Pore Network Modeling of Hydrodynamic Dispersion and Irreversible Adsorption of Nanoparticles in Partially Saturated Granular Media

by

Stephen Dauphinais

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Chemical Engineering (Water)

Waterloo, Ontario, Canada, 2022

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The transport of nanoparticles in porous media, whether deliberate or accidental, is of critical importance in several fields including migration of contaminants, wastewater clean-up, and enhanced oil recovery. Pore scale models that determine the fate of nanoparticles can be advantageous compared to continuum models as they are able to account for various phenomena that occur at the pore scale and are controlled by the interaction between nanoparticles and solid-fluid or fluid-fluid interfaces. These phenomena include hindered diffusion, size exclusion, adsorption, deposition and erosion, to name a few. Here a Eulerian pore scale model is built using OpenPNM, and simulations are performed on a pore network based on a column packed with spherical beads. This pore network is extracted from a voxel image using PMEAL's SNOW algorithm, and it is found to accurately represent key properties of a real packed column, namely the permeability, tortuosity, and capillary breakthrough pressure. The resulting pore network model is used to investigate two aspects of nanoparticle transport that have so far received limited attention. One aspect concerns the effects of hindered diffusion and velocity profile exclusion on nanoparticle dispersion in saturated porous media. The other aspect concerns the irreversible adsorption of nanoparticles on trapped immobile non-wetting phase ganglia present in partially-saturated porous media.

The longitudinal dispersion coefficient is determined by simulating transient advection and diffusion in the pore network, introducing either a pulse or step-change injection, and then fitting analytical solutions to the resulting elution curve. Both pulse and step-change injections fit well with the corresponding analytical solution, but step change injections are more accurate. Pore network simulations aimed at reproducing the experiments of Chrysikopoulos and Katzourakis (2015) show that hindered diffusion and velocity profile exclusion do not explain the experimental observations, as hypothesized by the authors. It is found that nanoparticle size influences the dispersion coefficient or the effective particle velocity only when the ratio of particle to bead (solid grain) size is sufficiently high (greater than about 0.01). When this ratio is sufficiently high the nanoparticles experience an earlier breakthrough due to the velocity profile exclusion. It was also determined that hindered diffusion plays a significant role only when the Peclet number is less than 10.

Irreversible adsorption of nanoparticles onto aqueous-non aqueous phase interfaces is also simulated. The non-wetting phase interface is created by running an invasion-percolation algorithm on the pore network. The adsorption constant is taken to be a function of the maximum energy barrier to adsorption and a blocking function, and the total fluid-fluid interface area is determined using the pores occupied by

the non-wetting phase but connected to the water phase. This method is found to provide a reasonable prediction for what the surface area of the non-wetting phase is by comparing it to real-world simulations. It is shown that information such as whether blocking is occurring can be inferred based on the shape of the breakthrough curve under certain conditions. Further, it is shown that smaller particles increase the rate of adsorption. When the Damkholer number is much greater than 10 the rate of adsorption is transport limited, and when the Damkholer number is much less than 10 the simulation is limited by the energy barrier to adsorption.

Acknowledgements

I would like to thank my supervisor Dr. Marios Ioannidis for his continuous support and mentorship. This work would not have been possible without the guidance he has given me. I would also like to extend my sincere gratitude to Dr. Jeff Gostick for his continuous advice and feedback. This work also would not have been possible without his support. I am also very appreciative of Dr. Nasser Mohieddin Abukhdeir for serving as a member of my examining committee.

I would also like to thank the Natural Sciences and Engineering Research Council for finical assistance, as well as the University of Waterloo's Water Institute for both additional funding and guidance.

Finally, I would like to extend special thanks to my parents John and Kathy, as well as my brother Paul for supporting me throughout this path.

