent particle diameter, consistently throughout this work.

3.1.2 Sphere-based Algorithm

The sphere-based algorithm starts by randomly distributing many monomers in unique sites within a pre-defined 3D lattice. Unlike the pixel-based algorithm, the coordinate space here is continuous, where the particle sizes and position information are stored in a single data structure and processed by the algorithm before generating the actual binary mesh. The algorithm executes a series of time iterations where at each iteration, the monomers, which are identified by their center coordinates and radii, are allowed to diffuse in the 3D space and grow in size by a fraction of their size at the previous time step. When a collision is detected between two particles, they form a cluster that grows as more collisions take place. Convergence is achieved when all particles form a single cluster. On some occasions, the program may take too long to fulfill such a strict convergence criterion. Therefore, convergence is re-defined such that the largest cluster contains 95% (this percentage may be adjusted) or more of the total mass in the system.

3.1.2.1 Input and Output

The input to this program contains the following parameter set:

1. Lattice size. The length of one edge of the cubic lattice, making the number of total available sites equal to the cube of the lattice size.
2. Number of monomers. Number of monomer seeds to be randomly distributed in unique sites within the lattice.
3. Own-growth factor. This is a fraction of the particle diameter by which the particle's diameter grows at each time iteration.
4. Base radius. This is the initial radius of all particles at t = 0. It was found that it is easier to configure the base radius as a fraction of the lattice size.
5. Target extrinsic porosity. Because of the growth phenomena that this algorithm dictates, mass is being generated and therefore the user can only set a target porosity such that when it is reached, the own growth stops while diffusion and collision detection commands continue to be executed.
6. Base diffusion. This is the distance in pixels by which particles (or clusters) are allowed to jump at each time iteration.
7. Maximum overlap ratio. This parameter serves as the degree to which particles' overlap is controlled, to avoid complete fusion of spherical particles into each other.
8. Number of time steps. This is the maximum number of time steps or time iterations that the program is allowed to execute if convergence is not reached.

Overlap ratio is defined as the ratio of 2R (R being the sphere radius, where all spherical particles are identical in size) to the distance between the centers of the two colliding spheres. The maximum overlap ratio is the threshold beyond which the colliding particles are forced to repulse each other and move backward until the distance between their centers is equal to the sum of their radii; that is, the overlap ratio is equal to unity. Even with this overlap treatment put in place, unresolved overlaps still exist near the system boundaries. This happens when the backward

repulsion leads to it exceeding the system bounds, where the program always prioritizes the treatment of system boundary violation.

The output from the sphere-based algorithm is similar to the pixel-based algorithm: A series of binary images are generated that can be stacked together for 3D visualization, or compressed to a single data file as input to the LBM solver.

3.1.2.2 Mass Balance

Even though overlap is controlled to a reasonable degree, mass is still lost in the process when two separate particles fuse into each other partially without exceeding the overalp limit. Therefore, two types of porosity must be introduced: the first one is the extrinsic or hypothetical porosity, ρ_{ex} calculated at each time iteration according to Equation (3.2).

$$\rho_{ex} = 1 - \frac{N_{mon}\left(\frac{4}{3}\pi r_p^3\right)}{V_{lattice}} \tag{3.2}$$

where N_{mon} is the number of initial monomers, $V_{lattice}$ is the total volume of the system, and r_p is the particle radius. This superficial porosity serves only as an estimate to tell the program when to disable the particles' growth feature. The other porosity type is the intrinsic porosity ρ_{int}, calculated at the final step of the algorithm, after drawing the final binary mesh of the spherical objects, by using Equation (3.3).

$$\rho_{int} = 1 - \frac{V_{solid}}{V_{lattice}} \tag{3.3}$$

where V_{solid} is the volume occupied by solid-phase sites, which is equivalent to the number of voxels filled with logical ones. The intrinsic porosity is the actual system porosity used in subsequent calculations during the flow and mass transfer simulations. Having these two values for porosity closes the problem of mass balance where mass loss due to partial overlap can be estimated by Equation (3.4)

$$M_{loss} = V_{lattice}(\rho_{int} - \rho_{ex}) \tag{3.4}$$

This equation is consistent with the fact that ρ_{int} is smaller than ρ_{ex}, resulting in a negative value for the material loss.

3.1.2.3 Boundary Conditions

There are two types of boundary conditions in the sphere-based algorithm. The first one is the particle collision which is detected whenever the sum of two particles' radii is larger than or equal to the distance between their center coordinates expressed in Equation (3.5).

$$\left(r_{p_1} + r_{p_2}\right) \geq \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{3.5}$$

When this condition is encountered, the algorithm does two things; first, the two particles are registered as members of a cluster, such that their subsequent diffusion steps will be identical in magnitude and direction. The second thing is to further check how much the particles fused into each other, specifically if the maximum overlap ratio is exceeded. If so, the overlap is resolved by moving both particles in a repulsive motion away from each other until the sum of their radii is equal to the distance between their centers. To do this numerically, each of the colliding particles' coordinate is updated according to Equation (3.6).

$$x_i(t) = x_i(t-1) \pm \frac{0.5\left(\Sigma r_{p_i} - D_{ij}\right)}{\sqrt{1 + \alpha_{yx}^2 + \alpha_{zx}^2}} \tag{3.6}$$

where D_{ij} is the distance between the centers of the two particles, defined by Equation (3.7).

$$D_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2} \tag{3.7}$$

The parameter α expresses the relative difference in coordinate values with respect to the two axes in the subscripts, as illustrated by Equations (3.8) and (3.9).

$$\alpha_{yx} = \frac{y_1 - y_2}{x_1 - x_2} \tag{3.8}$$

$$\alpha_{zx} = \frac{z_1 - z_2}{x_1 - x_2} \tag{3.9}$$

This equation is applied for each of the three coordinates (x, y, and z) and α values are calculated accordingly. The $\pm$ sign alternates according to which coordinate value is higher. For example, if $x_1 > x_2$ Equation (3.6) expands to Equations (3.10) and (3.11).

$$x_1(t) = x_1(t-1) + \frac{0.5\left(\Sigma r_{p_i} - D_{12}\right)}{\sqrt{1 + \alpha_{yx}^2 + \alpha_{zx}^2}} \tag{3.10}$$

$$x_2(t) = x_2(t-1) - \frac{0.5\left(\Sigma r_{p_i} - D_{12}\right)}{\sqrt{1 + \alpha_{yx}^2 + \alpha_{zx}^2}} \tag{3.11}$$

Similarly for the y-coordinate, if $y_1 > y_2$ then Equation (3.6) becomes Equations (3.12) and (3.13).

$$y_1(t) = y_1(t-1) + \frac{0.5\left(\Sigma r_{p_i} - D_{12}\right)}{\sqrt{1 + \alpha_{xy}^2 + \alpha_{zy}^2}} \tag{3.12}$$

$$y_2(t) = y_2(t-1) - \frac{0.5\left(\Sigma r_{p_i} - D_{12}\right)}{\sqrt{1 + \alpha_{xy}^2 + \alpha_{zy}^2}} \tag{3.13}$$

Notice the change of subscripts of α as we switch from x to y. The equations above were derived from linear geometry analysis in 3D space, and it was tested multiple times in line by asking the program to print the overlap ratio before and after the treatment.

The second boundary condition is encountered at the system limits, and it is declared whenever a particle's radius is larger than the distance between its center and a system wall. System boundary violation can be resolved by a simple bounce-back

scheme, where the particle's coordinates are updated such that the distance between its center and the wall is equal to its radius, which is expressed mathematically in Equations (3.14) and (3.15).

$$x_i(t) = L - r_p, \quad x_i(t-1) > L \tag{3.14}$$

$$x_i(t) = 0 + r_p, \quad x_i(t-1) < 0 \tag{3.15}$$

Here the subscript i represents the coordinate axis; $i = x, y$ or z. The program imposes this boundary condition to all particles once every time iteration, right after the diffusive displacement. However, treatment of this boundary condition by bounce-back introduced a problem where the particles would precipitate in one corner of the lattice, which increased the anisotropy level of the final structure. Periodicity, as a more convenient approach, was later implemented as a replacement for bounce-back. The application of periodic boundaries is described by the following two equations.

$$x_i(t) = x(t-1) - L, \quad x_i(t-1) > L \tag{3.16}$$

$$x_i(t) = L - x_i(t-1), \quad x_i(t-1) < 0 \tag{3.17}$$

These equations were applied for each of the three coordinates. It is important to highlight the advantage of using this approach as opposed to bounce-back, which is that it allowed for the subsequent flow and mass transfer simulations to be performed with periodicity as well.

3.2 Simulation of Flow

The simulation of flow is conducted by using the Lattice Boltzmann Method (LBM). Due to the recent advances in massively parallel and very fast machines, the LBM has developed as a numerical scheme for simulating flow and other physical phenomena in fluids. When compared to traditional numerical methods that involve

the discrete forms of macroscopic continuum differential equations, the LBM excels in systems with complex boundaries and interfacial dynamics such as porous media.

3.2.1 The PALABOS Library

The Parallel Lattice Boltzmann Solver (PALABOS) is an open-source library, first released in 2010, that serves as a framework for modelling LBM problems numerically (Latt *et al.*, 2021). The code is written in C++ and is based on a Message Passing Interface (MPI) supporting parallel executions. In the simulation of flow in this work, the library was implemented with modifications to one of its auxiliary codes named 'permeability.cpp' located in the examples folder. The original script would output only the velocity magnitudes at certain cross-sections, hence another function was needed to have the code output the complete local velocity field in all the three coordinates. This function was developed as part of this work and is given in Appendix C.3.

The input parameters to the LBM solver are the following:

1. System size in pixels, as a three-component 1D array for the system length in the three coordinates x, y, and z.
2. Geometry data file. This file is generated using a helper function from the series of images representing the geometry. The number of images must be equal to the x- coordinate value. The height and width of each image must be equal to the y- and z- coordinate values. Therefore, coordinate x is used to denote flow direction, in all subsequent discussions within this text.
3. Applied pressure difference. For a pressure-driven flow, the difference between inlet and outlet pressures must be given to the solver in lattice units (non-dimensional). A method of conversion between lattice and physical units is explained in Appendix A.

Additional input parameters such as the fluid kinematic viscosity can also be declared within the script. A default value of 1/6 corresponding to the kinematic

viscosity of water was used. In some occasions, a tolerance parameter can be modified to control convergence speed at the expense of convergence quality, so it might be considered as an additional input.

The output from the LBM solver consists of multiple velocity data files. The number of files is three times the number of geometry cross-sections initially fed as input since there are three velocity components (x, y and z) at each cross-section.

3.2.2 Extraction of the Complete Velocity Field

The way the source code was modified is by defining a new function named `computeVelocities(args)` that is called once at the end of the simulation. The function loops through the resolved lattice cross-sections and calls a built-in function named `computVelocityComponent(args)` three times (for x, y and z) at each cross-section. Finally, the function saves each velocity component to a specified directory in a .dat format. Therefore, if the system length is 200 pixels, 200 binary images should be fed as an input, and 600 (200×3) velocity files are the expected output at the end of the simulation.

3.2.3 Visualization Tools

Once the complete velocity distribution is obtained, local MATLAB scripts were written to post-process the velocity files into a single MATLAB construct, three 3D matrices for each component. The script to do just that makes use of MATLAB cell arrays and other built-in concatenation methods. Then the velocity field structures are ready to be loaded as input to the mass transfer random walk code. The 3D visualization of the geometry is achieved by first generating an STL file from the original geometry mesh, with the help of custom functions. The STL file was rendered with Meshlab, a powerful open-source visualization software. Visualization of volumetric renderings of the velocity field was done by the ParaView software, which is excellent at creating automatically scaled color maps.

3.2.4 Boundary Conditions

The LBM solver offers both bounce-back and periodic boundary conditions at the solid-fluid interface, and the choice is made by the user on either of the two BCs. In this work, both periodic and bounce-back BCs were used to simulate both the rigid-spheres and the permeable-spheres geometries, which result in four system configurations given in Table 1.1. Moreover, a constant pressure difference was applied as the third boundary condition between the entrance and exit planes.

3.3 Simulation of Mass Transfer

Mass transfer is simulated by allowing a number of point tracer particles to diffuse through the system and subjected to the velocity field obtained from the LBM flow simulation. The method of coupling diffusion and convection by tracking tracers is known as the random walk particle tracking method (RWPT). The RWPT is originally modeled by a stochastic differential equation for the tracer position as in Equation (3.18).

$$dx(t) = v\big(x(t)\big) \cdot dt + B \cdot dW(t) \tag{3.18}$$

where B is a matrix of constant values that are proportional to the diffusion coefficient of the solute, and $W(t)$ is a continuous Gaussian process such as the Brownian motion. To numerically solve the random walk equation above, we followed the Euler approximation expressed in Equation (3.19)

$$x(t + \delta t) = x(t) + v\big(x(t)\big) \cdot \delta t + W\sqrt{2D_m \delta t} \tag{3.19}$$

where $v(x)$ is calculated locally by trilinear interpolation which means velocity varies linearly in x, y and z directions within a single voxel, δt is the timestep between two successive jumps, and W is the stochastic component that follows a binomial distribution and takes one of two values from the set {-1, 1}.

3.3.1 Input and Output

The RWPT method (Equation (3.19)) forms the backbone of the mass transfer simulation and was implemented by an in-house MATLAB code we created for this purpose. The algorithm takes many input parameters, the most important ones are the following:

1. Number of particles: Number of point tracers to be tracked while diffusing through the system. The higher the value of this parameter, the smaller the effect of randomness, but also the more expensive the computation gets.
2. Column length: A point in the axial direction such that when it is reached by a particle, its exit time is recorded and the program proceeds to the next iteration with a new particle. The length of the simulated geometry makes the upper bound for the value of this parameter.
3. Capacity factor: A parameter that measures solute retention to the stationary phase. It is defined as the ratio of the fraction of solute in the stationary phase to the fraction of solute in the mobile phase at any time step. The capacity factor takes values between 0 and 5.
4. Velocity field: The output data files from the LBM solver are processed into a MATLAB construct and fed as a three-component velocity field into the mass transfer code.
5. Geometry files: A series of binary images representing the constructed geometry. The mass transfer code processes the images inline into a 3D matrix which is used to identify interface boundaries.

The main output from the mass transfer code is an array of particle's exit time values that can be plotted as an RTD (Retention Time Distribution). Further post-processing of this information gives the time-based dispersion coefficient and plate height. It is also possible to have the program output the complete path (trajectory) of the particles through the system if trajectory analysis is desired, but this additional information requires more memory space during the simulation.

3.3.2 Choice of Simulation Time Step

While the LBM-obtained velocity field is scaled by a velocity multiplier u_f to vary the Péclet number, the time step δt was adjusted accordingly to satisfy the Courant–Friedrichs–Lewy condition in Equation (3.20).

$$u_f u_{max} \delta t + \sqrt{2 D_m \delta t} \leq \frac{l}{2} \tag{3.20}$$

which means that a tracer particle cannot undergo a displacement of more than half a lattice spacing, l, in one timestep. This approach requires all tracers to start from coordinates only within the mobile phase, which is done by distributing the tracers randomly on the non-solid points across the entrance plane.

3.3.3 Interpolation of the Velocity Field

The velocity field obtained from the LBM solver constitutes several arrays of discrete values given at rounded indices rather than in a continuous form. Therefore, some sort of interpolation of local velocities is necessary since the diffusive step is upper-bounded by a half lattice spacing according to Equation (3.20). In this work, a linear interpolation scheme is implemented in 3D (also called trilinear interpolation). Since there are eight voxel edges surrounding the point of interest, the corresponding eight velocity values are initially imported from the velocity field: $v_{x1}, v_{x2}, v_{x3} \dots v_{x8}$. Then linear interpolation takes place in three layers. The first layer is interpolating across an arbitrary dimension (Let's choose the z-coordinate) to reduce the eight velocity values to four values, as shown in Equations (3.21)-(3.24).

$$v_{x12} = v_{x1} + (v_{x2} - v_{x1})(z - \lfloor z \rfloor) \tag{3.21}$$

$$v_{x34} = v_{x3} + (v_{x4} - v_{x3})(z - \lfloor z \rfloor) \tag{3.22}$$

$$v_{x56} = v_{x5} + (v_{x6} - v_{x5})(z - \lfloor z \rfloor) \tag{3.23}$$

$$v_{x67} = v_{x6} + (v_{x7} - v_{x6})(z - \lfloor z \rfloor) \tag{3.24}$$

In the second layer of interpolation, the resulting four values from the first layer are processed similarly across one of the remaining two dimensions (Let's choose the y-coordinateh ere). This reduces the four velocities from the first layer to two velocities, as given in Equations (3.25) and (3.26).

$$v_{x1234} = v_{x12} + (v_{x34} - v_{x12})(y - \lfloor y \rfloor) \tag{3.25}$$

$$v_{x5678} = v_{x56} + (v_{x78} - v_{x56})(y - \lfloor y \rfloor) \tag{3.26}$$

The third and last interpolation layer simply interpolates between the two velocity values resulting from the second layer across the remaining dimension (x-coordinate), as expressed in Equation (3.28).

$$v_x = v_{x1234} + (v_{x5678} - v_{x1234})(x - \lfloor x \rfloor) \tag{3.27}$$

Here v_x is the interpolated x-component velocity at position (x, y, z). In this set of equations, the operator $\lfloor x_i \rfloor$ gives the floor of the coordinate value. Equations (3.21) to (3.28) are expressed for the x-component only of the velocity field (v_x), and for the other two field components (v_y and v_z) the same equations are applied on the corresponding edge velocities; that is, $v_{y1}, v_{y2} \ldots v_{y8}$, and $v_{z1}, v_{z2} \ldots v_{z8}$ for the y- and z- components, respectively.

3.3.4 Boundary Conditions

There are two types of boundary conditions in the mass transfer simulation. The first one is encountered at the solid-fluid interface. Although the random walk model seems to be well-established for flow regions sufficiently far from solid boundaries, there does not appear to be a strong consensus on a correct approach to describe particle behavior at the solid-fluid interface. Here we followed a bounce-back approach such that when a particle's position is detected to be in the solid phase, it is sent back to its position at the previous time step, as shown in Equation (3.28)

$$x_i(t) = x_i(t - \delta t), \quad x_i(t) \in \mathcal{P}_s \tag{3.28}$$

where x_I is the particle's position, δt is the time step, and $\mathcal{P}_s$ is the solid phase space. The second boundary condition is encountered at the system walls when a particle diffuses into a new coordinate beyond the system limits. The treatment of such a condition is done by updating the violating coordinate value to be exactly at the system boundary. For example, when a particle's $x-$coordinate is detected to have a higher value than the system length, L, the coordinate value becomes equal to L. This can be generalized for the two system boundaries, $x_i = L$ and $x_i = 0$, as described by Equations (3.29) and (3.30).

$$x_i(t) = L, \quad x_i(t - 1) > L \tag{3.29}$$

$$x_i(t) = 0, \quad x_i(t - 1) < 0 \tag{3.30}$$

An alternative setup for treating the second boundary condition (at system boundaries) is the use of periodic boundary conditions such that when a particle exceeds the system bounds, it appears from the opposite side by the same offset, which is described by Equations (3.31) and (3.32).

$$x_i(t) = 0 + \delta x, \quad x_i(t - 1) = L + \delta x \tag{3.31}$$

$$x_i(t) = L - \delta x, \quad x_i(t - 1) = 0 - \delta x \tag{3.32}$$

Both periodic and bounce-back schemes were investigated in this work. The use of periodic boundaries is, theoretically, more convenient since the simulated geometry is too small to represent an actual system, therefore, applying periodicity treats the simulated sample as a central region from a larger geometry.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Constructed Porous Media Systems

In this section, the virtually constructed porous geometries are presented. As described in the previous chapter, two algorithms were used to generate two types of porous media: pixel-based and sphere-based. The pixel-based structure has a skeleton thickness of one voxel and a pore diameter of 1-2 voxels. Compared to the pixel-based ones, the resulting sphere-based structures are a few orders of magnitude larger in particle size as well as in pore size.

Therefore, it was found convenient to merge the two structures (Figure 4.4) to form a hierarchically structured porous media with large thorough pores for convective flow and small microspores for diffusion, which closely resembles real-life silica-based monolithic columns (Table 4.5). In the early stages of this work, the flow and dispersion simulations were performed only on the pixel-based structure since the sphere-based algorithm was still under development.

The pixel-based structure is highly uniform in terms particle and pore size distributions in such a way that it doesn't resemble real-life chromatographic media. Therefore, after successfully generating the sphere-based structure, the role of the pixel-based one was reduced to model the intra-particle porous space. In this chapter, the flow and dispersion simulations are presented for the sphere-based geometry and the hierarchically porous structure resulting from the merge (Figure 4.4), which are referred to as the rigid-spheres system and the permeable-spheres system, and the simulation results from early work on the pixel-based structure are given in Appendix D for reference.

4.1.1 Pixel-based Structure

The pixel-based algorithm aimed to generate virtual porous structures that resemble mesopores in a silica-based monolith. The algorithmic implementation was presented in more detail in the previous chapter. There are four adjustable parameters in this model: system size, number of time iterations, porosity, and connectivity specifier. Both the resulting final structure and the computational performance of the algorithm are highly sensitive to variations in the input values as will be discussed shortly. The constructed structure using the pixel-based algorithm is presented in 3D in Figure 4.1 with a size of 100 voxels on each side (top), and a close-up view (bottom) that shows more details about the structure surface. The four adjustable parameters in this model were porosity, maximum number of iterations, system size,

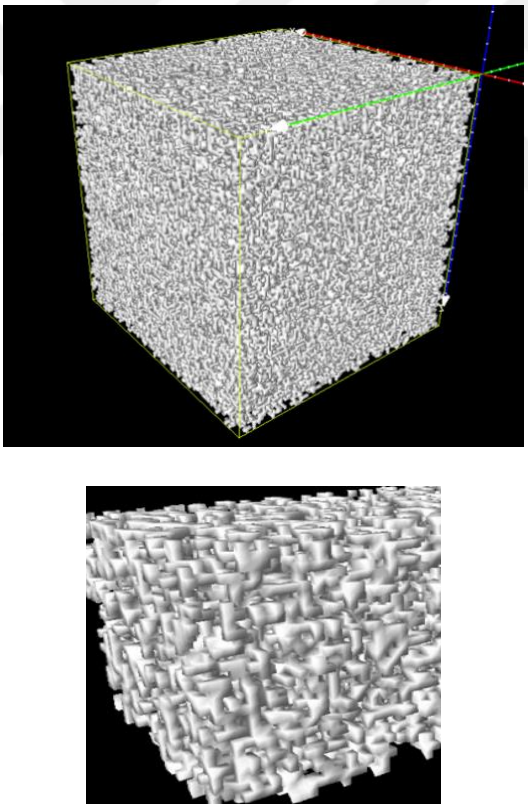


Figure 4.1. The pixelated porous media with a size of 100^3 voxels (top), and a regional cut from the top edge for a closer view (bottom). The size of 100 pixels is equivalent to 2.45 μm.

and a connectivity specifier. Setting lower values for porosity had a slight effect on convergence as a high number of monomers take less time to form a single block, but leads to denser media. The choice of 0.70 for porosity is justified as a typical value in commercial silica monoliths (Hlushkou, Bruns, & Tallarek, 2010), despite its leading to high convergence time (About 30 minutes on an 6-core 3.4 GHz Intel Processor with serial computing).