Table of Contents

List of Figures

List of Tables

Nomenclature

a_o	Water-NWP interfacial area per unit volume
A_{ads}	Surface area available for adsorption
A	Cross-sectional area
A_c	Cross-sectional area of column
A_s	Surface area of a pore $= \pi d_{pore}^2$
$A_{NWP,surf}$	Total surface area of NWP surface pores
$\bar{B}(\theta)$	Blocking function Eqn. (2.17
C	Concentration
C_e	Eluted Concentration
C_i, C_j	Concentration for pore *i/j*
C_0	Bulk concentration
Da_d	Diffusive Damkholer number Eqn. (5.5
Da_h	Hydrodynamic Damkholer number Eqn. (5.4
d_b	Diameter of beads
d_c	Diameter of column
$\langle d_L \rangle$	Characteristic throat length
d_{pore}	Diameter of pore
d_p	Diameter of nanoparticle
D_{AB}	Mass diffusivity of component A diffusing through component B
D_{eff}	Effective diffusivity using Eqn. (2.3
D_{ETD}	Effective Taylor dispersion
D_{HD}	Hindered diffusion
D_L	Longitudinal dispersion coefficient
D_t	Transverse dispersion coefficient
F	See Eqns. (3.5 and (3.8
$F_1(\varphi), F_2(\varphi)$	Hindered diffusion correction factors in Eqns. (2.9 and (2.10
g_i^h, g_{ij}^h, g_j^h	Hydraulic conductance for given pore/throat

g_i^d, g_{ij}^d, g_j^d	Diffusive conductance for given pore/throat
G_{ij}^{ad}, G_{ji}^{ad}	Advective diffusive conductance terms for pore-throat-pore conduit *ij*
G_{ij}^h	Hydraulic conductance for pore-throat-pore conduit *ij*
G_{ij}^{d}	Diffusive conductance for pore-throat-pore conduit *ij*
I_p^*	Specific polar moment
$\vec{j}$	Adsorption flux
k	Permeability
k_a	Adsorption constant
k_b	Boltzmann constant ($=1.380649 \times 10^{-23}\ m^2\ kg\ s^{-2}\ K^{-1}$)
k_{KC}	Kozeny-Carman permeability found with Eqn. (4.1
k_s	Simulated permeability
L	Length
L_c	Column length
L_{ctc}	Conduit to conduit length
n	Number of time-steps
n_A	Mass flux of solvent A
n_{ads}	Number of adsorbed particles
n_i	Pore of interest
n_j	Pore connected to pore of interest
n_p	Total number of pores in a network
n_t	Total number of throats in a network
m	Fitting parameter for blocking function ($m = 3$ for spheres)
$\dot{m}_{ij}$	Mass flow rate from pore *i* to *j*
M_{inj}	Total mass injected
p	Pressure
p_c°	Breakthrough capillary pressure
Pe_m	Molecular Peclet Number $= U_{fitted} \times d_b / D_{eff}$
q	Flow rate across entire porous medium
q_{ij}	Flow rate across throat *ij*

r_i^{ads}	Rate of particle adsorption
R_i	Accumulation rate
$\langle R_t \rangle$	Characteristic throat radius
S	Cross-sectional area of particle ($= \pi d_p^2/4$)
S_{NWP}	Non-Wetting Phase Saturation
$S_{NWP,calc}$	Calculated NWP saturation directly from pore network
T	Temperature (T = 298 K for all simulations)
t	time
t_{step}	Length of time step
U	Interstitial velocity
$\overline{U}$	Mean velocity
U_{Darcy}	Darcy (superficial) Velocity, $= U/\varepsilon$
U_{EPV}	Effective Particle Velocity (See Eqn. (2.11)
U_{fitted}	Interstitial velocity determined by fitting analytical dispersion equation
V_B	Bulk Volume
V_i	Volume of individual pore
V_P	Pore (void) volume
V_S	Solid volume
V_w	Water volume
x	distance

Greek

α	Parameter for Eqn. (5.1
β	Parameter for Eqn. (5.1
α_L	Longitudinal dispersivity
γ	Interfacial tension
ε	Porosity
ε_{eff}	Effective porosity ($= V_W/V_B$)
θ	Surface coverage

θ_{total}	Total surface coverage
θ_{max}	Jammed/full coverage (0.91)
λ	Fitting parameter to determine tortuosity
μ_B	Dynamic viscosity of solvent B
ρ_p	Mass density of particles
τ	Tortuosity
ϕ_b	Maximum energy barrier to adsorption
φ	Reduced pore diameter $(= d_p/d_{pore})$
Φ_V	Volume fraction of NPs

Acronyms

HD	Hindered Diffusion
IP	Invasion Percolation
NP	Nanoparticle
NWP	Non-Wetting Phase
PNM	Pore Network Model
PV	Pore Volume
TAD	Transient Advection Diffusion
VPE	Velocity Profile Exclusion

Chapter 1 Introduction

1.1 Background and Motivation

Colloidal particles are typically defined as particles ranging from 1 *nm* to 10 μm [1], a subset of which are considered nanoparticles (NPs) which are typically defined as particles ranging from 1 to 100 *nm*. Transport of NPs in porous media is a phenomenon involved with several scientific and engineering fields. In groundwater aquifers containments may adsorb onto colloidal particles [2, 3], NPs can be used for environmental clean-up [4, 5], and NPs such as viruses and bacteria may be containments themselves [6, 7]. The high surface to volume ratio of NPs also makes them quite useful in the oil and gas industry where they can be used in drilling and completion [8, 9], reservoir characterization [9, 10], and enhanced oil recovery [10]. Transport of NPs in porous media also has several biomedical applications such as delivery of pharmaceuticals [11] and the uses of medical masks preventing the spread of colloidal droplets of water containing viruses [12].

Computationally predicting the transport of NPs can either be done by use of a volume-averaged continuum model, or with a pore-network model (PNM). Continuum models typically are based on resolving partial differential equations, which are generally computationally simpler than pore-scale models; a solution fails to capture pore-scale phenomena. Furthermore, continuum models are typically based on experimentally measured relationships to describe macroscopic properties of the porous medium. On the other hand, PNMs describe the void-space of the porous media as a network of pores and throats [13] whose connectivity and geometry are defined to represent that of the desired porous medium. There are two theoretical approaches used to preform transport of NP in porous media simulations: Lagrangian and Eulerian [14]. The Lagrangian method describes the trajectory of individual particles [15-17], and typically perform Monte-Carlo based simulations [14]. On the other hand, is the Eulerian approach, which is the method implemented in this thesis. This approach describes the evolution of all NPs in the porous medium as a concentration and resolves the transport by resolving 1-dimentional advection-diffusion equations for each pore-throat-pore assembly [13, 14], which is a method that has sparsely been implemented to model NP transport in porous media. In this thesis, a Eulerian model describing the transport of is built using OpenPNM, which is an open-source software that runs on Python. Two phenomena regarding the transport of NPs in porous media are simulated

using the models built in this thesis: determination of the dispersion of NPs in porous media, and irreversible adsorption of NPs onto a fluid-fluid interface.

Prediction and quantification of the transport of both solvents and particles in porous media is largely based on dispersion coefficients. Longitudinal and transverse are the two different types of dispersion, where longitudinal dispersion occurs in the same direction as pressure driven flow, while transverse dispersion occurs perpendicular to flow. Transverse dispersion is useful when studying plume migration [18], but longitudinal dispersion is generally the more dominant dispersion force [19]. Thus, in this work only longitudinal dispersion is considered. It has been observed that nanoparticles experience earlier breakthrough times than that of a passive solute [15, 20-23]. This observation is typically attributed to either exclusion from the low velocity region of the Poiseuille velocity profile [20] or to streamlining due to particles being excluded from throats of smaller diameter [23]. Another phenomenon that NPs experience is a reduction in their effective diffusivity when near pore walls, an effect commonly referred to as "hindered diffusion" [24-26]. Although there has been many studies investigating the relationship between NP size and dispersion, the relationship between the two remains poorly understood. NPs can be captured onto both solid surfaces [27-31] and onto a non-wetting phase such as oil or water [32-36]. This capture is governed by short-range interactions between the particles and the interfaces, which results in either reversible or irreversible adsorption [37]. There are many studies that estimate the capture of NPs either by means of a continuum model [34, 38] or with a Lagrangian PNM simulation, but this thesis will attempt to do so by means of a Eulerian based PNM simulation.