The second parameter was the maximum number of iterations. This parameter served only as an upper limit below which convergence is expected, therefore it had no practical effect on the final structure. The third parameter was the system size taken as the edge length of the cubic lattice. This parameter was extremely important for convergence speed. If one chooses to generate a larger system, the time complexity of the simulation increases exponentially. The effect of varying system size was not apparent in the final structure in terms of its morphology. The fourth parameter was the connectivity specifier, which is an integer that can be set as 6, 18, or 26, representing the mode by which neighboring voxels are declared connected. For example, a value of 6 means two voxels are connected if they share a face (a cube has 6 faces). If set to 8, connectivity is achieved if the two voxels share a face or an edge (6 ‘faces’ + 12 ‘edges’ = 18).

The last value that this parameter takes is 26, which includes the vortices of the voxels (8 vortices in a cube). As can be deduced, the higher the value of the connectivity specifier parameter, the less strict the connectivity requirement is, leading to faster convergence. In this work, the smallest value of 6 (two voxels must share a face to be declared connected and form a block) was chosen, which leads to a geometry of the highest possible quality. Due to the nature of the model and the algorithmic implementation, the skeleton size of this structure is limited to 1 voxel and the pore size to 1 – 2 voxels. Calculation of the pore size distribution using advanced image processing will be given in the following sections.

4.1.2 Sphere-based Geometry

The sphere-based algorithm generates porous structures virtually which aims to emulate a number of monomers that grow in size and diffuse to form clusters in a pre-defined cubic lattice, similar to the sol-gel method. A detailed description of the implementation was given in the previous chapter. The resulting structure from the sphere-based algorithm is shown in Figure 4.2 which has a size of 500^3 voxels. The structure is inherently isotropic with a porosity of 0.66, determined by an input parameter during initialization. The effect of varying three of the input parameters is discussed below. These parameters are growth factor, base radius and number of monomers.

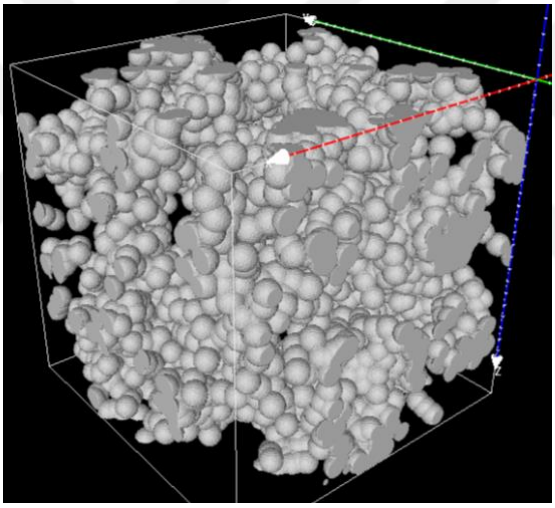


Figure 4.2. The generated structure by the sphere-based algorithm. The size of the lattice is 500^3 voxels (12.2^3μm). The total porosity of this structure is 66%.

4.1.2.1 Particle Growth Factor

The particle growth factor σ is defined as the fraction of the particle's radius by which the radius is increased at each time step.

$$r_p(t) = r_p(t-1) + \sigma\, r_p(t-1) \tag{4.1}$$

where r_p is the particle radius and t is the simulation timestep. To study the effect of varying the growth factor σ, the remaining input parameters are kept constant as given in Table 4.1.

Table 4.1 Fixed parameters while optimizing particle growth factor.

Parameter	Value
Lattice size	5 pixels
Number of monomers	240
Base Radius	0.005 × Lattice size
Target porosity	0.60
Base diffusive step	0.1 × Lattice size

The growth factor varied in the range [0.01, 0.05, 0.1, 0.5]. It was concluded that when growth speed is high, particles grow and fill the space before they collide naturally due to diffusion. Particle collision in this case is caused by the rapid growth in size, rather than by diffusion, which leads to the formation being more similar to random sphere packing rather than monolithic structures. The choice of 0.05 for the growth factor was made because it was large enough to avoid elongated clusters that would create large flow channels and small enough such that the final geometry would not look like a random sphere packing.

4.1.2.2 Base Radius

Similarly, the base radius was varied in the range [0.001, 0.005, 0.01, 0.05], and it was found that when the initial radius is large, this contributes to faster convergence of forming a single cluster, which means the particles will form clusters due to their large size rather than due to diffusion. A large initial radius makes the final structure look like a random sphere pack. A small initial radius leads to forming elongated clusters, causing large pores to form in the structure. Correspondingly, a value of

0.01 for this parameter was chosen which avoids forming large flow channels as well as avoiding the formation of a very narrow pore sizes distribution.

4.1.2.3 Number of Initial Monomers

At the initialization step, a finite number of monomers were randomly distributed within the 3D space. This parameter can be conceived as the resolution of the simulation; the higher the number of particles, the higher the computational complexity, and the better the resulting structure in likelihood of representing a real material. The number of monomers varied across three values: 100, 250, and 500. It was found that increasing the number of monomers at a fixed target porosity produces a structure with smaller particle sizes and a closer resemblance to contemporary chromatographic stationary phases. This also produces fewer flow channels and a more compact structure. Figure 4.3 shows a comparison of two simulation runs at different values for the initial number of monomers.

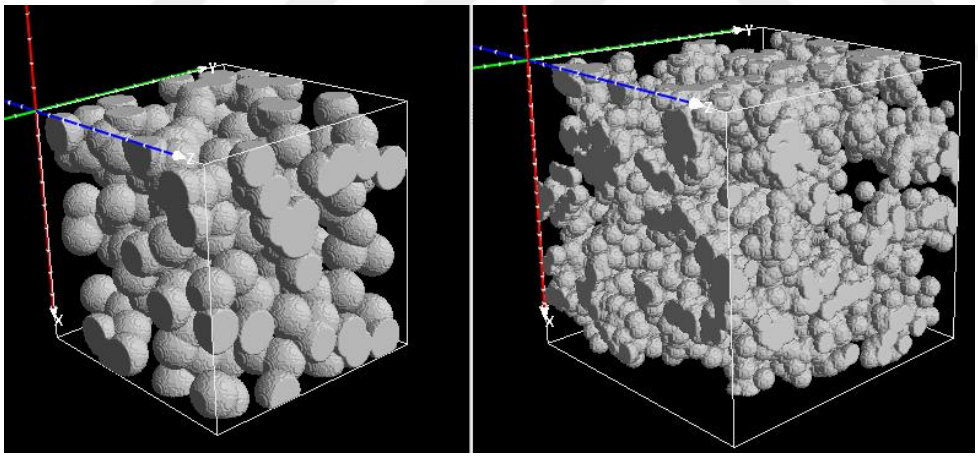


Figure 4.3. Effect of increasing the initial number of particles on the resulting structure while fixing the other parameters. The simulation was run for 240 particles for the structure on the left, and 1600 particles for the one on the right.

The main challenge regarding the presented algorithms for constructing porous media virtually is the parallelization of the codes to speed up the simulation for larger systems to eliminate the effect of a pixilated solid phase. As this particular simulation

consists of a time-based loop and two nested particle-based loops such that the results from a certain iteration directly influence the next one, simple task parallelization could not be performed. More work is required to implement the appropriate parallelism technique which has to introduce major changes to the core of the algorithm.

4.1.3 Construction of a Heterogeneous Porous Media

This section presents the results of combining the two structures discussed in the previous two sections to form a heterogenous porous media in an attempt to resemble a structure with bimodal porosity, e.g. in a silica monolith. The idea was to merge the structure shown in Figure 4.1 with the one shown in Figure 4.2 by first defining a new 3D box of size 500^3 voxels which had the same size as that of the sphere-based system. Then the solid space information was imported from the pixel-based system while the void space information was taken from the sphere-based system. This masking method is further illustrated in Figure 4.4 in 2D, although the procedure was performed originally on the 3D structures.

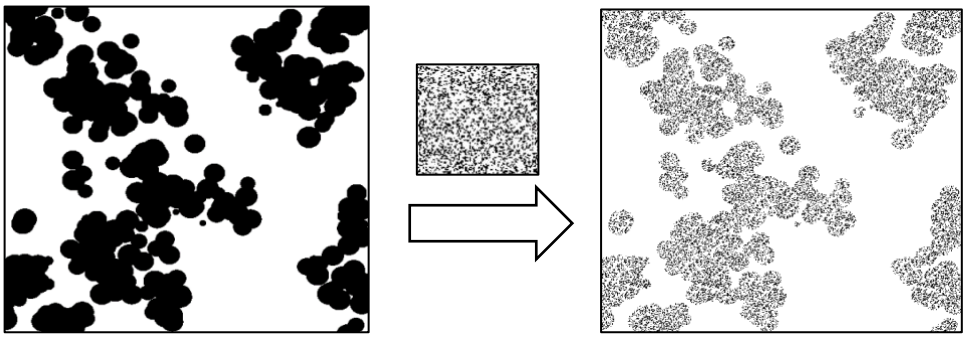


Figure 4.4. Merging of the pixel-based and sphere-based systems to make a heterogeneous porous media of permeable spherical particles. The size of the resulting (as the original) frame is 500X500 pixels ($12.2\mu m \times 12.2\mu m$), and the size of the pixelated base structure is 100X100 pixels ($2.45\mu m \times 2.45\mu m$).

As a result of this geometry merge, the total porosity was increased from 0.66 to 0.89 given that the porosity of pixelated geometry was 0.70 as shown in Figure 4.5 where the cross-sectional porosity is plotted against axial length before and after the merge.

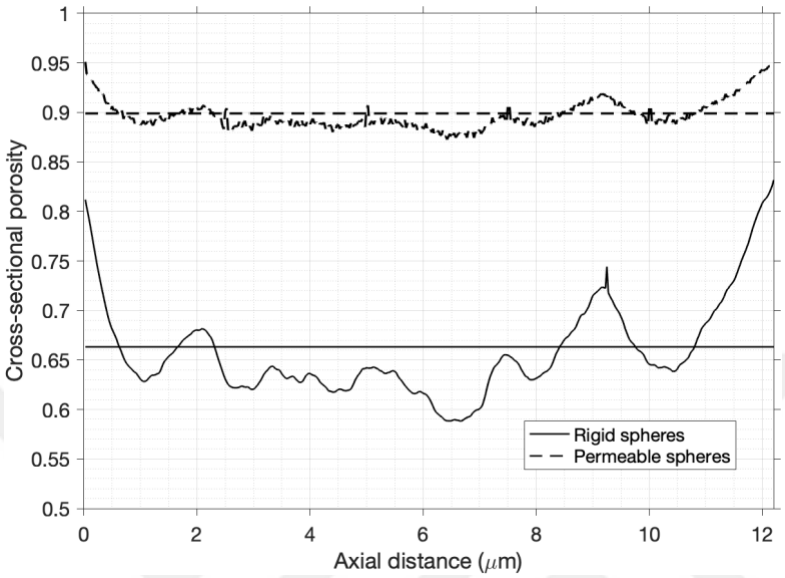


Figure 4.5. Distribution of cross-sectional total porosity for both the rigid and permeable spheres geometries. The horizontal lines indicate the average total porosities as 0.66 for the rigid spheres system and 0.89 for the permeable spheres system.

The pore size distribution of the resulting media was calculated by the method of skeletonization (Delerue *et al.*, 1999) which was implemented in MATLAB by our research group prior to this work (Koku, 2011). Figure 4.6 (right) shows a color map of pore sizes in terms of their diameters. To visualize the mesopores within the spherical particles, Figure 4.6 (left) is a smaller domain of the same slice that gives the pore sizes within the intraparticle mesoporous space. In the context of this study, the term macropores corresponds to pores that are larger than 1 μm in dimater, and below that, it is referred to as mesopores. The smallest pore size in this study is limited to 1 pixel which is equivalent to 0.025 μm.

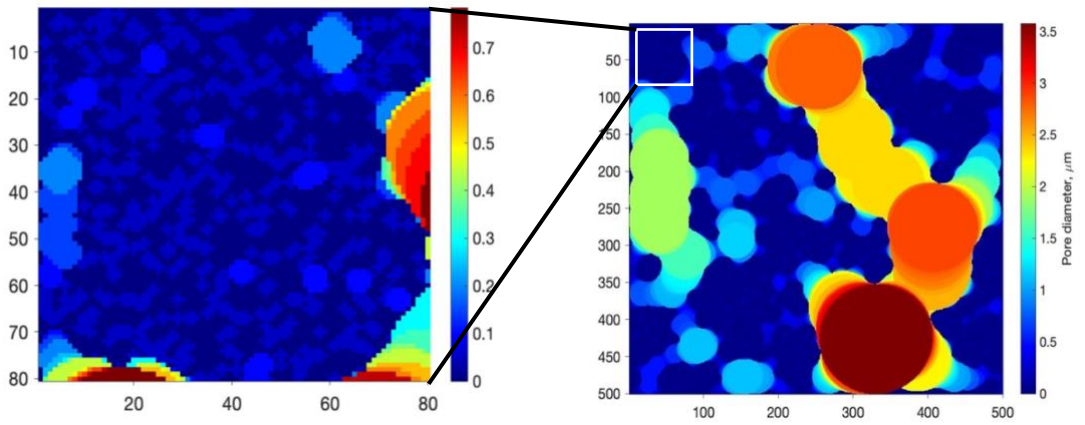


Figure 4.6. Distribution of pore sizes through the method of skeletonization for the middle slice of the generated heterogeneous system on the right frame; and a cut from the top-left region to show the pore size distribution within the spherical particles on the left frame. The values in the color bars are given in micrometers and represent the pore diameters.

4.2 Simulation of Flow

In this section, the results of implementing the Lattice Boltzmann Method for solving the velocity field are presented. The PALABOS library was used with some modifications to perform the Lattice Boltzmann Method with a D3Q19 descriptor. As a result of the constructed geometries discussed in the previous section, the velocity field was calculated for two systems: Rigid spheres, and permeable spheres, and in two modes of boundary conditions: bounce-back and periodic. The flowing medium was assumed to be water with a kinematic viscosity of 1×10^{-6} m^2/s. The pressure difference between the entrance and exit planes was imposed as a boundary condition and was set as 6.0 kPa for all of the four simulations.

The pressure gradient was calculated as 0.55 MPa/m for the base runs. This value is within the recommended range for chromatographic stationary phases (Leinweber & Tallarek, 2003). The average velocity was calculated as the sum of all local velocity

values divided by the total volume in the lattice. Table 4.2 summarizes the calculated permeabilities, average velocities and Reynolds numbers for the four runs. The average velocity was calculated as the sum of local velocities divided by the total volume of the lattice.

Table 4.2 Summary of LBM simulations on the four configurations.

System	ID	u_{avg}^{x} (m/s)	Permeability (m^2)	Re
Rigid spheres, Bounce-back	1	0.025	4.82E-14	0.468
Rigid spheres, Periodic	2	0.032	6.51E-14	0.663
Permeable spheres, Bounce-back	3	0.029	5.18E-14	0.101
Permeable spheres, Periodic	4	0.038	6.97E-14	0.151

The system permeability κ was calculated by Darcy's law

$$\kappa = \frac{\mu u_{avg}^{x}}{\Delta P/L} \tag{4.2}$$

where μ is the fluid kinematic viscosity, u_{avg}^{x} is the average velocity of the axial component, averaged over both phases of the geometry, and $\Delta P/L$ is the applied pressure drop per unit length. The dimensionless Reynolds number

$$Re = \frac{u_{avg}^{x} \cdot d_e}{\nu} \tag{4.3}$$

where ν is the kinemetic viscosity and d_e is the equivalent particle size, was required to determine the flow regime. Values of the Reynolds number for permeable spheres are smaller, although with high permeabilities, due to the fact that the equivalent particle size is smaller for the permeable-spheres system. If the Reynolds number is below unity, the flow is within the creeping flow regime and this permits linear scaling of the velocity field for analyzing the relationship of Péclet number and dispersion coefficient as discussed later with the mass transfer simulation results.

Since the equivalent particle size d_e is required to calculate the Reynolds number, it was estimated using the Kozeny-Carman equation combined with Darcy's law

$$d_e = \sqrt{\frac{150(1-\epsilon)^2}{\epsilon^3}\,\kappa} \tag{4.4}$$

where ϵ is the void fraction of the media and κ is the permeability. The equivalent particle size was estimated in this method as 1.68 μm as a result of the rigid spheres simulation. However, for the heterogenous system where the particles are permeable, the hydraulic particle size was calculated as 0.38 μm by the same method.

As a reference comparison, the actual sphere diameter which resulted from the geometry-making simulation was 0.82 μm. The smallest spacing of one pixel is the nominal skeleton width that makes up the small pores within the particles, and it is equivalent to 0.025 μm. The use of periodic boundaries during the LBM simulations had a negligible effect on hydraulic particle size values in both configurations.

4.2.1 Velocity Distribution

The velocity distribution was calculated by first converting the velocity field to a one-dimensional array and then associating each velocity value with its frequency number, excluding the zero values corresponding to the solid phase. Then the velocity values were plotted on the horizontal axis and the corresponding frequency (normalized by the number of pore voxels) wass plotted on the vertical axis. Figure 4.7 shows the lateral and axial velocity distributions for the four system configurations. Only the y-coordinate lateral velocity distribution is shown as the distribution is identical for both lateral coordinates (y and z).

The lateral velocity distributions are symmetrical about zero while the axial distribution is skewed to the right due to the uneven distribution of the volumetric flow throughout the geometry matrix, where the high flow rates are relatively scarce and constitute smaller fractions of the porous space.

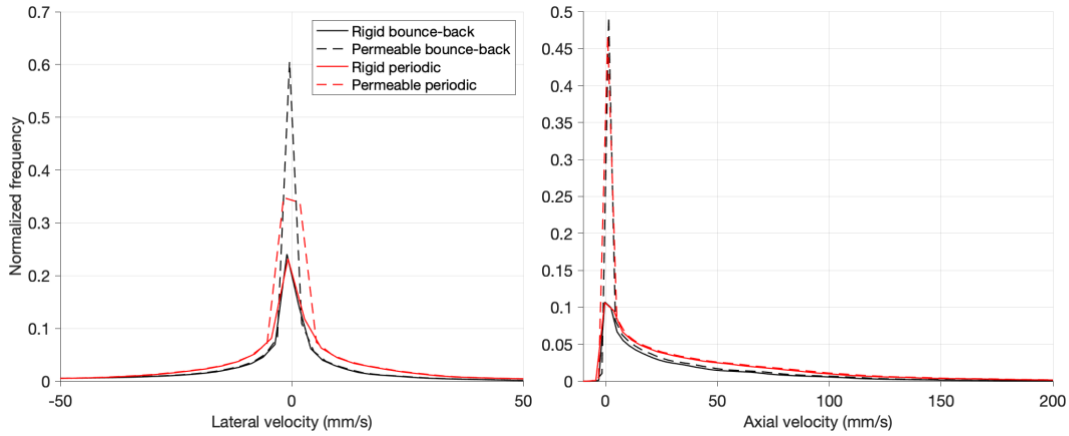


Figure 4.7. Axial and lateral velocity distributions for the four configurations. Lateral velocities are symmetrical at zero. Systems with permeable particles show higher peaks compared to those with rigid spheres.

Both lateral and axial distributions exhibit higher peaks for the permeable spheres system because of the higher number of pore voxels, which translates to higher frequencies for the velocity values that correspond to the interparticle pore space. The axial velocity distributions shown in Figure 4.7 covers a range starting at -8 mm/s and ending at a maximum axial velocity of 260 mm/s. The lateral velocity distribution forms a symmetrical bell shape with a median and mean value of 0 mm/s. The velocity distributions discussed above are in agreement with similar literature works (Hlushkou, Bruns, & Tallarek, 2010; Koku *et al.*, 2011).

4.2.2 Visualization of Local Velocities

So far the velocity fields for the two systems were presented in terms of distribution histograms which provide an important insight into the global behavior of the flow. It is also important to visualize localized velocity magnitudes to have a more detailed outlook of the flow field, especially concerning the interactions between the flow and the stationary phase. Figure 4.8 depicts the distribution of velocity magnitudes throughout the 3D lattice, highlighting the difference between bounce-back (top panel) and periodic (bottom panel) schemes as boundary conditions. All values in

Figure 4.8 are in lattice units and can be converted to physical units by considering the conversion factors for velocity which is equal to $245\,\mathrm{m/s}$ and for length 2.45×10^{-8} m.

With this mode of visualization, only the rigid-spheres system is considered, since the velocities associated with the interparticle porous space are very close to zero as evident by the large peak in the axial velocity distribution (see Figure 4.7), and therefore the volumetric renderings of the permeable-spheres and rigid-spheres systems are similar. The effect of using periodic boundaries on the velocity field is apparent in Figure 4.8 where higher values for the maximum velocity ($340\,\mathrm{mm/s}$) were achieved compared to the simulation with the bounce-back boundary condition ($270\,\mathrm{mm/s}$). This is expected as a result of the absence of walls as the total fluid-solid interactions are smaller, leading to lower resistance to fluid momentum.

For a closer look at the distribution of flow within the geometry, two cross-sectional 2D velocity fields are shown in Figure 4.9 at the middle plane (x = 250 pixels = 6.12 μm) for the rigid-spheres system with the bounce-back boundary condition (top image) and periodic (bottom image). The no-slip phenomenon is apparent in the flow field with bounce-back, where velocities have minimum values near the lattice boundaries.