1.2 Objectives and Scope of Thesis

In this study, a 3-dimensional voxel image of a column packed with spherical beads is generated, and a representative pore network is generated by mapping pores and throats onto the void space of the voxel image. The accuracy of the pore network is determined by running simulations to determine the permeability, tortuosity, and breakthrough capillary pressure and comparing these results to known correlations. Transient advection-diffusion problems can be simulated on this network by defining the thermophysical properties of the particles suspended in solvent water and then determining the necessary transport parameters such as hydraulic and diffusive conductance for each pore and throat. The diffusivity of the NPs is determined using the Stokes-Einstein equation, and the diffusivity and

hydraulic conductance is further adjusted to account for the hindered diffusion and velocity profile exclusion effects.

Values for longitudinal dispersion can then be determined by generating an elution curve for the concentration at the outlet of the pore-network for either a step-change or instantaneous pulse boundary condition at the inlet and fitting the analytical solution to the governing advection-diffusion equation for one-dimensional flow to the elution curve. Results from the simulations are compared to results from in laboratory experiments found in [21], and the influence that including the hindered diffusion and velocity profile exclusion effects has on the simulations is analyzed.

Simulating adsorption onto a non-wetting phase is achieved by introducing a non-wetting phase by use of an invasion percolation algorithm. A method of calculating the total fluid-fluid interface area is defined, and results from this method is compared to real-world experiments. The adsorption simulations are achieved by using a transient advection diffusion algorithm, but with a term added to account for the adsorbed particles. The shape of the resulting elution curve is observed in an effort to see if information regarding the adsorption/blocking of the NPs can be inferred. Finally, the impact that the size of NPs, the defined energy barrier to adsorption, and the velocity has on the total adsorption is observed and quantified by use of Damkholer numbers.

1.3 Structure of Thesis

This thesis is organized into 6 chapters and 1 appendix. Chapter 2 reviews the theoretical considerations regarding transport phenomena in porous media and nanoparticle properties, as well as background on pore network modeling and OpenPNM. Chapter 3 presents various models developed in this thesis including extracting and generating pore networks from voxel images, running transient advection-dispersion simulations to determine dispersion, creation of immobile non-wetting phase, and modeling the adsorption of NPs onto the water-non-wetting phase interface. In Chapter 4, the results from simulating dispersion simulations are presented and compared to results from [21], and the influence that including hindered diffusion and velocity-profile exclusion has on the simulations are analyzed. Chapter 5 analyzes properties of immobile non-wetting phases, and results from simulating adsorption onto the water-non-wetting phase is presented discussed. Chapter 6 summarizes the key findings in this thesis and offers recommendations for future studies. Finally, in Appendix A through Appendix D the code used in simulations is presented.

Chapter 2 Background and Theory

A porous medium is a solid material that contains voids commonly referred to as pores. Pores are typically interconnected, allowing the movement of viscous fluids. Porous materials of geologic origin may be unconsolidated and easy to disaggregate (e.g., a sand pack) or consolidated (e.g., sandstone or limestone). They are characterized at the microscopic scale by the shape and size distribution of individual pores and by the degree of pore interconnectedness. At the continuum (macroscopic) scale, a fundamental property of any porous medium is the porosity, ε, which is the ratio of void volume to the bulk volume (the sum of pore and solid volumes)

$$\varepsilon = \frac{V_P}{V_P + V_S} \qquad \textbf{(2.1)}$$

where V_P is the void volume, and V_S is the solid volume. The void space in a porous medium is generally occupied by one or more fluids, such as air, water, and oil. When the entire void space is occupied by the same fluid, the medium is referred to as "saturated", and when there is more than one fluid occupying the void space the medium is considered "unsaturated".