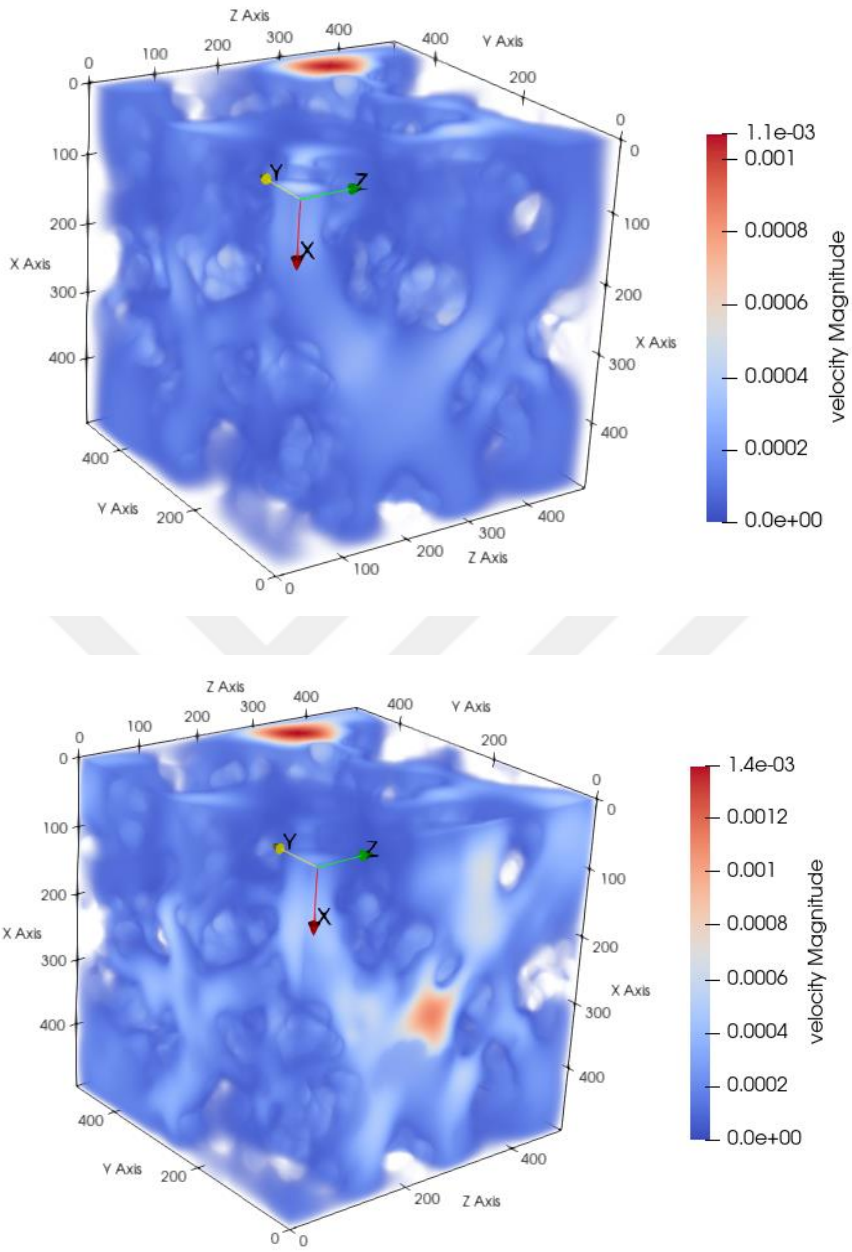


Figure 4.8. Three-dimensional rendering of the velocity distribution in the constructed rigid spheres geometry with the bounce-back boundary condition (top) and periodic boundaries (bottom). The color code shows the localized velocity magnitude in dimensionless simulation (lattice) units, while conversion to physical units is achieved by considering a velocity conversion factor of $245\ \mathrm{m/s}$.

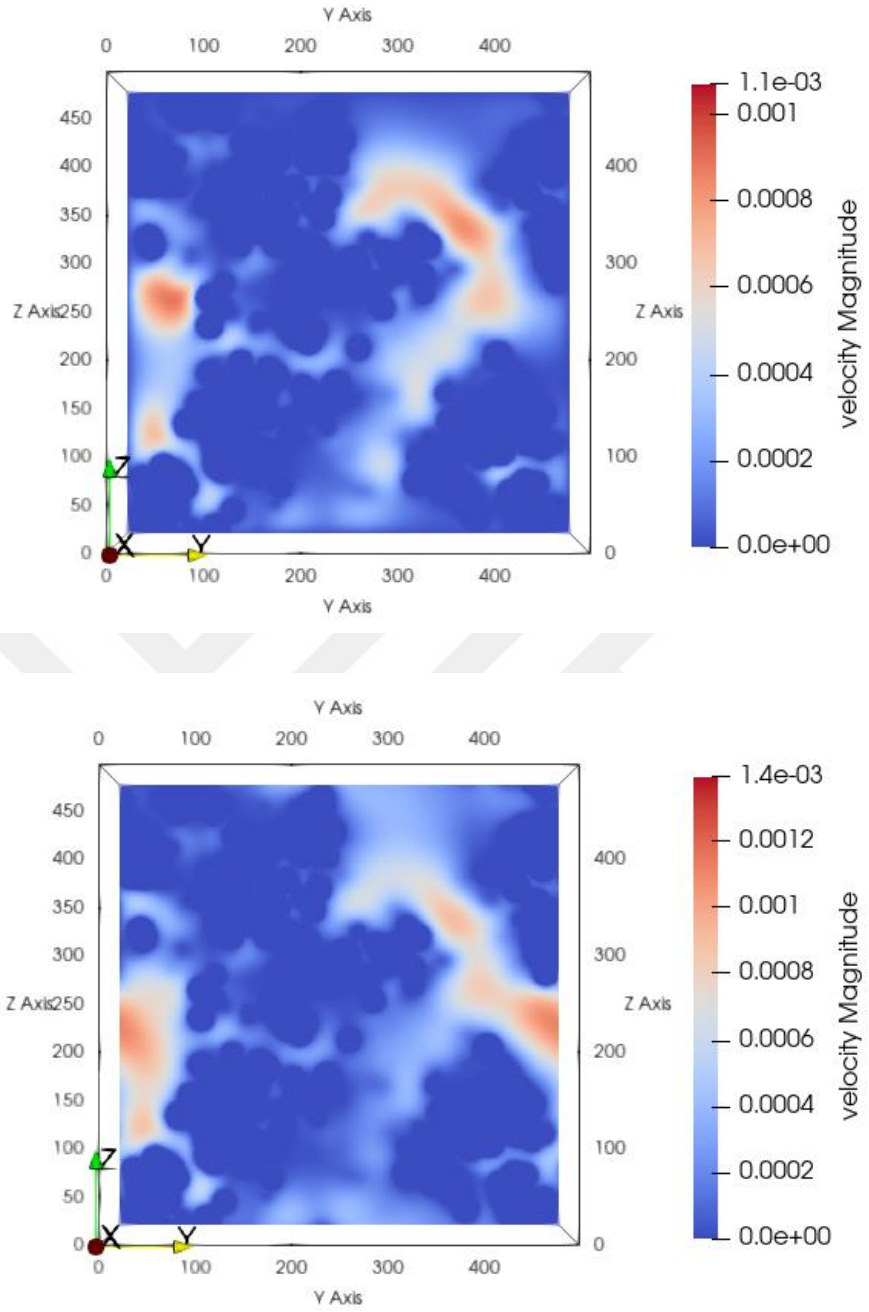


Figure 4.9. Cross-sectional velocity field for the rigid-spheres system at the halfway plane (x = 6.12 μm) for the rigid spheres system with bounce-back (Top) and periodicity (Bottom). Values of the velocity field are given in dimensionless lattice units, which can be converted to physical units by multiplying by the velocity conversion factor of 245 m/s.

4.2.3 Comparison with Similar Literature Studies

The flow field as visualized in the previous section proves the existence of hot spots associated with large-pore regions, and the maximum velocity was achieved in such regions, which amounts to 10.6 times the average velocity. To assess these results, it is convenient to compare the velocity distribution of this work with similar works in the literature.

Figure 4.10 shows a comparison between the axial velocity distribution of this work for the rigid-spheres system with the results of (Hlushkou, 2010) in their work on a flow simulation of a reconstructed silica monolith. This comparison illustrates how the flow field distributions are similar, which indicates that such flow heterogeneities are common and expected in porous media flow modeling.

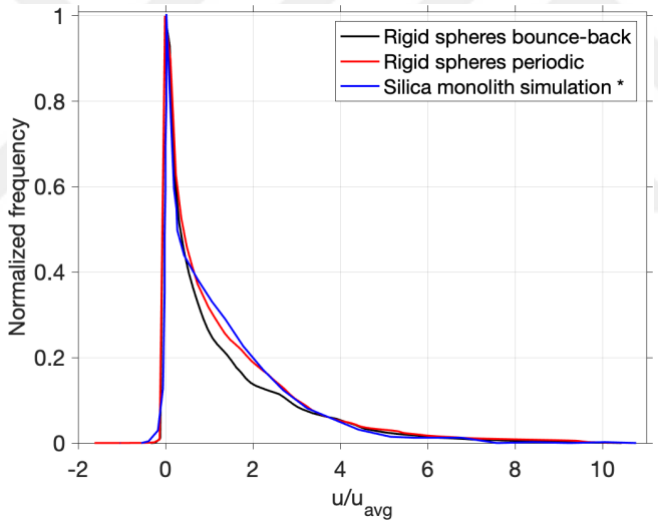


Figure 4.10. Comparison of the flow field distribution obtained in this work to a reconstructed silica monolith simulation (*Hlushkou, 2010). The axial velocity is normalized by its volumetric average, and the frequency by its maximum.

4.3 Simulation of Dispersion in Virtual Porous Media

The Random Walk Particle Tracking (RWPT) is a well-established approach to model dispersion, especially in complex media where analytical solutions are not available. The RWPT method was implemented in this work to emulate the diffusion-advection displacement of a large number of point tracers, ultimately estimating their dispersity in the constructed geometries.

4.3.1 Validation of the Random Walk Implementation

After the random walk script was developed, and before running it on the constructed structures, the code was examined on simpler geometries for which analytical solutions of the velocity field, concentration distribution, and dispersion are available. Two test geometries were considered: A cylindrical tube and a rectangular duct. The analytical close-form solutions for these two systems are given in (Chatwin & Sullivan, 1982; Taylor, 1953).

The concentration profile for a fully developed flow in a cylindrical tube was calculated both analytically and using the RWPT code and the comparison result is shown in Figure 4.11 (top). The velocity field that was used in the RWPT-based results was obtained by the lattice Boltzmann method.

Therefore, considering the resulting curves, both the LBM-based flow solution and the random walk code were verified. Furthermore, the dispersion coefficient was plotted for a range of Péclet numbers for a cylindrical cube and a rectangular duct, and the results are in excellent agreement, as shown in Figure 4.11 (bottom left) and Figure 4.11 (bottom right).

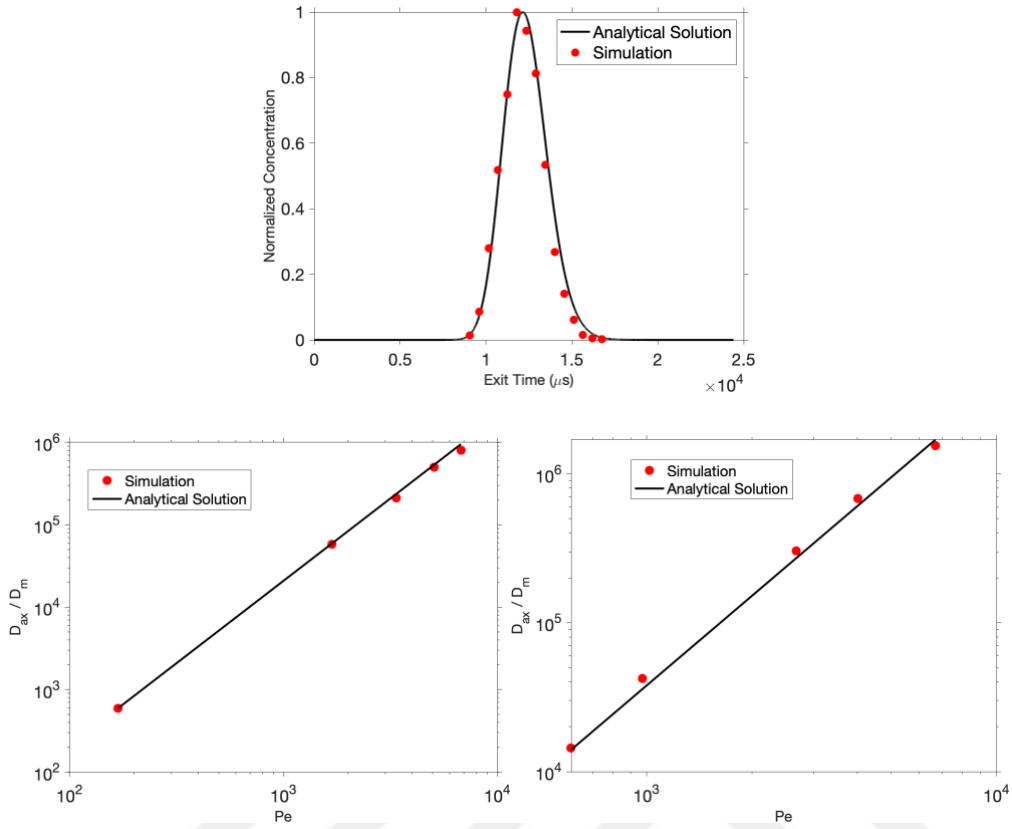


Figure 4.11. Validation of the implemented random walk code by comparison with analytical solutions on test systems. (top panel) Exit time distribution in a cylindrical tube. Normalized axial dispersion coefficient in a cylindrical tube (bottom left), and a rectangular duct (bottom right).

4.3.2 Simulation of Dispersion

For the simulation of dispersion in this study, two main assumptions were made regarding the tracer particles. The first is that the tracers are point tracers rather than having finite sizes, and the second assumption is that the interactions between the tracers were ignored. These two assumptions were used in previous simulation studies in the literature on various types of porous media, especially for stationary phases in chromatography.

The advantage of ignoring particle-particle interactions is that it allows for parallel execution since each particle-based iteration executes the model commands regardless of the state of the other particles. Nonetheless, this assumption is justified by the fact that many practical applications of analytical chromatography utilizes pulse injection of the analyte sample, which means the analyte concentration is sufficiently low to ignore the interactions of the sample molecules. Moreover, this assumption does not affect the objective of this work, which is to develop a numerical framework that serves mainly as a directional guide to aid the research in the field of development of chromatographic stationary phases.

The molecular diffusion was, consistently throughout this work, assumed to be $1 \times 10^{-10} \mathrm{m}^2/\mathrm{s}$. This value is sufficiently close to that of common protein samples separated by chromatography methods. For example, the diffusion coefficients for small to medium size proteins like lysozyme and ovalbumin were reported as 1.1×10^{-10}, and $0.8 \times 10^{-10} \mathrm{m}^2/\mathrm{s}$, respectively (Bello *et al.*, 1994). To vary the Péclet number, an array in the log space was generated as a velocity multiplier array which scales the velocity field and is justified by the low Reynolds numbers, as given in Table 4.2, for which the creeping flow regime is valid.

The elution curves for the tracer particles for three Péclet numbers are shown in Figure 4.12 as an illustration of the implemented flow scaling approach. The total range of the velocity multiplier array was decided such that the Reynolds number does not exceed unity. For the four system configurations, the ranges of Pe number, Reynolds number, average velocity, and pressure gradient are given in Table 4.3.

It is important to note that the ranges given in Table 4.3 are extended beyond and below the operating conditions of typical stationary phases. For example, in the study of Hlushkou *et. al* on a commercial silica monolith, the experimental operating range for the average velocity and the pressured gradient is given in Table 4.4, which is only a subset of the range investigated in this work.

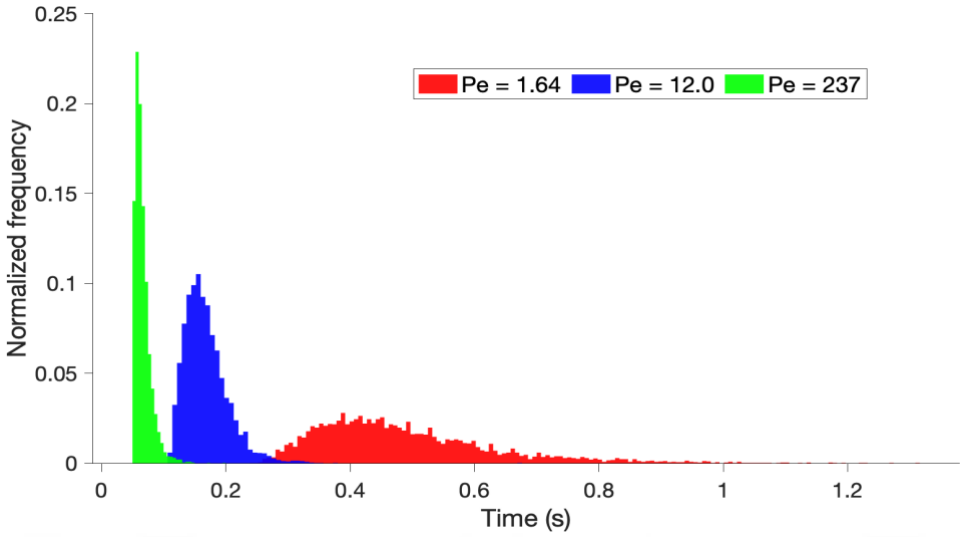


Figure 4.12. Example exit time histograms for different values of Péclet number as a result of the random walk simulation on the rigid-sphere system.

Table 4.3 Results of the linear scaling of the velocity field on the four systems.

System ID	Re		Pe		u^{x}_{avg} (mm/s)		$\Delta P/L$ (MPa/m)	
	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
1	4.35e-05	0.653	0.060	894	0.002	35.4	0.049	736
2	6.62e-05	0.663	0.094	935	0.003	31.9	0.049	490
3	8.93e-06	0.447	0.009	467	0.003	127	0.049	2450
4	1.34e-05	0.673	0.015	727	0.003	170	0.049	2450

Table 4.4 Experimental range for a commercial silica monolith reported in (Hlushkou *et. al*, 2010)

Parameter	Minimum	Maximum
Average velocity (mm/s)	0.25	1.75
Pressure gradient (MPa/m)	5.0	29

Before discussing the results of the mass transfer simulations according to the generated range of linearly scaled flow fields, it is worth noting that some of the data points did show a high level of discrepancy when compared with other literature studies, particularly when the Reynolds number is high or close to unity. Therefore, only relevant subsets of the generated range were used when compared with other studies. In Figure 4.13, all data points that correspond to the range given in Table 4.3 are presented in terms of the calculated dispersion coefficient normalized by molecular diffusivity in relationship with the Péclet number.

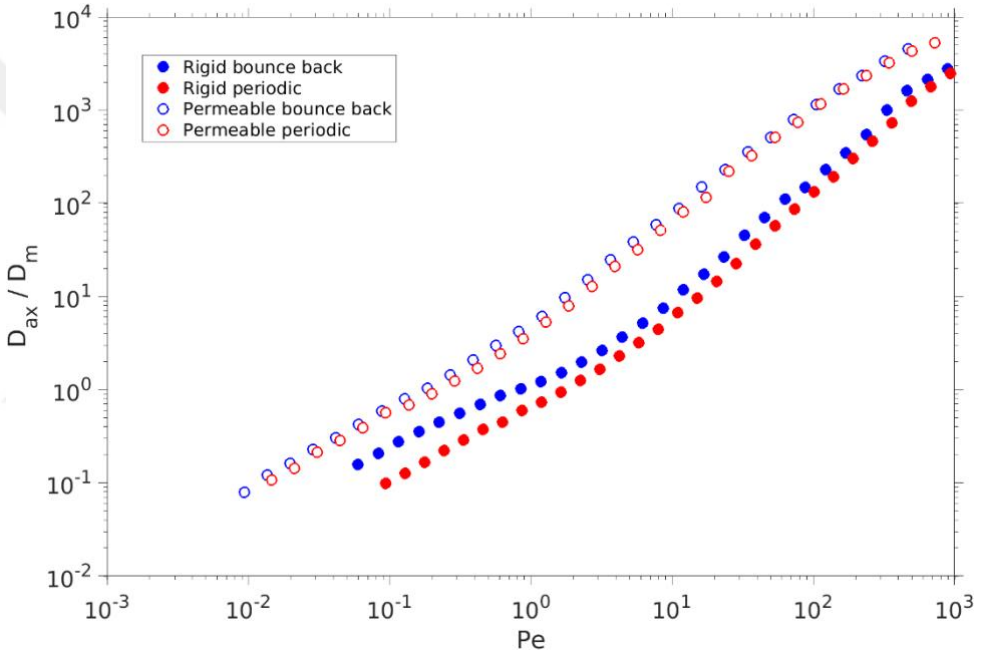


Figure 4.13. Simulation of dispersion on the four system configurations using the random walk method with 4900 tracers, indicating higher dispersion for the geometry with permeable particles.

There are a total of 30 data points corresponding to 30 random-walk simulations for each of the four systems, which sums up to 120 runs. This amount of computation took approximately 15 days on a 3.4 GHz, 6-core Intel processor, including the LBM simulations. The resulted curve of dispersion as a function of the Péclet number in Figure 4.13 is similar in shape to the theoretical prediction as well as the findings in

the literature as discussed later. There are two comments to be made regarding the presented data in the resulting dispersion plots. First, it is observed that dispersion in the permeable-spheres system is higher than in the rigid-spheres system. This is due to the existence of stagnant zones in the intraparticle pore space where the velocities are practically zero in such regions (see Figure 4.7). The effect of stagnant zones on dispersion in porous media has been analyzed in a dedicated study (Kandhai *et al.*, 2002). It was shown earlier that both the permeability as well as the average velocity of the permeable-spheres system were higher than the rigid-spheres system, as a result of the LBM simulation. These two observations combine to confirm the trade-off in chromatographic stationary phase development, between permeability and separation quality in terms of dispersion. The second comment is about the close similarity between the values obtained with periodic and those with bounce-back boundary conditions.

To be more precise, the application of the periodicity scheme is more effective in rigid-sphere systems than in permeable-sphere systems. One can settle with a conclusion that the effect of periodicity is small enough to completely neglect it, and to attribute the minor discrepancies to the stochastic nature of the random walk model. However, by looking closely at Figure 4.13, strictly all of the 30 data points emerging from the bounce-back simulations have higher dispersion than their counterparts. Therefore, it can be safely concluded that implementing periodic boundaries lead to, even slightly, lower values for dispersion.

Considering the calculated permeabilities and average velocities given in Table 4.2, the periodicity-based simulation gave a higher permeability and a higher average velocity which is straightforward to explain by the decreased flow resistance that would otherwise be imposed by the impermeable walls. Thus, the finding that periodic boundaries lead to both higher permeabilities, as well lower plate heights, is related to fact that tracers interact with the system boundaries in the same manner as the tracers far from the boundaries. Therefore, their migration through the system is probabilistically identical to the rest of the band. On the other hand, the existence

of impermeable walls at the lattice boundaries would effectively create an additional velocity gradient that intuitively enhances band broadening.

4.3.3 Comparison with Previous Literature Works

The results of the dispersion simulations obtained in this study were compared to similar studies in the literature, both numerical and experimental. Before presenting the comparisons, It is important to state that several variations and discrepancies were noticed as expected due to variations in the geometries themselves, operating conditions of the experiments, and methods of estimating certain parameters. All of these variations are discussed separately. The results of this work were first compared to a similar simulation study by (Koku & Lenhoff, 2012) as shown in Figure 4.14, where the main plot compares the calculated reduced plate height and the top-right inset plot compares the normalized dispersion coefficient.

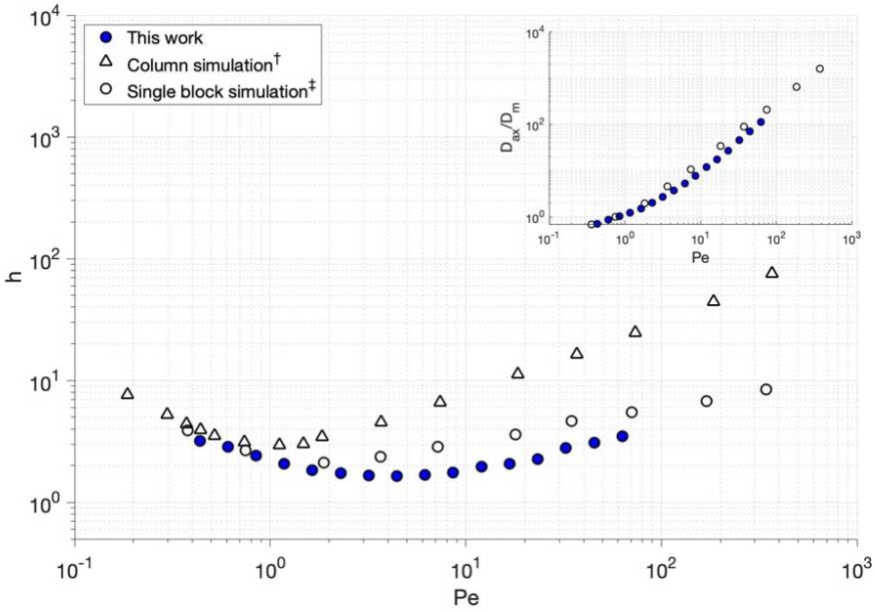


Figure 4.14. Comparison of the calculated dispersion coefficient in this work with the simulation of reconstructed polymer monolith by (†‡ Koku & Lenhoff, 2012). The main plot compares the plate height curves and the inset plot compares the normalized dispersion, as a function of varied Péclet numbers.

In their work, Koku & Lenhoff simulated a physical reconstruction of a polymer monolith using the random walk method, where they performed the simulation on a single block of the reconstructed domain as well as an extended version of several connected blocks that effectively resembled a chromatography column. The results of this work, therefore, were comparable to the single block simulation, which is shown as open circles in Figure 4.14. The slight discrepancies in this comparison can be safely attributed to differences in the morphology of both geometries such as porosity: 0.66 in our rigid-spheres geometry, and 0.57 in the reconstructed CIM monolith.

Next, the results of this work for the rigid spheres with periodic boundaries were compared to those of (Maier, 2000) where they simulated dispersion in a random packing of spheres using the random walk approach with a periodic boundary condition, in addition to experimental results on random sphere packs by (Gunn, 1969) as shown in Figure 4.15. The comparison show that the results of this simulation work on the rigid sphere system behaves in the same manner as those obtained by experiment on packed spheres.

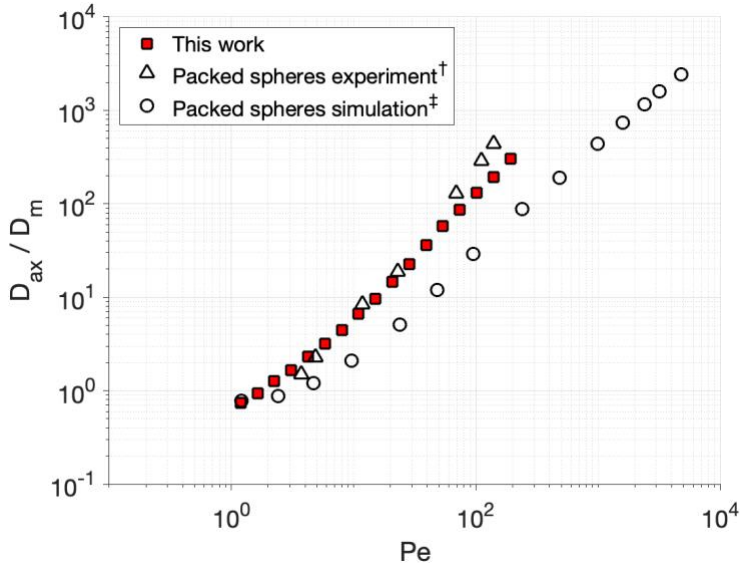


Figure 4.15. Comparison of axial dispersion results of this work on the rigid-sphere geometry to a random walk-based packed spheres simulation (‡Maier, 2000) and an experimental study on cubic sphere packing (†Gunn, 1969).

It is important to note that the Péclet number calculations may have been different in the compared results. In this work, the average velocity was used where the solid voxels, in terms of zero velocities, were excluded. However, both literature studies in Figure 4.15 did not reveal the method of velocity averaging, whether intrinsic or superficial. Therefore, Figure 4.15 assumes that all three data sets used intrinsic average velocities while calculating the Péclet numbers. If this assumption was invalid, the literature data would be shifted roughly by a factor of two ($1/\epsilon$) to the right. Regardless, the trend in Figure 4.15 shows a degree of similarity with experiments and a previous RWPT simulation with minor discrepancies that can safely be attributed to variances in the simulated structures and operating conditions.

The experimental study (Tallarek, 2003) of dispersion on silica-based monoliths was also considered for comparison with this work. An SEM image of the silica-based monolith is shown in Figure 4.16. Although the structures simulated in this work were made of perfect spheres, they share some characteristic parameters with the silica monolith simulated in Tallarek's study.

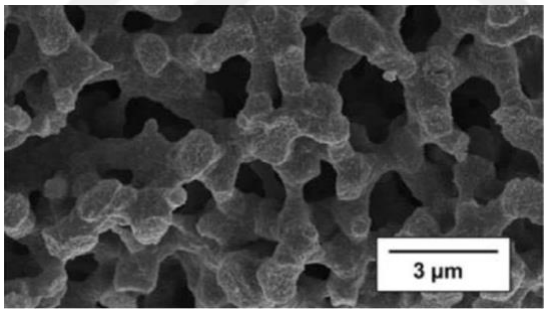


Figure 4.16 SEM image of a silica-based monolith adapted from (Tallarek, 2003), which was used in their study of dispersion.

These parameters are summarized in Table 4.5, where d_{macro} is the equivalent particle diameter calculated without considering the intraparticle mesopores, ϵ_{total} is the total porosity of the system, ϵ_{inter} is the porosity of the system without considering the mesopores, and ϵ_{intra} is the intra-particle porosity.

Table 4.5 Comparison of some characteristic parameters between the silica monolith used in Tallarek's study to the permeable-spheres system generated in this work.

Parameter	d_{macro} (μm)	ϵ_{total}	ϵ_{inter}	ϵ_{intra}
Silica monolith	1.9	0.92	0.72	0.70
Permeable-spheres geometry	1.68	0.89	0.66	0.70

In this work, the intraparticle porosity is the porosity of the pixel-based geometry shown in Figure 4.1. To compare the calculated dispersion curves of this work, the permeable-spheres system simulated with periodic boundaries was considered, and the comparison is shown in Figure 4.17.

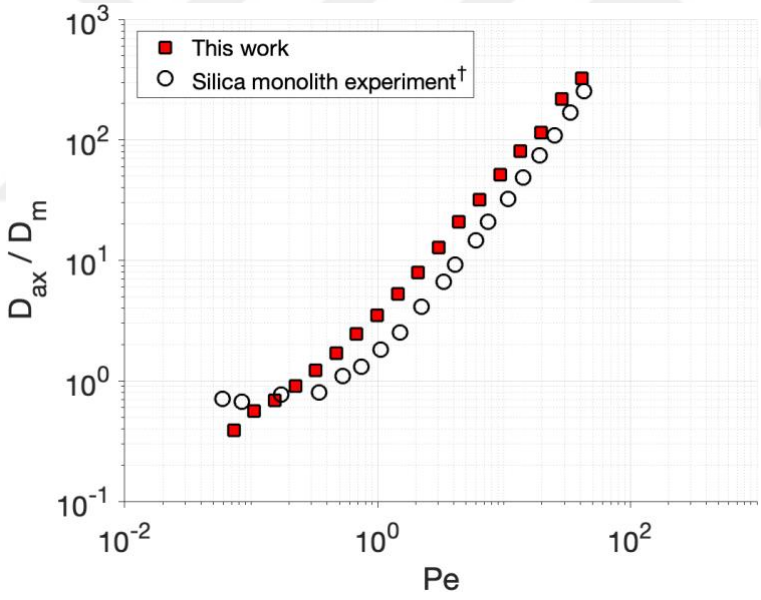


Figure 4.17. Dimensionless dispersion vs. Péclet number resulting from the random walk simulation on the permeable-spheres system, compared to experimental results on a commercial silica-based monolith (†Tallarek, 2003).

The Péclet number of the data set of this work in Figure 4.17 was calculated based on the superficial average velocity to be consistent with the results of the

experimental study. The first noticeable difference here is related to the behavior of the curve at low Péclet numbers. Dispersion in porous media is controlled totally by diffusion at very low Péclet numbers which means the normalized dispersion coefficient should, ideally, start at unity in a typical D_{ax}/D_m vs. Pe plot. However, in studies regarding dispersion in porous media, only a fraction of the bulk diffusion coefficient is recovered as the effective diffusion coefficient due to the tortuosity factor, which explains why the first few dispersion points are below unity in the diffusion-controlled regime.

From Figure 4.17, it can be inferred that the results of the simulation performed in this study show a similarity in the relationship between the Péclet number and dispersion to those obtained from the silica monolith experiment in the mechanical dispersion region (moderate to high Péclet numbers), but they differ at Péclet numbers below unity. Considering the entire range of results displayed in Figure 4.13, all of the four system configurations simulated in this work exhibited dimensionless dispersion reaching as low as 0.1, corresponding to Péclet numbers as low as 0.001 which is low enough that it falls below the operating range of a typical stationary phase (See Table 4.3 and Table 4.4).

A way to explain this can be by considering the differences in process conditions that directly affect diffusion in porous media, such as temperature, molecular interactions, or the presence of external fields, but these factors are not relevant to this simulation. Therefore, a more convenient explanation for Figure 4.17, considering the diffusion-controlled region, is that the simulated system in this work has a lower effective diffusion coefficient than the silica-based monolith, which suggests that the presence of solid obstacles is more frequent in the virtual geometry, for which the mesoporous space is highly fractal since it is made of individual cubic voxels, leading to a high degree of hindrance to diffusion. For the hydraulic dispersion-controlled region, above Péclet number of 1, the discrepancy appears systematic, and the simulation results show higher axial dispersion for nearly all data points. This can be explained by the differences in particle size distribution.

In a study by (Han, 1985) on the effect of particle size distribution on dispersion, it was revealed by experiment that mechanical dispersion in media of uniform particles is slightly higher than in media with a non-uniform size distribution if hydraulic radii were kept identical. As discussed in the previous sections, the raw sphere-based geometry constructed in this work is constituted of identical spherical particles, therefore, it is expected to see some differences when a comparison is made with a system of a non-uniform PSD. Figure 4.18 shows a comparison of the results of this work with the experimental results reported in (Han, 1985) on packed beds with uniform and non-uniform particle size distributions. As expected, the results of this work (rigid-spheres system) are closer to the results of the experiment on packed beds with uniform particle sizes.

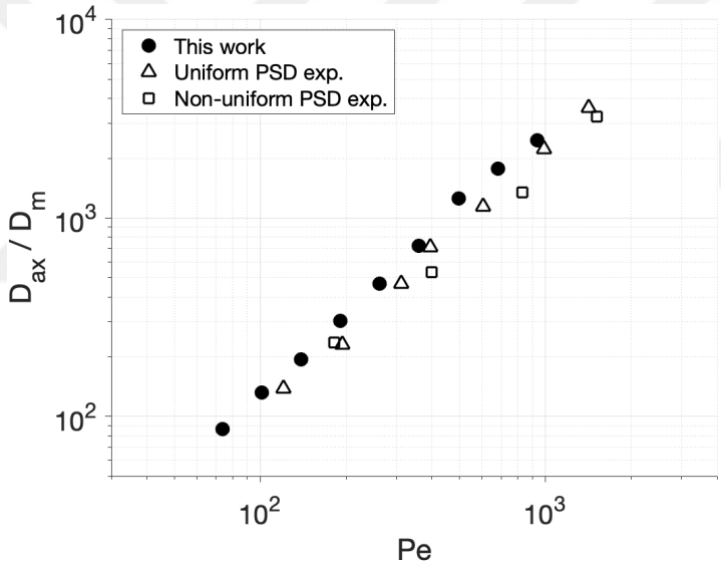


Figure 4.18. Comparison of the mechanical dispersion-controlled regime between the simulation on the rigid-spheres system with packed beds experiments (Han, 1985) with uniform and non-uniform particle size distributions.

4.3.4 On Asymptotic Dispersion

So far, an extended comparison was discussed between the results of this work and a couple of experimental and simulation results from the literature of similar studies.

Close similarities were observed in terms of curves behaviour that comply with the theoretical expectations, which indicates a reasonable degree of reliability of the model presented in this study, from the construction of tunable virtual porous media to the simulation of dispersion by the random walk particle tracking method. However, some discrepancies were addressed separately in each comparison. A portion of such discrepancies can also be discussed collectively by investigating the time evolution of dispersion (see Chapter 2). In Figure 4.19, axial dispersion is plotted with time for the rigid-spheres system (rigid bounce-back and rigid periodic) at two Péclet numbers for each configuration. Asymptotic dispersion may or may not be reached depending on various factors, such as the existence of stagnant zones due to mesoporous particles, the fractal nature of the geometry and heterogeneities of the media (Hulin, 1994; Koch & Brady, 1987).

Theoretically, the sampled solute particles need to laterally scan all preferential paths for asymptotic dispersion to be reached (Kandhai *et al.*, 2002). This is usually achieved at low Péclet numbers as the particles spend enough time to effectively sample all paths. The simulations performed in this work were not expected to deliver the asymptotic values of dispersion as that requires generating large geometries that are long enough for the simulated tracers to spend sufficient time to reach asymptotic behavior. Alternatively, the use of periodic repetition of the original matrix is another approach to investigate asymptotic dispersion. However, the latter approach is not realistic because it inevitably results in dead ends at the periodic boundaries. Nevertheless, this can be done by applying an axial periodic boundary condition and setting the solute exit point a few multiples of the original matrix size. Either of the two approaches; natural extension or periodic repetition, requires drastically added expenses in computational runtime and memory to store and analyse the time-based trajectories of the particles.

However, if the random walk code developed in this work is optimized with sufficient hardware to store the particle's axial positions at each time step, it might be worthwhile to repeat the simulations targeting asymptotic dispersion. Nevertheless, the time evolution of dispersion is plotted in Figure 4.19 as a proof of

concept. By looking at the plots, asymptotic dispersion was indeed reached for only two cases (at least as the plots show); that is, the rigid spheres with bounce-back, and the rigid spheres with periodic boundary conditions, both at their relatively low Peclet numbers.

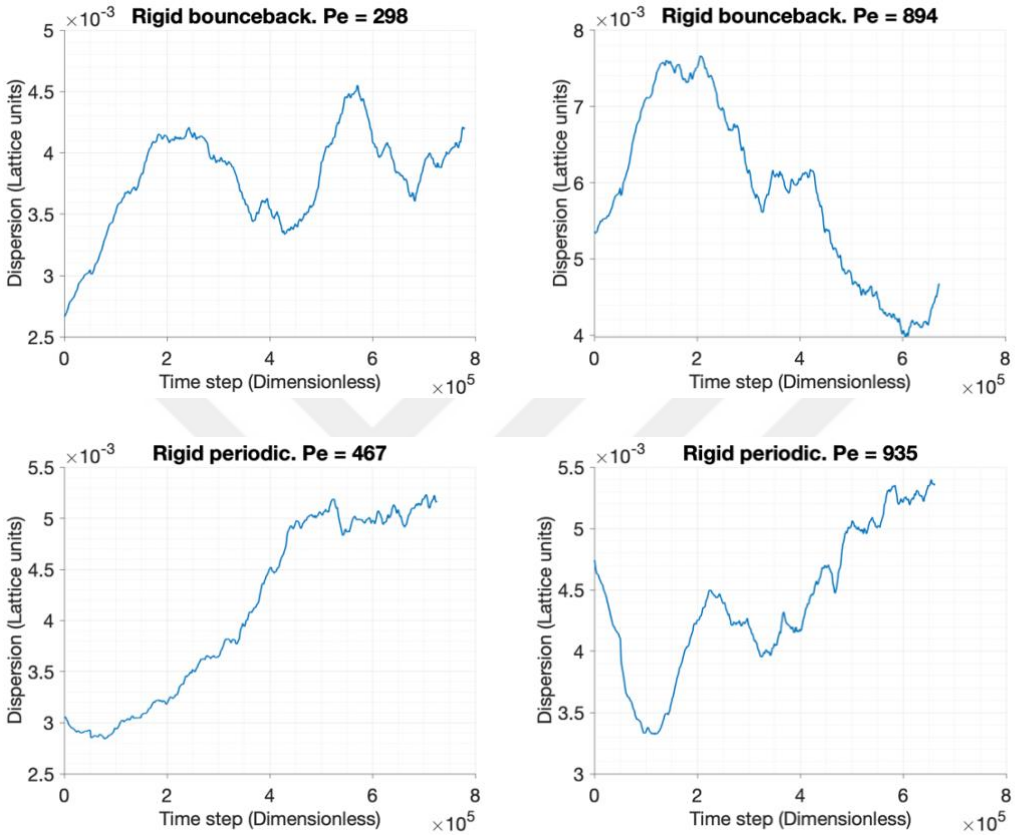


Figure 4.19. Time evolution of dispersion for the rigid-spheres system at different values of Péclet number. The units are all in lattice units and can be converted to physical units by considering the conversion factor for dispersion $C_D = 6.0 \times 10^{-6}\ m^2/s$ and the conversion factor for time $C_t = 1.0 \times 10^{-10}$ s. The dispersion data in all four plots were processed by a moving average function with a moving window of 10,000 timesteps.

This observation falls in line with the theoretical criterion of asymptotic dispersion, as the permeable-spheres system contains intraparticle stagnant zones that induce non-asymptotic behavior, and the fact that they were achieved at the low Péclet numbers also confirms the prediction that was mentioned above: The tracers were

given sufficient time to explore the preferential flow paths, which is not the case when they were exposed to high elution rates with high Péclet numbers as shown in the right column of Figure 4.19.

CHAPTER 5

CONCLUSIONS

In this work, a three-step model for modelling dispersion in porous media was presented. The strategy starts by virtual construction of bimodal porous system with an excellent degree of control over the desired macroscopic properties. Then the generated systems are flow-simulated using the Lattice Boltzmann Method and the complete velocity field was obtained, and the third step was the emulation of the displacement of a pulse injection of inert point tracers using a coupled diffusion-advection random walk tracking algorithm.

After the model was developed, the three steps discussed above were followed to simulate mass transfer in simple geometries for which analytical solutions are available. The results were found in excellent agreement. After the initial model verficaction is complete, a sphere-based geometry with a bimodal pore size distribution with large throughpores and small restricted mesopores, was generated to resemble silica monoliths. Then the system was simulated for flow and mass transfer. The results were calculated in terms of normalized dispersion coefficient vs Péclet number, and then compared well to a couple of similar studies in the literature.

Moreover, the constcructed geometry was also simulated with a monomodal pore size distribution; that is, impermeable particles. Two modes of boundary conditions were use; bounce-back and periodic boundary condtions. The comparsion with literature was found positive, confirming a degree of reliablity for the developed model. Some discerpenecies were encountered and were addressed with respect to the theoretical basis of the disperion phenomenon. The model aims to aid reachers for finding more efficieint designs by allowing them to perform the initial analysis virtually before investing in laboratory experiments.

CHAPTER 6

RECOMMENDATIONS

The model presented in this study still suffers from some limitations. The first, and most important one, is that there's a large requirement of hardware and software in order to construct and simulate realistic monolithic columns. For now, what the model can deal with is a small region from the middle of a system of realistic length. The assumption here is that the simulated partial geometry can sufficiently convey all macro and microscopic properties of the real system, except for the effect of the walls where higher stationary phase density is encountered in real columns. However, at least in theory, if enough computational power is available to the model, it can simulate a real system with an acceptable degree of accuracy.

Another important limitation is that the three main algorithms of this work can only operate in series, due to the nature of the problem that it is trying to solve. It is recommended for future work to unify the three algorithms in a single low-level programming language, and probably to write a new script to run all of the three steps at once by giving the input of the three steps initially. With this, one can also try to find an appropriate parallelization technique that could divide the system to be simulated into several segments and work on them in parallel with shared memory, similar to the segmentation method used in (Koku *et al.*, 2012b).