2.1 Transport Phenomena in Porous Media

Viscous forces within interconnected narrow conduits are associated with significant dissipation of mechanical energy, such that a pressure gradient always accompanies flow through a porous medium. The relationship governing pressure driven flow through a porous continuum is Darcy's Law [39]

$$q = U_{Darcy} \times A = \frac{kA}{\mu_B}\frac{\Delta p}{L} \qquad \textbf{(2.2)}$$

where q is the volumetric flow rate of the fluid, A is the cross-sectional area, μ_B is the viscosity of the fluid, Δp is the pressure drop across the porous medium, L is the length, and k is the permeability. Permeability is a fundamental property of porous media that quantifies the ease with which a viscous fluid can flow through the media in question. A distinct feature of porous media is the effect the presence of a "tortuous" path, delimited by the solid-void boundary, imposes on transport. This effect is quantified in a porous medium by the so-called tortuosity, τ, which refers to the reduction in the diffusive molar flux compared to the diffusion coefficient of the solute [40]. The tortuosity of the pore

network is determined by observing steady Fickian diffusion of a solute. To this end, a solute concentration difference is applied between the inlet and outlet of the pore network. The mass flux n_A through the network as a result of the applied concentration gradient is determined and used to compute the effective diffusivity using Fick's 1st law:

$$D_{eff} = \frac{n_A L_c}{A_c \Delta C} \quad \textbf{(2.3)}$$

The measured tortuosity is then determined as follows:

$$\tau = \frac{\varepsilon D_{AB}}{D_{eff}} \quad \textbf{(2.4)}$$

In the presence of pressure-driven flow through porous media, advection is superimposed on diffusion. The "spreading" of the solute resulting from merging and splitting of fluid streamlines as the solvent flows through the porous medium is referred to as "dispersion", which is caused by three mechanisms [41]:

- Advection from a higher-pressure region to a lower pressure region
- Diffusion from a higher concentration section of the porous medium to a lower pressure region
- Mechanical mixing that arises from the tortuous paths in a porous medium, as well as "swirling" that occurs as flow diverges and converges around obstacles

Even when at the macroscale flow is characterized by a single velocity component (one-dimensional flow), dispersion is manifested in two directions relative to the direction of flow in the pore network: along the same path as flow, and perpendicular to the path of flow [19]. Dispersion in the same path as flow is called "longitudinal dispersion", D_L, and perpendicular dispersion is called "transverse dispersion", D_T. In this study only D_L is considered.

The transport of both nanoparticles and solutes can be described using the advection-dispersion equation [42, 43], which for one-dimensional flow is

$$\frac{\partial C}{\partial t} = D_L \frac{\partial^2 C}{\partial x^2} - U \frac{\partial C}{\partial x} \quad \textbf{(2.5)}$$

where the longitudinal dispersion coefficient D_L is the sum of mechanical dispersion and the effective molecular diffusion. The mechanical dispersion is generally expressed as a power of the interstitial velocity, with U raised to an empirical constant n, and multiplied by the longitudinal dispersivity α_L.

$$D_L = \alpha_L U^n + \frac{D_{AB}}{\tau} \quad \textbf{(2.6)}$$

For a continuum model, the effective molecular diffusion can be found by dividing the molecular diffusion coefficient D_{AB} by the tortuosity τ. According to the hydrodynamic theory, the diffusion coefficient of a spherical particle with diameter d_P, in the absence of fluid slip at the surface of the particle, is given by the Stokes-Einstein equation [44]

$$D_{AB} = \frac{k_b T}{3\pi\mu_B d_P} \quad \textbf{(2.7)}$$

where k_b is Boltzmann's constant, T is the absolute temperature, and μ_B is the dynamic viscosity of the interstitial fluid.

2.2 Nanoparticles in Porous Media

This study aims to investigate the differences nanoparticles (NPs) experience compared to soluble (molecular) tracers in transport problems in porous media. Not only are NPs characterized by much smaller diffusion coefficients, but they are subject to two additional effects. The first of these effects is the so-called hindered diffusion (HD), where the existence of pore walls causes additional reduction of the effective diffusion coefficient for particles with sizes comparable to the pore size. The second effect is a velocity profile exclusion (VPE) effect, where the full Poiseuille velocity profile of a fluid flowing in a pore is not accessible to particle due to its finite size. This results in an increased effective pore velocity and a reduced effective Taylor dispersion coefficient.