The presented model accounts for retention of the analyte with a given retention factor, where it uses a basic probability-based approach to determine adsorption then adds to the individual exit time accordingly (see Appendix E for a sample result). Further analysis of the retention phenomena by implementing advanced adsorption-desorption kinetics is also a possible continuation to this study. Moreover, modification of the code for finite-size tracers may be done depending on the molecular size and method of injection of the target application.

REFERENCES

Bear, J. (1972). Dynamics of Fluids in Porous Media (Dover Civil and Mechanical Engineering). *Published in 1972 Reprint in 1988 in New York NY) by Dover*, 784.

Bello, M. S., Rezzonico, R., & Righetti, P. G. (1994). Use of Taylor-Aris dispersion for measurement of a solute diffusion coefficient in thin capillaries. *Science*, *266*(5186), 773–776.

Bernsdorf, J., Brenner, G., & Durst, F. (2000). Numerical analysis of the pressure drop in porous media flow with lattice Boltzmann (BGK) automata. *Computer Physics Communications*, *129*(1–3), 247–255.

Bird, R. B., Stewart, W. E., & Lightfoot, E. N. (1960). Transport Phenomena. *Wiley New York*.

Bristow, P. A., & Knox, J. H. (1977). Standardization of test conditions for high performance liquid chromatography columns. *Chromatographia*, *10*(6), 279–289.

Cabrera, K., Lubda, D., Eggenweiler, H., Minakuchi, H., & Nakanishi, K. (2000). A new monolithic-type HPLC column for fast separations. *Journal of High Resolution Chromatography*, *23*(1), 93–99.

Carman, P. C. (1937). Fluid flow through granular beds. *Trans. Inst. Chem. Eng.*, *15*, 150–166.

Charlaix, E., Hulin, J. P., & Plona, T. J. (1987). Experimental study of tracer dispersion in sintered glass porous materials of variable compaction. *The Physics of Fluids*, *30*(6), 1690–1698.

Chatwin, P. C., & Sullivan, P. J. (1982). The effect of aspect ratio on longitudinal diffusivity in rectangular channels. *Journal of Fluid Mechanics*, *120*, 347–358.

Chen, S., & Doolen, G. D. (1998). Lattice boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, *30*, 329–364. https://doi.org/10.1146/ANNUREV.FLUID.30.1.329

Czok, M., & Guiochon, G. (1990). Comparison of the results obtained with different models for the simulation of preparative chromatography. *Computers & Chemical Engineering*, *14*(12), 1435–1443.

Delerue, J. F., Perrier, E., Yu, Z. Y., & Velde, B. (1999). New algorithms in 3D image analysis and their application to the measurement of a spatialized pore size distribution in soils. *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy*, *24*(7), 639–644.

Einstein, A. (1905). On the motion of small particles suspended in liquids at rest required by the molecular-kinetic theory of heat. *Annalen Der Physik*, *17*(549–560), 208.

Gardiner, C. W. (1985). *Handbook of stochastic methods* (Vol. 3). springer Berlin.

Gelhar, L. W., Welty, C., & Rehfeldt, K. R. (n.d.). A critical review of data on field-scale dispersion in aquifers. *Water Resources Research*, *28*(7), 1955–1974. Retrieved October 25, 2021, from https://www.academia.edu/28388251/A_critical_review_of_data_on_field_scale_dispersion_in_aquifers

Giddings, J. Caluin. (2017). Dynamics of chromatography: Principles and theory. In *Dynamics of Chromatography: Principles and Theory*. https://doi.org/10.1201/9781315275871

Giddings, J Calvin. (1965). Dynamics of Chromatography Marcel Dekker. *Inc., New York.*

Giddings, J Calvin, Gudzinowicz, B. J., Snyder, L. R., Kaiser, R., & DEKKER, M. (1965). Chromatographic Science. *Dynamics of Chromatography Part I Principles and Theory*.

Guiochon, G. (2006). The limits of the separation power of unidimensional column liquid chromatography. *Journal of Chromatography A*, *1126*(1–2), 6–49.

Guiochon, G. (2007a). Monolithic columns in high-performance liquid chromatography. *Journal of Chromatography A*, *1168*(1–2), 101–168. https://doi.org/10.1016/j.chroma.2007.05.090

Guiochon, G. (2007b). Monolithic columns in high-performance liquid chromatography. *Journal of Chromatography A*, *1168*(1–2), 101–168.

Guiochon, G., Felinger, A., Shirazi, D. G., & Katti, A. M. (2006). *Fundamentals of Preparative Chomatography*. 990.

He, X., & Luo, L. S. (1997). Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, *55*(6), 6811–6820. https://doi.org/10.1103/PHYSREVE.56.6811

Hlushkou, D., Bruns, S., Ho, A., & Tallarek, U. (2010). *From Pore Scale to Column Scale Dispersion in Capillary Silica Monoliths*. *82*(17), 7150–7159.

Hlushkou, D., Bruns, S., & Tallarek, U. (2010). High-performance computing of flow and transport in physically reconstructed silica monoliths. *Journal of Chromatography A*, *1217*(23), 3674–3682. https://doi.org/10.1016/j.chroma.2010.04.004

Horvath, C., & Lin, H.-J. (1976). Movement and band spreading of unsorbed solutes in liquid chromatography. *Journal of Chromatography A*, *126*, 401–420.

Hulin, J. P. (1994). Porous media: A model system for the physics of disorder. *Advances in Colloid and Interface Science*, *49*, 47–84.

Ito, K., Itô, K., Itô, K., Mathématicien, J., Itô, K., & Mathematician, J. (1951). *On stochastic differential equations* (Vol. 4). American Mathematical Society New York.

Josić, D., & Štrancar, A. (1999). Application of membranes and compact, porous units for the separation of biopolymers. *Industrial & Engineering Chemistry Research*, *38*(2), 333–342.

Junk, M., & Klar, A. (2000). Discretizations for the incompressible Navier-Stokes equations based on the lattice Boltzmann method. *SIAM Journal on Scientific Computing*, *22*(1), 1–19. https://doi.org/10.1137/S1064827599357188

Kandhai, D. (2002). Numerical simulation and measurement of liquid hold-up in biporous media containing discrete stagnant zones. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, *360*(1792), 521–534.

Kandhai, D., Kandhai, D., Hlushkou, D., Hlushkou, D., Hoekstra, A. G., Sloot, P. M., Van As, H., Tallarek, U., & Tallarek, U. (2002). Influence of stagnant zones on transient and asymptotic dispersion in macroscopically homogeneous porous media. *Physical Review Letters*, *88*(23), 2345011–2345014. https://doi.org/10.1103/PhysRevLett.88.234501

Kinzelbach, W., & Uffink, G. (1991). The random walk method and extensions in groundwater modelling. In *Transport processes in porous media* (pp. 761–787). Springer.

Knox, J H_, & Saleem, M. (1969). Kinetic conditions for optimum speed and resolution in column chromatography. *Journal of Chromatographic Science*, *7*(10), 614–622.

Knox, John H. (1977). Practical aspects of LC theory. *Journal of Chromatographic Science*, *15*(9), 352–364.

Koch, D. L., & Brady, J. F. (1985). Dispersion in fixed beds. *Journal of Fluid Mechanics*, *154*, 399–427.

Koch, D. L., & Brady, J. F. (1987). Nonlocal dispersion in porous media: Nonmechanical effects. *Chemical Engineering Science*, *42*(6), 1377–1392. https://doi.org/10.1016/0009-2509(87)85010-8

Koku, H., Maier, R. S., Czymmek, K. J., Schure, M. R., & Lenhoff, A. M. (2011). Modeling of flow in a polymeric chromatographic monolith. *Journal of Chromatography A*, *1218*(22), 3466–3475. https://doi.org/10.1016/j.chroma.2011.03.064

Koku, H., Maier, R. S., Schure, M. R., & Lenhoff, A. M. (2012a). Modeling of

dispersion in a polymeric chromatographic monolith. *Journal of Chromatography A*, *1237*, 55–63.

Koku, H., Maier, R. S., Schure, M. R., & Lenhoff, A. M. (2012b). Modeling of dispersion in a polymeric chromatographic monolith. *Journal of Chromatography A*, *1237*, 55–63. https://doi.org/10.1016/j.chroma.2012.03.005

Kolb, M. (1996). Models for Irreversible Gel Formation. *Polymer Gels and Networks*, *4*(5–6), 375–382. https://doi.org/10.1016/S0966-7822(96)00025-1

Kutay, M. E., Aydilek, A. H., & Masad, E. (2006). Laboratory validation of lattice Boltzmann method for modeling pore-scale flow in granular materials. *Computers and Geotechnics*, *33*(8), 381–395.

LaBolle, E. M., Fogg, G. E., & Tompson, A. F. B. (1996). Random-walk simulation of transport in heterogeneous porous media: Local mass-conservation problem and implementation methods. *Water Resources Research*, *32*(3), 583–593.

Langevin, P. (1908). On the theory of brownian motion. *CR Acad Sci (Paris)*, *146*, 530.

Langford, J. F., Schure, M. R., Yao, Y., Maloney, S. F., & Lenhoff, A. M. (2006). Effects of pore structure and molecular size on diffusion in chromatographic adsorbents. *Journal of Chromatography A*, *1126*(1–2), 95–106.

Latt, J., Malaspinas, O., Kontaxakis, D., Parmigiani, A., Lagrava, D., Brogi, F., Belgacem, M. Ben, Thorimbert, Y., Leclaire, S., Li, S., Marson, F., Lemus, J., Kotsalos, C., Conradin, R., Coreixas, C., Petkantchin, R., Raynaud, F., Beny, J., & Chopard, B. (2021). Palabos: Parallel Lattice Boltzmann Solver. *Computers and Mathematics with Applications*, *81*, 334–350. https://doi.org/10.1016/j.camwa.2020.03.022

Leinweber, F. C., & Tallarek, U. (2003). Chromatographic performance of monolithic and particulate stationary phases: Hydrodynamics and adsorption capacity. *Journal of Chromatography A*, *1006*(1–2), 207–228. https://doi.org/10.1016/S0021-9673(03)00391-1

Liapis, A. I., Meyers, J. J., & Crosser, O. K. (1999). Modeling and simulation of the dynamic behavior of monoliths: Effects of pore structure from pore network model analysis and comparison with columns packed with porous spherical particles. *Journal of Chromatography A*, *865*(1–2), 13–25.

Lubda, D., Cabrera, K., Kraas, W., Schaefer, C., & Cunningham, D. (2001). New developments in the application of monolithic HPLC columns. *LC GC North America*, *19*(12), 1186–1191.

Lubda, D., & Mueller, E. (2007). *Method for producing monolithic chromatography columns*. Google Patents.

Maier, R. S., Kroll, D. M., Bernard, R. S., Howington, S. E., Peters, J. F., & Davis, H. T. (2000a). Pore-scale simulation of dispersion. *Physics of Fluids*, *12*(8), 2065–2079. https://doi.org/10.1063/1.870452

Maier, R. S., Kroll, D. M., Bernard, R. S., Howington, S. E., Peters, J. F., & Davis, H. T. (2000b). Pore-scale simulation of dispersion. *Physics of Fluids*, *12*(8), 2065–2079.

Maier, R. S., Kroll, D. M., Bernard, R. S., Howington, S. E., Peters, J. F., & Davis, H. T. (2003). Hydrodynamic dispersion in confined packed beds. *Physics of Fluids*, *15*(12), 3795–3815.

Maier, R. S., Kroll, D. M., Davis, H. T., & Bernard, R. S. (1998). Pore-scale flow and dispersion. *International Journal of Modern Physics C*, *9*(08), 1523–1533.

Manwart, C., Aaltosalmi, U., Koponen, A., Hilfer, R., & Timonen, J. (2002). Lattice-Boltzmann and finite-difference simulations for the permeability for three-dimensional porous media. *Physical Review E*, *66*(1), 16702.

Manz, B., Gladden, L. F., & Warren, P. B. (1999). Flow and dispersion in porous media: Lattice-Boltzmann and NMR studies. *AIChE Journal*, *45*(9), 1845–1854.

McNamara, G. R., & Zanetti, G. (1988). Use of the Boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, *61*(20), 2332.

Meyers, J. J., & Liapis, A. I. (1999). Network modeling of the convective flow and diffusion of molecules adsorbing in monoliths and in porous particles packed in a chromatographic column. *Journal of Chromatography A*, *852*(1), 3–23.

Mihelič, I., Koloini, T., Podgornik, A., & Štrancar, A. (2000). Dynamic capacity studies of CIM (Convective Interaction Media)® monolithic columns. *Journal of High Resolution Chromatography*, *23*(1), 39–43.

Mihelič, I., Nemec, D., Podgornik, A., & Koloini, T. (2005). Pressure drop in CIM disk monolithic columns. *Journal of Chromatography A*, *1065*(1), 59–67.

Minakuchi, H., Nakanishi, K., Soga, N., Ishizuka, N., & Tanaka, N. (1997). Effect of skeleton size on the performance of octadecylsilylated continuous porous silica columns in reversed-phase liquid chromatography. *Journal of Chromatography A*, *762*(1–2), 135–146. https://doi.org/10.1016/S0021-9673(96)00944-2

Minakuchi, H., Nakanishi, K., Soga, N., Ishizuka, N., & Tanaka, N. (1998). Effect of domain size on the performance of octadecylsilylated continuous porous silica columns in reversed-phase liquid chromatography. *Journal of Chromatography A*, *797*(1–2), 121–131. https://doi.org/10.1016/S0021-9673(97)00947-3

Miyabe, K., Cavazzini, A., Gritti, F., Kele, M., & Guiochon, G. (2003). Moment

analysis of mass-transfer kinetics in C18-silica monolithic columns. *Analytical Chemistry*, *75*(24), 6975–6986.

Miyabe, K., & Guiochon, G. (2002). The moment equations of chromatography for monolithic stationary phases. *The Journal of Physical Chemistry B*, *106*(34), 8898–8909.

Moravcová, D., Jandera, P., Urban, J., & Planeta, J. (2004). Comparison of monolithic silica and polymethacrylate capillary columns for LC. *Journal of Separation Science*, *27*(10-11), 789–800.

Nakanishi, K., & Soga, N. (1991). Phase separation in gelling silica–organic polymer solution: systems containing poly (sodium styrenesulfonate). *Journal of the American Ceramic Society*, *74*(10), 2518–2530.

Nakanishi, K., & Soga, N. (1992a). Phase separation in silica sol-gel system containing polyacrylic acid I. Gel formaation behavior and effect of solvent composition. *Journal of Non-Crystalline Solids*, *139*, 1–13.

Nakanishi, K., & Soga, N. (1992b). Phase separation in silica sol-gel system containing polyacrylic acid II. Effects of molecular weight and temperature. *Journal of Non-Crystalline Solids*, *139*, 14–24.

Nakanishi, K., & Soga, N. (1997). *Inorganic porous material and process for making same*. Google Patents.

Neue, U. D. (1997). *{HPLC} columns: Theory, technology, and practice*. John Wiley & Sons.

Peters, E. C., Petro, M., Svec, F., & Fréchet, J. M. J. (1997). Molded rigid polymer monoliths as separation media for capillary electrochromatography. *Analytical Chemistry*, *69*(17), 3646–3649.

Rahmanian, M., & Kantzas, A. (2018). Stochastic generation of virtual porous media using a pseudo-crystallization approach. *Journal of Natural Gas Science and Engineering*, *53*(September 2017), 204–217. https://doi.org/10.1016/j.jngse.2018.02.016

Raoof, A., & Majid Hassanizadeh, S. (2010). A new method for generating pore-network models of porous media. *Transport in Porous Media*, *81*(3), 391–407. https://doi.org/10.1007/s11242-009-9412-3

Risken, H., & Risken, H. (1996). *Fokker-planck equation*. Springer.

Sahimi, M. (1993). Flow phenomena in rocks: from continuum models to fractals, percolation, cellular automata, and simulated annealing. *Reviews of Modern Physics*, *65*(4), 1393. https://doi.org/10.1103/RevModPhys.65.1393

Sahimi, M. (2011). *Flow and transport in porous media and fractured rock: from classical methods to modern approaches*. John Wiley & Sons.

Salles, J., Thovert, J., Delannay, R., Prevors, L., Auriault, J., & Adler, P. M. (1993). Taylor dispersion in porous media. Determination of the dispersion tensor. *Physics of Fluids A: Fluid Dynamics*, *5*(10), 2348–2376.

Senchenkova, E. M. (2003). *Michael Tswett-the creator of chromatography*.

Sinner, F., & Buchmeiser, M. R. (2000). A new class of continuous polymer supports prepared by ring-opening metathesis polymerization: a straightforward route to functionalized monoliths. *Macromolecules*, *33*(16), 5777–5786.

Skudas, R., Grimes, B. A., Thommes, M., & Unger, K. K. (2009). Flow-through pore characteristics of monolithic silicas and their impact on column performance in high-performance liquid chromatography. *Journal of Chromatography A*, *1216*(13), 2625–2636.

Succi, S. (2001). *The lattice Boltzmann equation: for fluid dynamics and beyond*. Oxford university press.

Svec, F., & Frechet, J. M. J. (1996). New designs of macroporous polymers and supports: from separation to biocatalysis. *Science*, *273*(5272), 205–211.

Svec, F., & Fréchet, J. M. J. (1992). Continuous rods of macroporous polymer as high-performance liquid chromatography separation media. *Analytical Chemistry*, *64*(7), 820–822.

Svec, F., & Lv, Y. (2015). Advances and recent trends in the field of monolithic columns for chromatography. *Analytical Chemistry*, *87*(1), 250–273. https://doi.org/10.1021/ac504059c

Tallarek, U., Hlushkou, D., Rybka, J., & Höltzel, A. (2019). Multiscale Simulation of Diffusion in Porous Media: From Interfacial Dynamics to Hierarchical Porosity. *Journal of Physical Chemistry C*, *123*(24), 15099–15112. https://doi.org/10.1021/acs.jpcc.9b03250

Tanaka, N., Kobayashi, H., Ishizuka, N., Minakuchi, H., Nakanishi, K., Hosoya, K., & Ikegami, T. (2002). Monolithic silica columns for high-efficiency chromatographic separations. *Journal of Chromatography A*, *965*(1–2), 35–49. https://doi.org/10.1016/S0021-9673(01)01582-5

Tanaka, N., Nagayama, H., Kobayashi, H., Ikegami, T., Hosoya, K., Ishizuka, N., Minakuchi, H., Nakanishi, K., Cabrera, K., & Lubda, D. (2000). Monolithic silica columns for HPLC, micro-HPLC, and CEC. *Journal of High Resolution Chromatography*, *23*(1), 111–116.

Taylor, G. I. (1953). Dispersion of soluble matter in solvent flowing slowly through a tube. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, *219*(1137), 186–203. https://doi.org/10.1098/rspa.1953.0139

Thomson, D. J., & Montgomery, M. R. (1994). Reflection boundary conditions for

random walk models of dispersion in non-Gaussian turbulence. *Atmospheric Environment*, *28*(12), 1981–1987.

Trilisky, E. I., & Lenhoff, A. M. (2009). Flow-dependent entrapment of large bioparticles in porous process media. *Biotechnology and Bioengineering*, *104*(1), 127–133.

TSWETT, M. S. (1954). Obshch. Estestvoistpyt., Otd. Biol., 14 (1903, publ. 1905) 20. *On the New Category of Adsorption Phenomena and Their Applications in Biochemical Analysi, Reprinted and Translated in G. Hesse and H. Weil," Michael Tswett's Erste Chromatographische Schrift"*.

Van Deemter, J. J., Zuiderweg, F. J., & Klinkenberg, A. van. (1956). Longitudinal diffusion and resistance to mass transfer as causes of nonideality in chromatography. *Chemical Engineering Science*, *5*(6), 271–289.

Vervoort, N., Saito, H., Nakanishi, K., & Desmet, G. (2005). Experimental validation of the tetrahedral skeleton model pressure drop correlation for silica monoliths and the influence of column heterogeneity. *Analytical Chemistry*, *77*(13), 3986–3992.

Vodopivec, M., Podgornik, A., Berovič, M., & Štrancar, A. (2000). Application of convective interaction media (CIM®) disk monolithic columns for fast separation and monitoring of organic acids. *Journal of Chromatographic Science*, *38*(11), 489–495.

Wang, Q. C., Švec, F., & Fréchet, J. M. J. (1994). Reversed-phase chromatography of small molecules and peptides on a continuous rod of macroporous poly (styrene-co-divinylbenzene). *Journal of Chromatography A*, *669*(1–2), 230–235.

Wang, X., Barber, W. E., & Carr, P. W. (2006). A practical approach to maximizing peak capacity by using long columns packed with pellicular stationary phases for proteomic research. *Journal of Chromatography A*, *1107*(1–2), 139–151.

Wang, X., Stoll, D. R., Schellinger, A. P., & Carr, P. W. (2006). Peak capacity optimization of peptide separations in reversed-phase gradient elution chromatography: fixed column format. *Analytical Chemistry*, *78*(10), 3406–3416.

Xie, S., Allington, R. W., Svec, F., & Fréchet, J. M. J. (1999). Rapid reversed-phase separation of proteins and peptides using optimized 'moulded'monolithic poly (styrene–co-divinylbenzene) columns. *Journal of Chromatography A*, *865*(1–2), 169–174.

Yang, X., Ma, L., & Carr, P. W. (2005). High temperature fast chromatography of proteins using a silica-based stationary phase with greatly enhanced low pH stability. *Journal of Chromatography A*, *1079*(1–2), 213–220.

Yeong, C. L. Y., & Torquato, S. (1998). Reconstructing Random Media I and II.

Physical Review E, *58*(1), 224–233. https://journals.aps.org/pre/pdf/10.1103/PhysRevE.57.495

Zahedzadeh, M., ABBASI, S., Dadvar, M., & Shadizadeh, S. R. (2010). *An improved model to simulate mud (drilling fluid) dispersion through porous media*.

Цвет, М. С. (1903). О новой категории адсорбционных явлений и о применении их к биохимическому анализу. *Труды Варшавского Общества Естествоиспытателей. Отд. Биологии*, *14*, 1.

APPENDICES

A. Conversion of Units

This section illustrates the method of converting units between the simulation domain and the physical domain, by introducing the concept of conversion factors such that a conversion factor C_γ is expressed as

$$C_\gamma = \gamma/\gamma^* \tag{A.1}$$

where γ is the physical quantity and γ^* is the simulation (dimensionless) quantity. To explain how this concept was used, we follow five steps.

Step 1. Define the resolution of the simulation; that is, define C_L which is the conversion factor for the length (Similarly, one can start by defining C_t, the conversion factor for time). In the case of physical re-construction of porous media, this factor is already defined as the ratio of physical length (in micrometers) to the length of the virtual mesh (in pixels).

Step 2: Define the simulated mobile phase by its physical kinematic viscosity v. For water:

$$v = 1 \times 10^{-6}\ m^2/s \tag{A.2}$$

The equivalent lattice viscosity in LBM simulations is $v^* = 1/6$. Therefore, the conversion factor for viscosity becomes

$$C_v = v/v^* = \frac{10^{-6}}{1/6} \tag{A.3}$$

Step 3: The fluid density (in the LBM simulation) is equal to 1. Therefore the conversion factor for density is equal to the mobile phase density

$$C_\rho = \rho/1 = \rho \tag{A.4}$$

Step 4: Use the dimensionless Reynolds number to relate different quanitites, and to derive the desired conversion factors as functions of the three defined ones in Steps 1-3. This is possible because such dimensionless numbers have the same values both in the physical and the virtual domains.

$$\mathrm{Re} = \frac{\mathrm{u}\ \mathrm{d_e}}{\mathrm{v}} = \frac{\mathrm{u^*}\ \mathrm{d_e^*}}{\mathrm{v^*}} \tag{A.5}$$

From Equations (A.5) and (A.1), it follows that

$$\mathrm{C_u C_L} = \mathrm{C_v} \tag{A.6}$$

Step 5: Derive the rest of conversion factors. For example, the conversion factor for velocity is

$$\mathrm{C_u} = \mathrm{C_v}/\mathrm{C_L} \tag{A.7}$$

The conversion factor for time is

$$\mathrm{C_t} = \mathrm{C_L}/\mathrm{C_u} = \mathrm{C_L^2}/\mathrm{C_v} \tag{A.8}$$

And for pressure

$$\mathrm{C_p} = \frac{\mathrm{C_\rho C_v}}{\mathrm{C_t}} = \frac{\mathrm{C_\rho C_v}}{\mathrm{C_L^2}/\mathrm{C_v}} = \frac{\mathrm{C_v^2 C_\rho}}{\mathrm{C_L^2}} \tag{A.9}$$

Note that the conversion factors for dispersion and diffusion, $\mathrm{C_D}$ are equvalent to that of the kinmetic viscosity, since they have the same units.

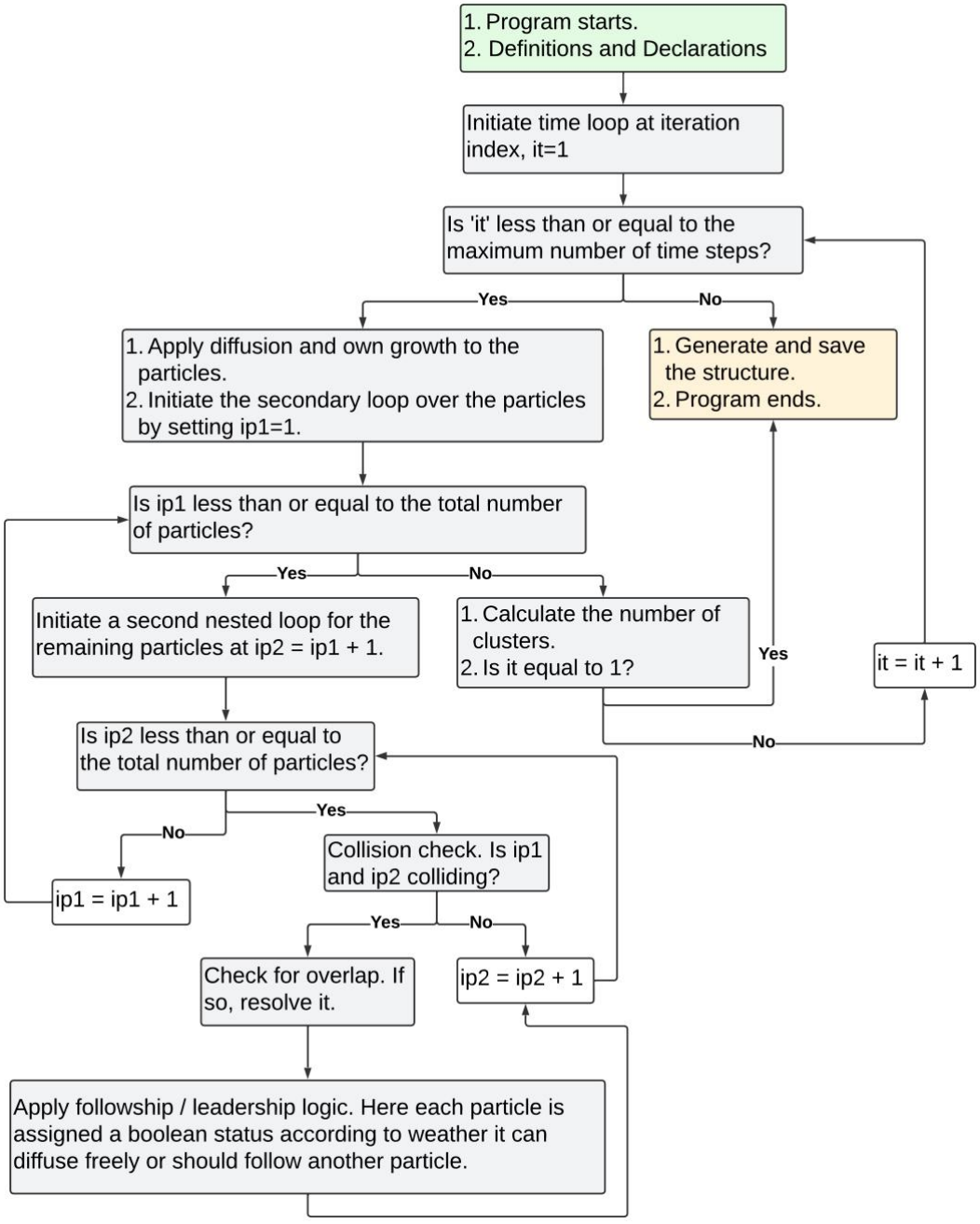
$$\mathrm{C_D} = \mathrm{C_v} \tag{A.10}$$

Alternatively, one may start by defining $\mathrm{C_t}$ instead of $\mathrm{C_L}$, and go through the same process. However, defining $\mathrm{C_L}$ is easier because one can choose it from the knowledge of the size of a typical physical geometry.

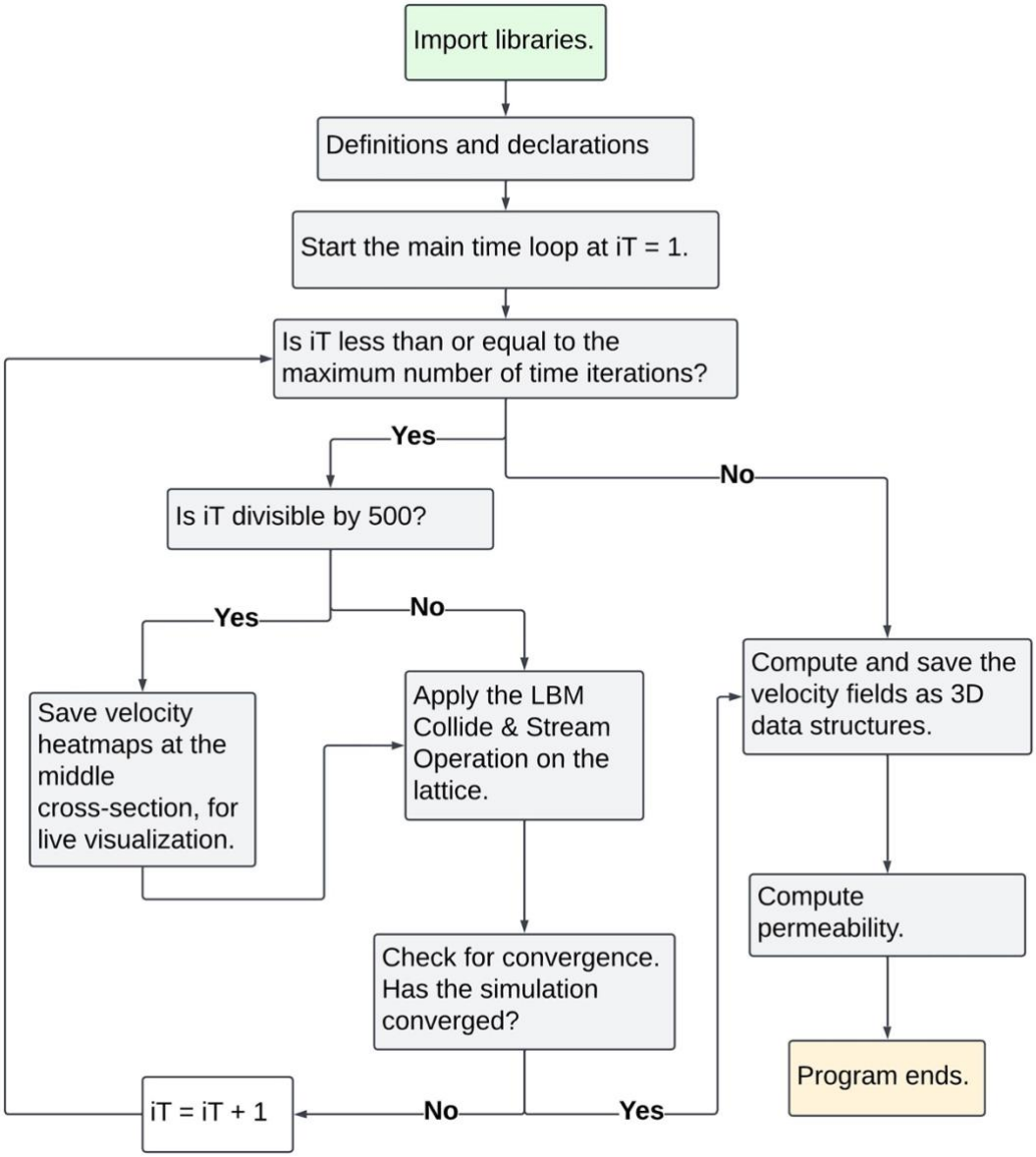
B. Algorithm Flowcharts

The algorithms for the geometry construction, LBM flow simulation and RWPT implementation are given below.

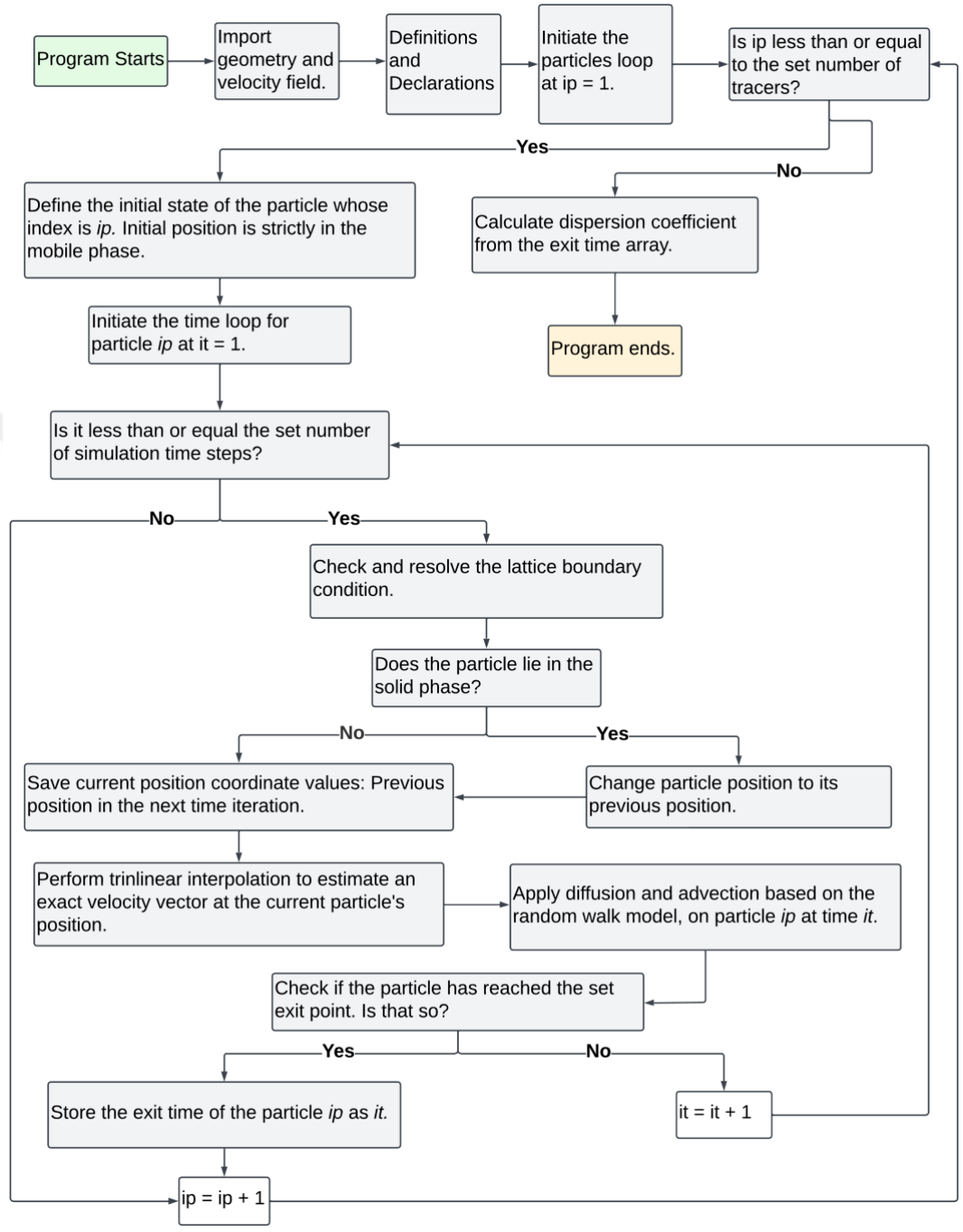
B.1 Construction of Geometries: Sphere-based Algorithm



B.2 Flow Simulation with PALABOS



B.3 Random Walk Particle Tracking



C. Code Scripts

C.1 Geometry Construction: Pixel-based Algorithm

Definitions and declarations:

```
clc; close all;
clear all

% Parameters
outfolder = '.\monolith_reaction\'; % Folder for output images
niter = 10000;   % Number of iterations
nrows = 200;   % Number of rows = columns
porogen = 0.7; % Fraction of pore pixels, i.e. 'porogen'
conn = 26;

basename = strcat('porogen',num2str(porogen*100),'_conn', num2str(conn),...
    '_size',num2str(nrows),'_iter',num2str(niter),'_');

output_freq = floor(niter/1000); % output result every niter/100 iterations
A = zeros(nrows,nrows,nrows);
B = zeros(nrows,nrows,nrows);

move_time = zeros(nrows,nrows,nrows);
sizeA = size(A);
next_post = zeros(nrows,nrows,nrows);
nelements = numel(A);

monomer = floor(numel(A)*(1-porogen)); % number of solid, or 'monomer', pixels
monomer_loc = randperm(nelements);

for k =1:monomer
    i_monomer(k) = monomer_loc(k);
end

A(i_monomer) = 1;
Ao = A;
A = logical(A);
C = zeros(nrows,nrows,nrows);
```

Main time loop:

```
for step = 1:niter
        % Save current image if output_freq iterations have occurred
    if mod(step,output_freq)==0
        fprintf(1,'iteration %5.0f of %5.0f\n',  step,niter);
    end
    [B, nclumps] = bwlabeln(A,conn); % Identify clumps
    if step==1
        disp('Initializing...')
        pause(0.1)
        fprintf(1,'initial number of clumps %10.0f\n', nclumps);
        output_freq2 = floor(nelements/80000);
        for t = 1:nclumps
            indices_clump_t = find(B==t);
            length_clump_t = length(indices_clump_t);
            C(indices_clump_t)=length_clump_t;
            if mod(t,output_freq2)==0
                fprintf(1,'iteration %5.0f of %5.0f\n',  t,nclumps);
            end
        end
        move_time = C;
```

```
next_post(find(A))= find(A);
        clear C
        disp('Done! Starting iterations...')
    end
    % Process individual clumps
    for cm = 1:nclumps
        cm;
        % equalize move times, change indices to subscript form
        clump_elements = find(B==cm);
        clump_cm_k = ceil(clump_elements/(nrows*nrows));
        sliceindex = clump_elements - (clump_cm_k-1)*nrows*nrows;
        clump_cm_j = ceil(sliceindex/nrows);
        clump_cm_i = sliceindex - (clump_cm_j-1)*nrows;
        move_time(clump_elements) = round(mean(move_time(clump_elements)));
        % Move if clump move time is down to zero
        if move_time(clump_elements(1))==0
            delta_cm_i = round(2*rand-1);
            delta_cm_j = round(2*rand-1);
            delta_cm_k = round(2*rand-1);
            % Determine next position of each element in clump
            next_i = clump_cm_i + delta_cm_i;
            next_j = clump_cm_j + delta_cm_j;
            next_k = clump_cm_k + delta_cm_k;
            if min(next_i) < 1 || max(next_i)>nrows % enforce horizontal borders
                next_i = clump_cm_i;
                next_j = clump_cm_j;
                next_k = clump_cm_k;
            end
            if min(next_j) < 1 || max(next_j)>nrows % enforce vertical borders
                next_j = clump_cm_j;
                next_i = clump_cm_i;
                next_k = clump_cm_k;
            end
            if min(next_k) < 1 || max(next_k)>nrows % enforce slice borders
                next_k = clump_cm_k;
                next_i = clump_cm_i;
                next_j = clump_cm_j;
            end
            for m =1:length(clump_elements)
                next_post(clump_cm_i(m),clump_cm_j(m),clump_cm_k(m))=...
                    next_i(m)+(next_j(m)-1)*nrows + (next_k(m)-1)*nrows*nrows;
            end
            % Move clump if all target regions are empty, reset timer for clump
            A(clump_elements) = 0; % Zero all clump locations, will recover
            % if movement is cancelled
            if nnz(A(next_post(clump_elements)))==0
                %disp('Moving')
                for m =1:length(clump_elements)
                    A(next_post(clump_cm_i(m), clump_cm_j(m),clump_cm_k(m)))=1;
                    move_time(next_post(clump_elements(m))) =
length(clump_elements); % reset timer
                end
            else
                A(clump_elements)=1; % Reset clump in case there's no motion
            end
            % If move time is not zero yet, tick timer
        else
            move_time(clump_elements) = move_time(clump_elements) - 1;
        end
    end
end
% FINALIZE AND OUTPUT RESULTS
[B, nclumps] = bwlabeln(A,conn);
C = zeros(nrows,nrows,nrows);
for t = 1:nclumps
    indices_clump_t = find(B==t);
    length_clump_t = length(indices_clump_t);
    C(indices_clump_t)=length_clump_t;
    if mod(t,output_freq2)==0
        fprintf(1,'iteration %5.0f of %5.0f\n',  t,nclumps);
```

C.2 Geometry Construction: Sphere-based algorithm

Definitions and declarations:

```
clc;
close all;
tic;
lattice_size = 10;
num_monomers = 1000;
own_growth_factor = 0.02;
base_radius = 0.02*lattice_size;
rho = 0.70; % Inital target porosity
base_dm = 0.10*lattice_size; % size of the diffusion step
mesh_size = 200;
scale_factor = mesh_size/lattice_size; % Scale factor for generating final mesh
max_overlap_ratio = 1.25;
nsteps = 100000;
% Calculate initial porosity
solid_volume = num_monomers * (4/3 * pi * base_radius^3);
total_volume = lattice_size^3;
current_rho = 1 - (solid_volume / total_volume);
if current_rho <= rho
    error("Initial porosity: " + current_rho + " is less than target porosity");
end
clear solid_volume total_volume
disp("Initial Porosity = " + current_rho)
```

Preparation of the particles dataframe:

```
dataframe = table('Size', [num_monomers 10], 'VariableNames', {'ID',
'coordinates','num_followers', 'follower', 'direct_leader', 'top_leader',
'rel_pos', 'radius', 'volume', 'dm'},'VariableType', {'int8', 'double', 'int8',
'logical', 'int8', 'int8','double', 'double', 'double', 'double'});
% Initiate variables & append to the dataframe
id = (1:num_monomers)'; % particle ID
dataframe.ID = id;
coordinate_space = linspace(0+base_radius, lattice_size-base_radius, 500);
while 1 % Make the inital coordinates random and unique
    coordinates = coordinate_space(randi(numel(coordinate_space), [num_monomers,
3]));
    u_coordinates = unique(coordinates, 'rows', 'stable');
    if isequal(u_coordinates, coordinates)
        disp("finsihed initial configurations.");
        break
    end
end
clear u_coordinates
dataframe.coordinates = coordinates;
follower = false(num_monomers, 1);
dataframe.follower = follower;
num_followers = ones(num_monomers, 1);
dataframe.num_followers = num_followers;
direct_leader = zeros(num_monomers, 1);
dataframe.direct_leader = direct_leader;
dataframe.top_leader = dataframe.ID;
rel_pos = zeros(num_monomers, 3);
dataframe.rel_pos = rel_pos;
radius = base_radius + zeros(num_monomers, 1); % Starting radius
dataframe.radius = radius;
dm = base_dm + zeros(num_monomers, 1);
dataframe.dm = dm;
dataframe.volume = 4/3 * pi * dataframe.radius;
clear id coordinates follower num_followers top_leader direct_leader rel_pos
radius dm
```