2.2.1 Hindered Diffusion

For particles in solvent-filled pores, the walls of the pores "hinder" the diffusion of the particles [26]. As the diameter of the particle approaches the diameter of the of the pore the diffusive transport of the particle is reduced. A common model to calculate the "hindered" diffusion coefficient takes the form of:

$$D_{HD} = D_{AB} F_1(\varphi) F_2(\varphi) \quad \textbf{(2.8)}$$

where D_{AB} is the diffusion constant calculated using the Stokes-Einstein equation, and $F_1(\varphi)$ and $F_2(\varphi)$ are correction factors accounting for different effects arising from the interactions of particles with the pore walls. Both correction factors are theoretically bounded by zero and unity and are both functions of the reduced pore diameter $\varphi = d_p/d_{pore}$. The first correction factor is the so-called steric partition coefficient, which is simply the ratio between the flux area available to the particle to the total flux area

$$F_1(\varphi) = \frac{Flux\ area\ available\ to\ solute}{total\ flux\ area} = \frac{\pi\left(d_{pore} - d_p\right)^2}{\pi {d_{pore}}^2} = (1-\varphi)^2 \quad \textbf{(2.9)}$$

The second correction factor $F_2(\varphi)$ is the hydrodynamic hindrance factor, which represents the hindered Brownian motion of the particle in the solvent filled pore. There are several different models to predict this correction factor. The most used equation predicting $F_2(\varphi)$ developed by [25] is the equation that is used in this work.

$$F_2(\varphi) = 1 - 2.104\varphi + 2.09\varphi^3 - 0.95\varphi^5 \quad \textbf{(2.10)}$$

2.2.2 Velocity Profile Exclusion Effect

One key difference between transport of particles and solvents is that solvents will experience the full velocity profile, while the velocity profile observed by particles is limited by their diameter, as seen in Figure 1. Particles have been shown to have a lower dispersion coefficient value and a higher velocity value that of a solute [22, 23, 45]. In [20], a mathematical model is developed to express the effective particle velocity, U_{EPV} and effective Taylor dispersion coefficient, D_{ETD} for spherical particles flowing in a water-saturated water fracture or cylindrical tube. The effective particle velocity, U_{EPV}, for a cylindrical tube is given by [20]

$$U_{EPV} = \overline{U}\left[1 + \frac{d_P}{R} - \frac{1}{4}\left(\frac{d_P}{R}\right)^2\right] \quad \textbf{(2.11)}$$

where $\overline{U}$ is the mean velocity given as $\overline{U} = \frac{U_{max}}{2}$, and R is the radius of the tube. Note that U_{EPV} is bounded by $\overline{U}$ as d_P approaches 0 and by U_{max}. It can further be shown that the effective Taylor dispersion coefficient for a particle is given by

$$D_{ETD} = D + \frac{1}{192}\frac{U_{max}^2 R^2}{D}\left(1 - \frac{d_P}{R}\right)^6 \quad \textbf{(2.12)}$$

where D is the Stokes-Einstein diffusion coefficient (optionally) corrected for hindered diffusion. As the diameter of the particle approaches zero, the expression for the equation for dispersion approaches the Taylor dispersion coefficient [46]

$$D_{TD} = D + \frac{1}{192}\frac{U_{max}^2 R^2}{D} \quad \textbf{(2.13)}$$

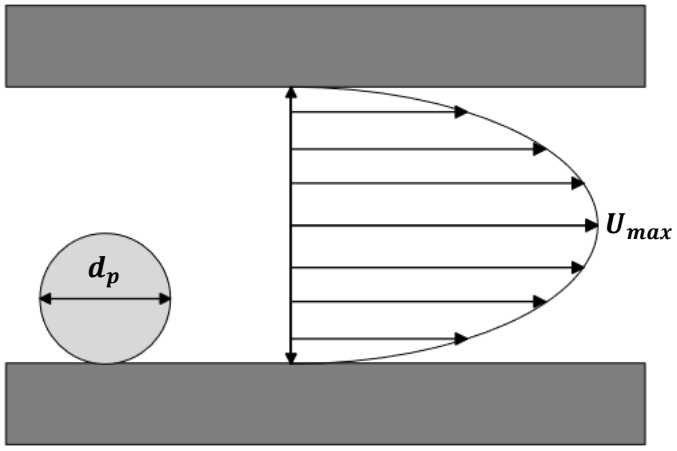


Figure 1: Schematic of a particle travelling in a cylindrical pore with the Poiseuille velocity profile

2.2.3 Irreversible Adsorption of Nanoparticles and Surface Blocking

In unsaturated porous media, NPs can adsorb onto both the fluid/solid interface and onto the fluid/fluid interface. When the adsorption energy is small, thermal fluctuations can expel particles back into the bulk fluid. In this case, the adsorption is known as reversible adsorption. When there is an

energy barrier much greater than the energy associated with thermal fluctuations, particles are practically indefinitely held onto the interface. This is known as irreversible adsorption. In this thesis, only irreversible adsorption is modeled.

Modeling the attachment of colloidal particles onto an interface is based on the generalized random sequential adsorption theory [37]. Initially, when the coverage of the interface with particles is very low, particle attachment is independent of the extent of coverage and the rate of adsorption is directly proportional to the particle concentration in the bulk – a so-called "linear adsorption regime". As adsorption progresses, however, a "blocking effect" emerges whereby particles adsorbed onto an interface limit the adsorption of additional particles. The development of a model that accounts for the "blocking effect" is presented in this section. This model assumes the following:

- Particles are adsorbed randomly onto the available sites, with equal probability for each section of the site
- Particles are adsorbed permanently
- The adsorption can continue until full coverage of the interface is achieved

The adsorption flux of NPs onto an interface, $\vec{j}$, under conditions of constant temperature and pressure can be described by the particle conservation statement [37]:

$$\frac{\partial C}{\partial t} + \nabla \cdot \vec{j} = 0 \qquad \textbf{(2.14)}$$

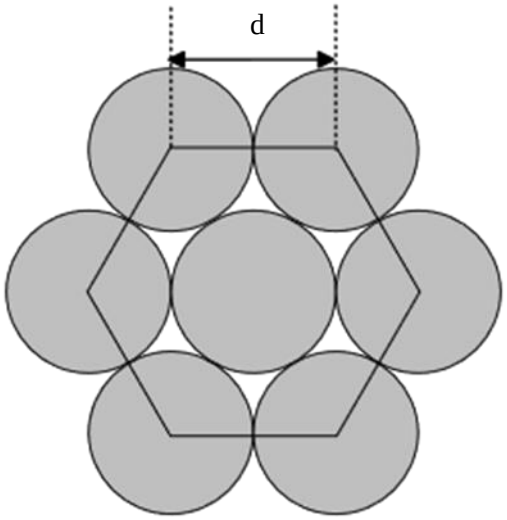
The particle flux is related to the fractional coverage of the interface as follows [37, 38]

$$-\vec{j} = \frac{1}{S}\frac{d\theta}{dt} = k_a C \bar{B}(\theta) \qquad \textbf{(2.15)}$$

where the cross-section of the NP is $S = \pi d_p^2/4$, θ is the fractional surface coverage, k_a is the adsorption constant, C is the concentration of NPs in the bulk, and $\bar{B}(\theta)$ is the so-called blocking function. The fractional surface coverage θ is calculated as

$$\theta = \frac{S n_{ads}}{A_{ads}} \qquad \textbf{(2.16)}$$

where n_{ads} is the total number of adsorbed particles on the interface and A_{ads} is the surface area of the interface available for adsorption. The value of θ is bounded by zero for a pristine surface and θ_{max} for a fully jammed surface. In this paper, the fully jammed surface is taken to be covered to a hexagonal close-packed lattice with no spacing between particles, which is shown to be $\theta_{max} = 0.91$ in Figure 2



$$\theta_{max} = \frac{\text{maximum occupied area}}{\text{total area}} = \frac{3\pi\left(\frac{d}{2}\right)^2}{\frac{3\sqrt{3}}{2}d^2} \cong 0.91$$

Figure 2: Geometrical pattern for close packed lattice showing $\theta_{max} = 0.91$

The blocking function $\bar{B}(\theta)$ is approximately given by [37]

$$\bar{B}(\theta) = 2.32\left(1 - \frac{\theta}{\theta_{max}}\right)^m \qquad \textbf{(2.17)}$$

where m is a fitting parameter that is equal to 3 for spheres. As θ approaches θ_{max}, the blocking function $\bar{B}(\theta)$ approaches 0, thus shutting down any additional adsorption as the interface becomes jammed.