Main loop for the aggregation and growth of the particles:

```
rand_list = [-1, 1];
print_rho = true;
num_chunks=num_monomers;
for it = 1:nsteps
    % Calculate current porosity
    dataframe.volume = 4/3 * pi * dataframe.radius.^3;
    solid_volume = sum(dataframe.volume);
    total_volume = lattice_size^3;
    current_rho = 1 - (solid_volume / total_volume);
    % Diffusion to leaders
    dataframe.coordinates(dataframe.follower==false, 1) =
dataframe.coordinates(dataframe.follower==false, 1) +
dataframe.dm(dataframe.follower==false).*rand_list(randi(2))./sqrt(dataframe.num
_followers(dataframe.follower==false));
    dataframe.coordinates(dataframe.follower==false, 2) =
dataframe.coordinates(dataframe.follower==false, 2) +
dataframe.dm(dataframe.follower==false).*rand_list(randi(2))./sqrt(dataframe.num
_followers(dataframe.follower==false));
    dataframe.coordinates(dataframe.follower==false, 3) =
dataframe.coordinates(dataframe.follower==false, 3) +
dataframe.dm(dataframe.follower==false).*rand_list(randi(2))./sqrt(dataframe.num
_followers(dataframe.follower==false));
    % Followship to followers
    df_followers = dataframe(dataframe.follower==true,:);
    dataframe.coordinates(dataframe.follower==true, :) =
dataframe.coordinates(df_followers.direct_leader, :) - df_followers.rel_pos(:,
:);
    % Periodic boundaries
    % > lattice_size
    dataframe.coordinates(dataframe.coordinates(:,1)>lattice_size, 1) = 0 +
(dataframe.coordinates(dataframe.coordinates(:,1)>lattice_size, 1) -
lattice_size);
    dataframe.coordinates(dataframe.coordinates(:,2)>lattice_size, 2) = 0 +
(dataframe.coordinates(dataframe.coordinates(:,2)>lattice_size, 2) -
lattice_size);
    dataframe.coordinates(dataframe.coordinates(:,3)>lattice_size, 3) = 0 +
(dataframe.coordinates(dataframe.coordinates(:,3)>lattice_size, 3) -
lattice_size);
    %  < 0
    dataframe.coordinates(dataframe.coordinates(:,1)<0, 1) = lattice_size +
dataframe.coordinates(dataframe.coordinates(:,1)<0, 1);
    dataframe.coordinates(dataframe.coordinates(:,2)<0, 2) = lattice_size +
dataframe.coordinates(dataframe.coordinates(:,2)<0, 2);
    dataframe.coordinates(dataframe.coordinates(:,3)<0, 3) = lattice_size +
dataframe.coordinates(dataframe.coordinates(:,3)<0, 3);
leader_pairs = nchoosek(dataframe.ID(dataframe.follower==false),2);
    leader_pairs = nchoosek(dataframe.ID(dataframe.follower==false),2);
    inflated_monomers = zeros(num_monomers,1);
    for i_leaders = 1:numel(leader_pairs(:,1))
        [L1,L2] =
meshgrid(dataframe.ID(dataframe.top_leader==leader_pairs(i_leaders, 1)),
dataframe.ID(dataframe.top_leader==leader_pairs(i_leaders, 2)));
        all_pairs = [L1(:), L2(:)];
        % particle's first loop
        for ip = 1:numel(all_pairs(:,1))
            ip1 = all_pairs(ip,1);
            ip2 = all_pairs(ip,2);
            % Apply own growth
            if current_rho > rho
                if ~ismember(ip1, inflated_monomers)
                    dataframe.radius(ip1) = (own_growth_factor+1) *
dataframe.radius(ip1);
                    inflated_monomers(ip1)=ip1;
                end
                if ~ismember(ip2, inflated_monomers)
```

```
                    inflated_monomers(ip2)=ip2;
                end
            elseif current_rho <= rho && print_rho == true
                print_rho = false;
                disp('Target porosity is exceeded. Stopping own growth.');
            end
            % Check for collision
            x_1 = dataframe.coordinates(ip1, 1);
            x_2 = dataframe.coordinates(ip2, 1);
            y_1 = dataframe.coordinates(ip1, 2);
            y_2 = dataframe.coordinates(ip2, 2);
            z_1 = dataframe.coordinates(ip1, 3);
            z_2 = dataframe.coordinates(ip2, 3);
            distance = sqrt((x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2);
            if distance <= (dataframe.radius(ip1)+dataframe.radius(ip2))
                % Overlap resolver
                sum_radii = (dataframe.radius(ip1)+dataframe.radius(ip2));
                overlap_ratio = sum_radii / distance;
                if overlap_ratio >= max_overlap_ratio
                    overlap_delta = sum_radii - distance;
                    target_displacement = 0.499 * overlap_delta;
                    if x_1 ~= x_2
                        alpha_yx = ((y_1-y_2)/(x_1-x_2))^2;
                        alpha_zx = ((z_1-z_2)/(x_1-x_2))^2;
                    end
                    if y_1 ~= y_2
                        alpha_xy = ((x_1-x_2)/(y_1-y_2))^2;
                        alpha_zy = ((z_1-z_2)/(y_1-y_2))^2;
                    end
                    if z_1 ~= z_2
                        alpha_xz = ((x_1-x_2)/(z_1-z_2))^2;
                        alpha_yz = ((y_1-y_2)/(z_1-z_2))^2;
                    end
                    if x_2 > x_1
                        dataframe.coordinates(ip1,1) = x_1 - target_displacement
/ sqrt(1+alpha_yx+alpha_zx);
                        dataframe.coordinates(ip2,1) = x_2 + target_displacement
/ sqrt(1+alpha_yx+alpha_zx);
                    elseif x_2 < x_1
                        dataframe.coordinates(ip1,1) = x_1 + target_displacement
/ sqrt(1+alpha_yx+alpha_zx);
                        dataframe.coordinates(ip2,1) = x_2 - target_displacement
/ sqrt(1+alpha_yx+alpha_zx);
                    end
                    if y_2 > y_1
                        dataframe.coordinates(ip1,2) = y_1 - target_displacement
/ sqrt(alpha_xy+1+alpha_zy);
                        dataframe.coordinates(ip2,2) = y_2 + target_displacement
/ sqrt(alpha_xy+1+alpha_zy);
                    elseif y_2 < y_1
                        dataframe.coordinates(ip1,2) = y_1 + target_displacement
/ sqrt(alpha_xy+1+alpha_zy);
                        dataframe.coordinates(ip2,2) = y_2 - target_displacement
/ sqrt(alpha_xy+1+alpha_zy);
                    end
                    if z_2 > z_1
                        dataframe.coordinates(ip1,3) = z_1 - target_displacement
/ sqrt(alpha_xz+alpha_yz+1);
                        dataframe.coordinates(ip2,3) = z_2 + target_displacement
/ sqrt(alpha_xz+alpha_yz+1);
                    elseif z_2 < z_1
                        dataframe.coordinates(ip1,3) = z_1 + target_displacement
/ sqrt(alpha_xz+alpha_yz+1);
                        dataframe.coordinates(ip2,3) = z_2 - target_displacement
/ sqrt(alpha_xz+alpha_yz+1);
                    end
                end
                % Apply followship logic
```

```
                    assumed_leader_2 = dataframe.direct_leader(ip2);
                    true_leader_1 = 0; true_leader_2 = 0;
                    while 1
                        if dataframe.follower(assumed_leader_1)
                            assumed_leader_1 =
dataframe.direct_leader(assumed_leader_1);
                        else
                            true_leader_1 = assumed_leader_1;
                        end
                        if dataframe.follower(assumed_leader_2)
                            assumed_leader_2 =
dataframe.direct_leader(assumed_leader_2);
                        else
                            true_leader_2 = assumed_leader_2;
                        end
                        if true_leader_1 > 0 && true_leader_2 > 0
                            supreme_leader = min([true_leader_1,
true_leader_2]);
                            other_leader = max([true_leader_1, true_leader_2]);
                            break
                        end
                    end
                    if supreme_leader ~= other_leader
                        x_s = dataframe.coordinates(supreme_leader, 1);
                        y_s = dataframe.coordinates(supreme_leader, 2);
                        z_s = dataframe.coordinates(supreme_leader, 3);
                        x_o = dataframe.coordinates(other_leader, 1);
                        y_o = dataframe.coordinates(other_leader, 2);
                        z_o = dataframe.coordinates(other_leader, 3);
                        dataframe.follower(other_leader) = true;
                        dataframe.direct_leader(other_leader) = supreme_leader;
                        dataframe.top_leader(other_leader) = supreme_leader;
                        for i_follow =
dataframe.ID(dataframe.direct_leader==other_leader)
                            dataframe.top_leader(i_follow) = supreme_leader;
                        end
                        for i_follow =
dataframe.ID(dataframe.top_leader==other_leader)
                            dataframe.top_leader(i_follow) = supreme_leader;
                        end
                        dataframe.rel_pos(other_leader, :) = [x_s-x_o, y_s-y_o,
z_s-z_o];
                        % Update cluster size
                        dataframe.num_followers(supreme_leader) =
dataframe.num_followers(supreme_leader) + dataframe.num_followers(other_leader);
                    end
                elseif dataframe.follower(ip2) && ~dataframe.follower(ip1)
                    assumed_leader_1 = ip1;
                    assumed_leader_2 = dataframe.direct_leader(ip2);
                    true_leader_1 = ip1; true_leader_2 = 0;
                    while 1
                        if dataframe.follower(assumed_leader_2)
                            assumed_leader_2 =
dataframe.direct_leader(assumed_leader_2);
                        else
                            true_leader_2 = assumed_leader_2;
                            supreme_leader = min([true_leader_1,
true_leader_2]);
                            other_leader = max([true_leader_1, true_leader_2]);
                            break
                        end
                    end
                    if supreme_leader ~= other_leader
                        x_s = dataframe.coordinates(supreme_leader, 1);
                        y_s = dataframe.coordinates(supreme_leader, 2);
                        z_s = dataframe.coordinates(supreme_leader, 3);
                        x_o = dataframe.coordinates(other_leader, 1);
                        y_o = dataframe.coordinates(other_leader, 2);
                        z_o = dataframe.coordinates(other_leader, 3);
                        dataframe.follower(other_leader) = true;
```

```
                        dataframe.top_leader(other_leader) = supreme_leader;
                        for i_follow =
dataframe.ID(dataframe.direct_leader==other_leader)
                            dataframe.top_leader(i_follow) = supreme_leader;
                        end
                        for i_follow =
dataframe.ID(dataframe.top_leader==other_leader)
                            dataframe.top_leader(i_follow) = supreme_leader;
                        end
                        dataframe.rel_pos(other_leader, :) = [x_s-x_o, y_s-y_o,
z_s-z_o];
                        % 3. Update chunk size
                        dataframe.num_followers(supreme_leader) =
dataframe.num_followers(supreme_leader) + dataframe.num_followers(other_leader);
                    end
                elseif ~dataframe.follower(ip2) && ~dataframe.follower(ip1)
                    true_leader_1 = ip2; true_leader_2 = ip1;
                    supreme_leader = min([true_leader_1, true_leader_2]);
                    other_leader = max([true_leader_1, true_leader_2]);
                    if supreme_leader ~= other_leader
                        x_s = dataframe.coordinates(supreme_leader, 1);
                        y_s = dataframe.coordinates(supreme_leader, 2);
                        z_s = dataframe.coordinates(supreme_leader, 3);
                        x_o = dataframe.coordinates(other_leader, 1);
                        y_o = dataframe.coordinates(other_leader, 2);
                        z_o = dataframe.coordinates(other_leader, 3);
                        dataframe.follower(other_leader) = true;
                        dataframe.direct_leader(other_leader) = supreme_leader;
                        dataframe.top_leader(other_leader) = supreme_leader;
                        for i_follow =
dataframe.ID(dataframe.direct_leader==other_leader)
                            dataframe.top_leader(i_follow) = supreme_leader;
                        end
                        for i_follow =
dataframe.ID(dataframe.top_leader==other_leader)
                            dataframe.top_leader(i_follow) = supreme_leader;
                        end
                        dataframe.rel_pos(other_leader, :) = [x_s-x_o, y_s-y_o,
z_s-z_o];
                        % 3. Update chunk size
                        dataframe.num_followers(supreme_leader) =
dataframe.num_followers(supreme_leader) + dataframe.num_followers(other_leader);
                    end
                end
            end
        end
    end
    % calculate number of clusters and determine if convergence is reached
    num_chunks = numel(dataframe.num_followers(dataframe.follower==false));
    conv_max = 100*max(dataframe.num_followers(dataframe.follower==false)) /
num_monomers;
    if mod(it, 1) == 0
        if print_rho
            disp("Completed: " + it + " Iterations " + " out of " + nsteps + ".
Porosity = " + current_rho + ". Number of chunks = " + num_chunks);
        else
            disp("Completed: " + it + " Iterations " + " out of " + nsteps + ".
Convergence = " + conv_max + "%" + ". Number of chunks = " + num_chunks);
        end
    end
    if  num_chunks == 1 %conv_max >= 99.9
        disp("Simulation has converged at iteration " + num2str(it) + " of " +
num2str(nsteps));
        disp("- Number of chunks = " + num_chunks);
        disp("- Convergence = " + conv_max+ "%");
        break
    end
end
```

Generating the geometry mesh as a binary 3D matrix:

```
disp("Generating the 3D mesh with a scaling factor of " + scale_factor + "
...");
coor_data = scale_factor * [dataframe.coordinates, dataframe.radius];
radius = 1.5*coor_data(:, 4);
posX = coor_data(:, 1)+max(radius); posY = coor_data(:, 2)+max(radius); posZ =
coor_data(:, 3)+max(radius);
[x, y, z] = meshgrid(1:mesh_size+2*max(radius));
M = zeros(size(x));
for ip_rho = 1:num_monomers
    if mod(ip_rho,0.1*num_monomers) == 0
        disp("Completed " + 100*ip_rho/num_monomers + "%");
    end
    R = sqrt((x-round(posX(ip_rho))).^2 + (y-round(posY(ip_rho))).^2 + (z-
round(posZ(ip_rho))).^2);
    M(R <= radius(ip_rho)) = 1;
end
actual_rho = nnz(~M)/numel(M);
disp("Final Porosity = " + actual_rho);
disp('Storing bitmap slices... ');
if exist('NEW!', 'dir')
    rmdir('NEW!', 's')
end
mkdir NEW!
for j = 1:100
    fileName = fullfile('NEW!', strcat('mono', num2str(j),'.bmp'));
    imwrite(squeeze(~M(j, :, :)), fileName);
end
% simulation ends, geometry slices are stored in a 'NEW!' folder
```

Merging of the pixel-based and sphere-based geometries:

```
%% Load sphere-based system
size_spheres_500 = [500 500 500]; % x y z; x is the direction of flow
geo_spheres = zeros(size_spheres_500);
for i = 1:size_spheres_500(1)
    imagee = imread(['slices_spheres_500/mono', num2str(i, '%03.f'), '.bmp']);
    geo_spheres(i, :, :) = imagee;
end
geo_spheres(geo_spheres==255) = 1;
rho_spheres = 1 - (numel(geo_spheres(geo_spheres==0))/numel(geo_spheres));
%% Load pixel-based system - trimmed to 200x150x150
size_pixels_100 = [100 100 100]; % x y z; x is the direction of flow
geo_pixels = zeros(size_pixels_100);
for i = 1:size_pixels_100(1)
    imagee = imread(['slices_pixels_100/mono', num2str(i, '%05.f'), '.bmp']);
    geo_pixels(i, :, :) = imagee;
end
geo_pixels(geo_pixels==255) = 1;
rho_pixels = 1 - (numel(geo_pixels(geo_pixels==0))/numel(geo_pixels));
%% Extend pixelated geometry to match the size of the sphere-based one
geo_pixels_extended = geo_pixels;
for i = 1:3
   geo_pixels_extended = cat(i, geo_pixels_extended,geo_pixels_extended,
geo_pixels_extended, geo_pixels_extended, geo_pixels_extended);
end
%% Mask the sphere-based geometry with the pixel-based one
geo_combined=geo_spheres;
geo_combined(geo_combined==0) = geo_pixels_extended(geo_combined==0);
subsection_size = 200;
geo_combined_subsection = geo_combined(1:subsection_size, 1:subsection_size,
```

C.3 Added Function to the Palabos Permeability Code:

```
void computeVelocities(MultiBlockLattice3D<T,DESCRIPTOR>& lattice, plint iter)
{
    const plint nx = lattice.getNx();
    const plint ny = lattice.getNy();
    const plint nz = lattice.getNz();
    for (plint i=0;i<nx; ++i) {
        std::string zStr = "zVF" + std::to_string(i) + ".dat";
        std::string xStr = "xVF" + std::to_string(i) + ".dat";
        std::string yStr = "yVF" + std::to_string(i) + ".dat";
        // Compute the x-direction of the velocity (direction of the flow).
        plb_ofstream XComp(xStr.c_str());
        XComp << std::setprecision(10) << *computeVelocityComponent(lattice,
Box3D(i, i, 0, ny-1, 0, nz-1), 0) << std::endl;
        // Compute the y-direction of the velocity.
        plb_ofstream YComp(yStr.c_str());
        YComp << std::setprecision(10) << *computeVelocityComponent(lattice,
Box3D(i, i, 0, ny-1, 0, nz-1), 1) << std::endl;
        // Compute the z-direction of the velocity.
        plb_ofstream ZComp(zStr.c_str());
        ZComp << std::setprecision(10) << *computeVelocityComponent(lattice,
Box3D(i, i, 0, ny-1, 0, nz-1), 2) << std::endl;
        // Compute norm at half way
        // plb_ofstream half("half.dat");
        // half << std::setprecision(10) << *computeVelocityNorm(lattice,
Box3D(nx/2, nx/2, 0, ny-1, 0, nz-1)) << std::endl;

    }
}
```

C.4 Random Walk Particle Tracking:

Initialization and loading of geometry and velocity field:

```
disp('Initializing..');
load velFields.mat;
lattice_size = [200 150 150]; % x y z; x is the direction of flow
geo3D = zeros(lattice_size);
for i = 1:lattice_size(1)
    imagee = imread(['slices/mono', num2str(i, '%05.f'), '.bmp']);
    geo3D(i, :, :) = imagee;
end
geo3D(geo3D==255) = 1;
rho = 1 - (numel(geo3D(geo3D==0))/numel(geo3D));
```

Definitions and declarations:

```
ut = 30; % preset time multiplier
np = 50; np = np^2; % number of particles
total_jump = 0.5;
exitPoint = 400; % column length
k = 0; % capacity factor, between 0 and 5
store_trajectories = false;
store_dispersion_evolution = false;
k_sim = 3.7697;
Dm_p = 1.0e-10; % diffusion coefficient, m2/s
R =  1/(1+k); % Retention factor
X_vel = uf * XvelField;
Y_vel = uf * YvelField;
Z_vel = uf * ZvelField;
umax_base = max(max(max(XvelField)));
umax = max(max(max(X_vel)));
uavg_scaled_real = mean(mean(mean(X_vel(X_vel~=0)))); % intrinsic average
velocity, pxl/ts
uavg_base_superficial = mean(mean(mean(XvelField))); % superficial base average
velocity, pxl/ts
uavg_scaled_superficial = mean(mean(mean(X_vel))); % superficial base average
velocity, pxl/ts
C_v = 1e-6 / (1/6); % kinematic viscosity conversion factor, m2/s
C_D = C_v; % Diffusion & Dispersion, m2/s
Dm = Dm_p / C_D;
fun = @(dt) (dt * max(X_vel(:))+sqrt(2*Dm*dt))-total_jump;
dt = fsolve(fun, 1);
dt_p = 1.0e-08;
C_t = dt_p;
C_rho = 1000; % Density conversion factor, kg/m3
C_L = sqrt(C_t * C_v); % simulation resolution, m
C_u = C_v/C_L; % velocity
C_P = C_v^2 * C_rho / C_L^2; % Pressure, Pa
L = lattice_size(1);
L_p = L * C_L; % physical length
dp = (1/6) * uavg_scaled_superficial * L / k_sim;
dp_base = (1/6) * uavg_base_superficial * L / k_sim;
dp_p = dp*C_P;
dp_grad = dp_p/L_p;
de = sqrt(200 * (1 - rho)^2 * k_sim / rho^3);
de_p = de * C_L;
uavg_p = uavg_scaled_superficial * C_u; % m/s
umax_p = umax * C_u; % m/s
k_darcy = (1/6) * uavg_base_superficial * 200 / dp_base;
k_darcy_p = k_darcy * C_L^2;
Pe = de * uavg_scaled_real / Dm;
```

```
nsteps = round(ut*exitPoint/(uavg_scaled_real*dt));
l = sqrt(2 * Dm * dt); % Legnth of a diffusive step, pxl
Courant = (umax * dt + l) / 0.5;
if Courant <= 1.0001
    disp('Courant critarion VERIFIED');
    disp(['Courant = ', num2str(Courant)]);
    disp(['Pe = ', num2str(Pe)]);
    disp(['Re = ', num2str(Re)]);
    disp(['k = ', num2str(k_darcy_p)]);
    disp(['dp/m = ', num2str(dp_grad/1e6)]);
    disp(['de_p = ', num2str(de_p*1e6)]);
else
    disp(['Courant = ', num2str(Courant)]);
    disp('Courant critarion NOT MET');
end
```

Main loop:

```
[row, col] = find(squeeze(geo3D(1, :, :))); % indices of nonzero elements
i_index = [row, col];
rand_array = [-1, 1];
passTime = 0.1 * ones(1, np);
initial_dist = zeros(np, 2);
if store_trajectories
    posMAT = zeros(np, nsteps, 3);
end
if store_dispersion_evolution
    dis_evol = zeros(np, nsteps);
end
tic;
disp('Simulation is running.')
parfor ip = 1:np
    random_row = randi(numel(i_index(:,1)));
    posX = 1;
    posX_global = 1;
    posY = i_index(random_row, 1);
    posZ = i_index(random_row, 2);
    initial_dist(ip, :) = [posY posZ];
    time_count = 0;
    for it = 1:nsteps
        if posX < 1
            posX = 1;
        end
        if posX_global < 1
            posX_global = 1;
        end
        if posX > lattice_size(1)
            posX = 1;
        end
        % Boundaries: lattice_size - Y component
        if posY < 1
            posY = 1;
        end
        if posY > lattice_size(2)
            posY = lattice_size(2);
        end
        % Boundaries: lattice_size - Z component
        if posZ < 1
            posZ = 1;
        end
        if posZ > lattice_size(3)
            posZ = lattice_size(3);
        end
        % displacement according to current phase
        if geo3D(round(posX), round(posY), round(posZ)) == 0
            if binornd(1, 1-R)
```

```
            end

            % Processing exit
            if store_trajectories
                posMAT(ip, it, :) = [posX_global posY posZ];
            end
            if store_dispersion_evolution
                dis_evol(ip, it) = posX_global;
            end
            % Updating position
            posX_global = posX_global_old;
            posX = posX_old;
            posY = posY_old;
            posZ = posZ_old;
        else
            posX_global_old = posX_global;
            posX_old = posX;
            posY_old = posY;
            posZ_old = posZ;

            % Trilinear interpolation for X component of velocity
            vx1 = X_vel(floor(posX), floor(posY), floor(posZ));
            vx2 = X_vel(ceil(posX), floor(posY), floor(posZ));
            vx3 = X_vel(floor(posX), ceil(posY), floor(posZ));
            vx4 = X_vel(ceil(posX), ceil(posY), floor(posZ));
            vx12 =  vx1 + (vx2-vx1)*(posZ-floor(posZ));
            vx34 =  vx3 + (vx4-vx3)*(posZ-floor(posZ));
            vx1234 = vx12 + (vx34-vx12)*(posY-floor(posY));
            vx5 = X_vel(floor(posX), floor(posY), ceil(posZ));
            vx6 = X_vel(ceil(posX), floor(posY), ceil(posZ));
            vx7 = X_vel(floor(posX), ceil(posY), ceil(posZ));
            vx8 = X_vel(ceil(posX), ceil(posY), ceil(posZ));
            vx56 =  vx5 + (vx6-vx5)*(posZ-floor(posZ));
            vx78 =  vx7 + (vx8-vx7)*(posZ-floor(posZ));
            vx5678 = vx56 + (vx78-vx56)*(posY-floor(posY));
            vx_interpolated = vx1234 + (vx5678-vx1234)*(posX-floor(posX));

            % Trilinear interpolation for Y component of velocity
            vy1 = Y_vel(floor(posX), floor(posY), floor(posZ));
            vy2 = Y_vel(ceil(posX), floor(posY), floor(posZ));
            vy3 = Y_vel(floor(posX), ceil(posY), floor(posZ));
            vy4 = Y_vel(ceil(posX), ceil(posY), floor(posZ));
            vy12 =  vy1 + (vy2-vy1)*(posZ-floor(posZ));
            vy34 =  vy3 + (vy4-vy3)*(posZ-floor(posZ));
            vy1234 = vy12 + (vy34-vy12)*(posY-floor(posY));
            vy5 = Y_vel(floor(posX), floor(posY), ceil(posZ));
            vy6 = Y_vel(ceil(posX), floor(posY), ceil(posZ));
            vy7 = Y_vel(floor(posX), ceil(posY), ceil(posZ));
            vy8 = Y_vel(ceil(posX), ceil(posY), ceil(posZ));
            vy56 =  vy5 + (vy6-vy5)*(posZ-floor(posZ));
            vy78 =  vy7 + (vy8-vy7)*(posZ-floor(posZ));
            vy5678 = vy56 + (vy78-vy56)*(posY-floor(posY));
            vy_interpolated = vy1234 + (vy5678-vy1234)*(posX-floor(posX));

            % Trilinear interpolation for Z component of velocity
            vz1 = Z_vel(floor(posX), floor(posY), floor(posZ));
            vz2 = Z_vel(ceil(posX), floor(posY), floor(posZ));
            vz3 = Z_vel(floor(posX), ceil(posY), floor(posZ));
            vz4 = Z_vel(ceil(posX), ceil(posY), floor(posZ));
            vz12 =  vz1 + (vz2-vz1)*(posZ-floor(posZ));
            vz34 =  vz3 + (vz4-vz3)*(posZ-floor(posZ));
            vz1234 = vz12 + (vz34-vz12)*(posY-floor(posY));
            vz5 = Z_vel(floor(posX), floor(posY), ceil(posZ));
            vz6 = Z_vel(ceil(posX), floor(posY), ceil(posZ));
            vz7 = Z_vel(floor(posX), ceil(posY), ceil(posZ));
            vz8 = Z_vel(ceil(posX), ceil(posY), ceil(posZ));
            vz56 =  vz5 + (vz6-vz5)*(posZ-floor(posZ));
            vz78 =  vz7 + (vz8-vz7)*(posZ-floor(posZ));
            vz5678 = vz56 + (vz78-vz56)*(posY-floor(posY));
```

```
            % Displacement of tracers (diffusion & drift)
            x_displacement = vx_interpolated * dt  ...
                + l * rand_array(randi(numel([-1,1])));
            y_displacement = vy_interpolated * dt ...
                + l * rand_array(randi(numel([-1,1])));
            z_displacement = vz_interpolated * dt ...
                + l * rand_array(randi(numel([-1,1])));
            posX_global = posX_global + x_displacement;
            posX = posX + x_displacement;
            posY = posY + y_displacement;
            posZ = posZ + z_displacement;
            if posX_global > exitPoint && passTime(ip) == 0.1
                passTime(ip) = time_count;
                if ~store_trajectories && ~store_dispersion_evolution
                    break;
                end
            end
            if store_dispersion_evolution
                dis_evol(ip, it) = posX_global;
            end
            if store_trajectories
                posMAT(ip, it, :) = [posX_global posY posZ];
            end
        end
        time_count = time_count + dt;
    end
end
toc;
disp('Simulation is finished.');
```

Calculate dispersion:

```
passTime2 = passTime(passTime ~= 0.1);
tau_simu = mean(passTime2);
phi = 1 - (numel(passTime2) / np); % retained particles, a fraction
sigma2 = sum((passTime2 - tau_simu).^2) ./ (numel(passTime2));
Dax_time = exitPoint * sigma2 * uavg_scaled_superficial / tau_simu^2 / 2;
h = 2*Dax_time / (Dm * Pe);
```

Helper Function: Converting LB velocity files to a MATLAB structure:

```
file_name = ["VF_rigid_bb", "VF_rigid_periodic", "VF_perm_bb",
"VF_perm_periodic"];
nX= 500;
nY = 500;
nZ = 500;
sz = [nZ nY];
for j = 1:numel(file_name)
    clear XvelField YvelField ZvelField
    XvelField=zeros(500,500,500);
    YvelField=zeros(500,500,500);
    ZvelField=zeros(500,500,500);
    for i = 1:nX
        disp(['System ',num2str(j),'.', ' Slice: ', num2str(i), ' out of 500.'])
        XvelField(i, :, :) = reshape(load(fullfile(file_name(j), ['xVF',
num2str(i-1), '.dat'])), sz);
        YvelField(i, :, :) = reshape(load(fullfile(file_name(j), ['yVF',
num2str(i-1), '.dat'])), sz);
        ZvelField(i, :, :) = reshape(load(fullfile(file_name(j), ['zVF',
num2str(i-1), '.dat'])), sz);
    end
    save(file_name(j));
end
```

Performing the RWPT simulation with multiple Peclet numbers:

```
%% System specifications
system_type = 'perm'; % takes 'rigid' for rigid spheres or 'perm' for permeable
spheres
vf_bc = 'bb'; % takes 'bb' or 'periodic' depending on the BC of the flow
simulation
k_sim = 86.3075; % system permeability from the flow simulation.
dp_used = 0.0001; % applied pressure difference in the flow simulation.
lattice_size = 500 * ones(1,3); % [500 500 500]; % x y z; x is the direction of
flow
de_original = 31.6932; % Origianl sphere diamter from the geometry construction
simulation
verbosity=0; % whether or not the particle number should be displayed during the
simulation. Turned off for multiple runs.
%% Velocity scale range
uf_range = [0.0001, 5];
uf_space = logspace(log10(uf_range(1)), log10(uf_range(2)), 30);
%% Loading geometry.
% The geometry folder containg .bmp images must be in the working directory
disp('Loading geometry.');
geo3D = zeros(lattice_size);
for i = 1:lattice_size(1)
    imagee = imread(['slices_', system_type, '/mono', num2str(i, '%03.f'),
'.bmp']);
    geo3D(i, :, :) = imagee;
end
geo3D(geo3D==255) = 1;
rho = 1 - (numel(geo3D(geo3D==0))/numel(geo3D));
disp(['Geometry loaded. Total porosity = ', num2str(rho)]);
%% Loading velocity field
% The velocity field must be in the working directory as a .mat file.
disp('Loading velocity field.');
load(['VF_', system_type, '_', vf_bc,'.mat']);
disp(['Velocity field loaded. Average velocity = ',
num2str(mean(XvelField(:)))]);
np = 70; np = np^2;
%% Run multiple simulations
counter = 0;
for uf = uf_space
    disp(['------------------- uf = ', num2str(uf), ' -------------------']);
    PMcodes_v4_spheres;
    counter = counter + 1;
    saveName = [system_type,'_', vf_bc, '_',num2str(counter), '.mat'];
    save(saveName, 'uf', 'Dax_time', 'Pe', 'Re', 'Dm', 'h', 'nsteps',
'k_darcy_p', 'phi', 'uavg_m', 'np', 'de', 'exitPoint', 'umax', 'k_sim', 'uavg',
'C_t', 'C_u', 'C_D', 'C_L', 'C_P', 'dp_grad', 'dp', 'dt', 'passTime2',
'passTime',...
        'system_type', 'vf_bc', 'uavg_p', 'umax_p', 'de_p');
    clearvars -except uf XvelField YvelField ZvelField geo3D rho lattice_size np
counter system_type vf_bc k_sim dp_used de_original verbosity
    pause(0.5);
```

D. Simulation Results on the Pixel-based Geometry

The pixel-based geometry which features a highly uniform porous media was the first type of geometry constructed in the early stages of this study. Later, when the sphere-based geometry construction algorithm was developed, the focus of the main results and discussion (Chapter 4) has been on the sphere-based system and the role of the pixel-based one was reduced to model intraparticle porous space and to generate a modified geometry with permeable spheres. Here the previous simulation results for the pixel-based geometry are given.

D.1. Generated Geometry

Figure D.1 shows the constructed geometry using the legacy (pixel-based) approach with a 0.9 porosity. Figure D.2 shows the middle cross-section of the system at the initialization step and at the final step, to illustrate the formation of clusters.

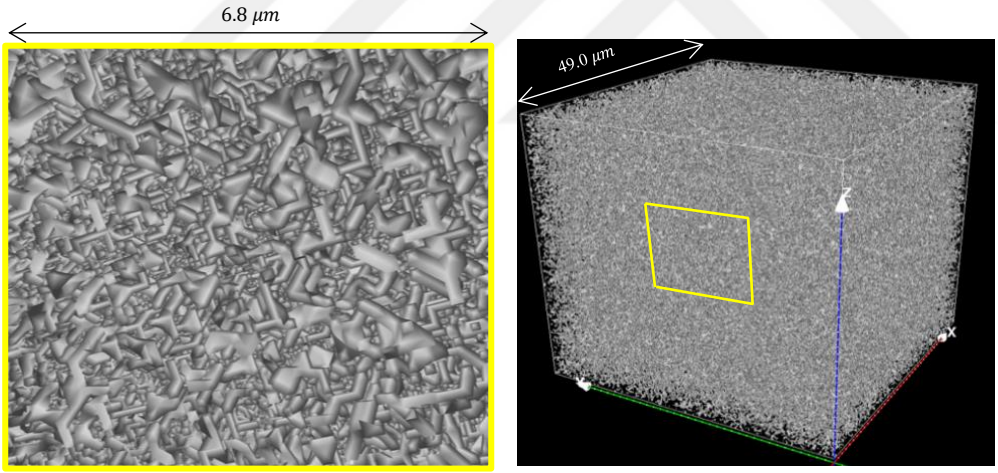


Figure D.1. The constructed porous medium using the pixel-based algorithm. The size of the system is $49^3 \mu m^3$ (200^3 voxels) and a porosity of 0.9. A zoomed-in region on the left is enclosed by the yellow frame on the right.

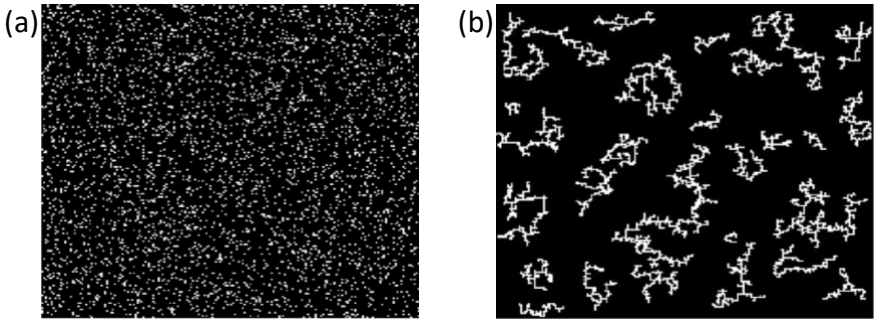


Figure D.2. A two-dimensional illustration of the pixel-based geometry construction method where a random configuration of solid sites (a) form connected clusters (b).

D.2 Flow Simulation

Figure D.3 shows a 3D rendering of the resulting LBM velocity field with bounce-back boundary conditions on the pixel-based geometry. The maximum velocity was reached at $24\ \mathrm{m/s}$. As expected, due the uniformity of the structure, the large velocities are more uniformly distributed throughout the system, than with the sphere-based system which are given in Figure 4.8. This is also evident from the axial velocity distribution shown in Figure D.4 (a), compared to the velocity distribution of the sphere-based system shown in Figure 4.7.

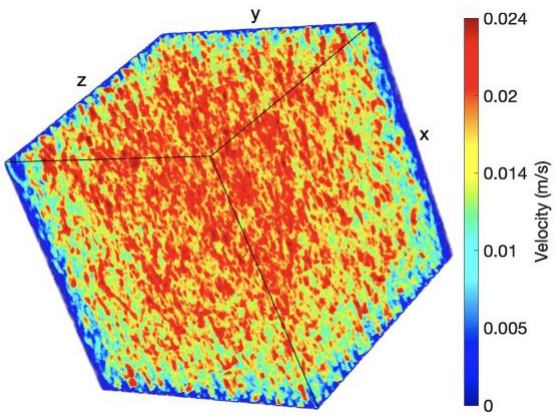


Figure D.3. A 3D colormap velocity field resulting from the LBM simulation on the pixel-based geometry shown in Figure D.1. The size is $200 \times 200 \times 200$ voxels, or $49 \times 49 \times 49\ \mu\mathrm{m}^3$.

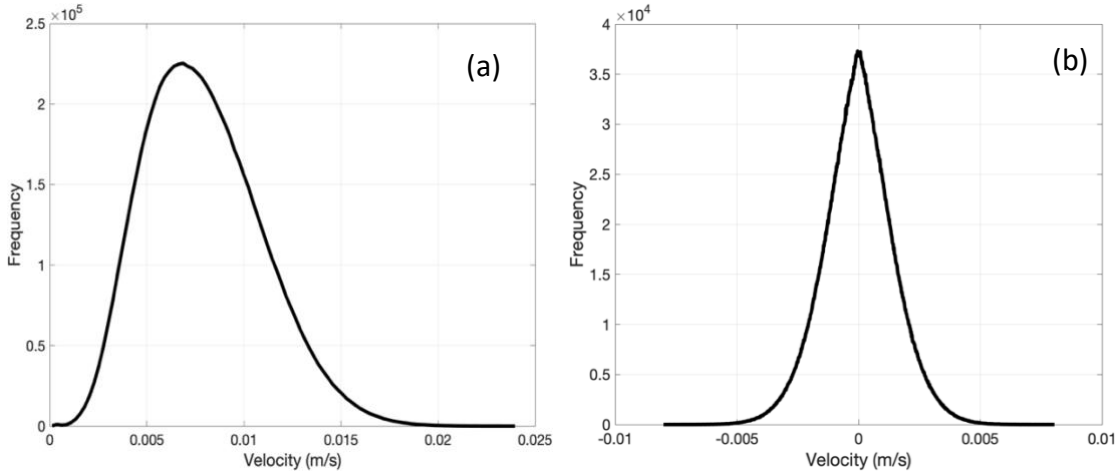


Figure D.4. Axial (a) and lateral (b) velocity distributions in the pixel-based system.

D.3 Dispersion Simulation

Figure D.5 shows the random-walk simulation results on the pixel-based system, compared to similar literature works. It is observed from the plot that small dispersion values were achieved when compared to the results from similar studies in the literature, which is expected due the uniformity of the pixel-based geometry.

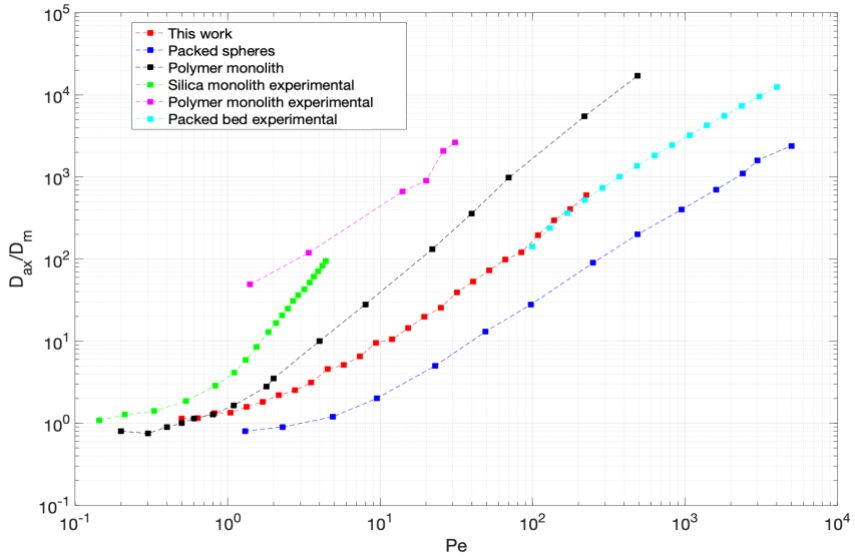


Figure D.5. Axial dispersion coefficient (normalized by molecular diffusion) vs. Péclet number resulting from the random-walk simulation on the pixel-based system.

To illustrate this further, and by referring to the discussion on asymptotic behavior of dispersion in Section 2.1.1 and Section 4.3.4, it is expected for asymptotic dispersion to be reached if the structure features less nonuniformities, which is the case for the pixel-based geometry, or as shown in Figure D.6.

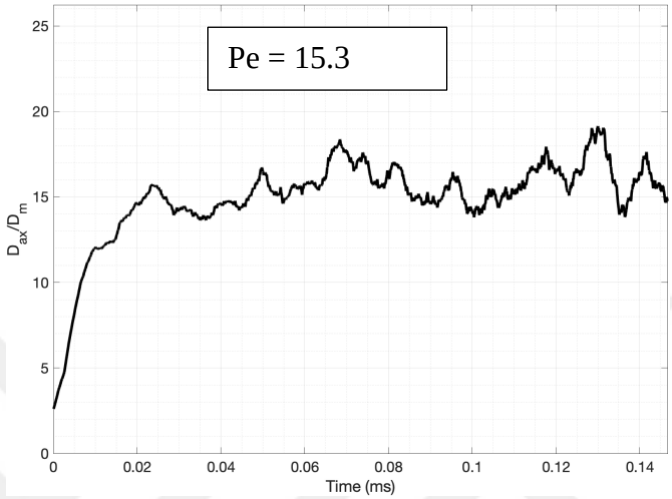


Figure D.6. Asymptotic dispersion, on the pixel-based geometry, was easily achieved due to high level of structural uniformity.

D.4 Simulation of Retention

The retention phenomenon was modeled in this work on the pixel-based system as a preliminary proof of concept. Two methods were followed. The first method was in-line within the random walk implementation by simply introducing a time delay of one time step to the particle if just collided onto the stationary phase with a given probability defined by a retention factor R which takes a value between 0 and 1. If R is set to 1, the probability that the colliding particle will receive a time delay in its trajectory profile is zero, meaning no retention in the simulation. If R is zero, then any collision leads to adsorption onto the stationary phase for a time delay of one time step. The result of three simulation runs with different R values is shown in Figure D.7 (top panel). The second approach was to scale down the axial velocity field (the drift term in the random walk model) to account for a total time delay on all of the tracer band, with the scale factor related to the retention factor. This is

shown in Figure D.7 (bottom panel). Both approaches yielded effectively the same results.

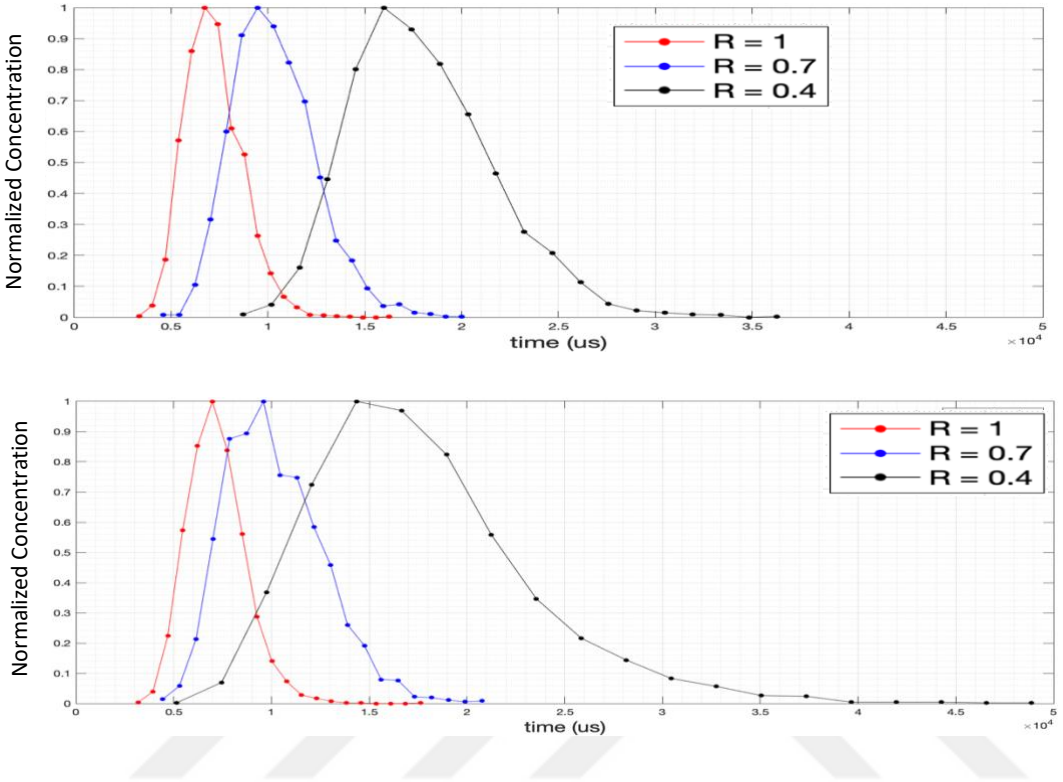


Figure D.7. Modeling of retention by a simple probability-based time delay approach using in-line implementation (top) and a scaled velocity field (bottom) at different values for the retention factor.