The adsorption constant k_a for a parabolic interaction energy profile is approximately given by

$$k_a = \frac{2D_{AB}}{d_p}\left(\frac{\phi_b}{\pi k_b T}\right)^{\frac{1}{2}} e^{\phi_b/k_b T} \qquad \textbf{(2.18)}$$

where D_{AB} is the Stokes-Einstein particle diffusion coefficient, d_p is the particle diameter, k_b is the Boltzmann constant, T is the temperature, and ϕ_b is the magnitude of the adsorption energy barrier as

visualized in Figure 3. Note that the energy interaction profile is determined by the all surface forces at play and when ϕ_b is of the same order of magnitude as the energy associated with thermal fluctuations, adsorption is diffusion-controlled.

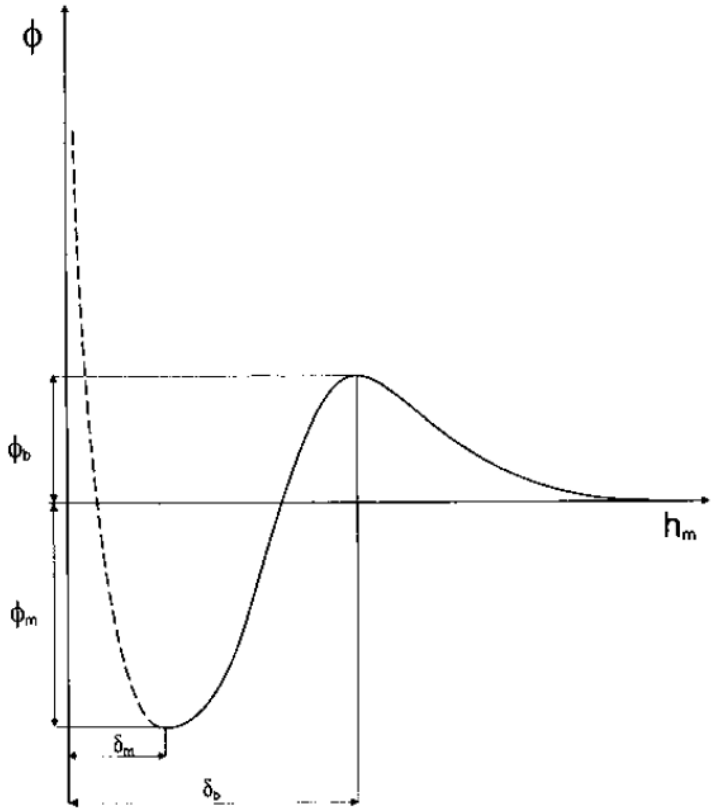


Figure 3: Potential energy profile of a colloidal particle [37]

2.3 Pore Network Modeling

Pore network modeling is a well-established approach for simulating transport phenomena in porous media. Pore network models (PNMs) are an alternative approach to continuum models. Continuum models are generally based on experimentally determined relationships and use much less computational power than pore network models. PNMs can be advantageous to continuum models as they are able to predict the distribution of phases throughout the medium [14]. In this study, OpenPNM

is the PNM package used, which is an open-source package that runs on Python. This study was done using OpenPNM version 2.8.2 [13] and Python version 3.8.12.

To represent the void space of a porous medium using pore network modeling “pores” are mapped onto the void spaces and are connected by “throats”. The sizes and connectivity of pores and throats are set to best match the porous medium. Data about each pore and throat that is used to preform calculations is stored in different “core” objects, which is a subclassed version of Python’s dictionaries. The 5 main “core” objects in OpenPNM are:

- Topology which contains information about the connectivity and location of pores and throats
- Geometry which contains information about the shapes and sizes of pores and throats
- Phases which contain information about the thermophysical properties of each phase
- Physics which takes information from geometry and phases to determine transport parameters such as hydraulic conductance
- Algorithm which contains data that was derived from some algorithm run in the simulation

In this work, the mixed-cell approach [47] is used where the variations of properties of the fluid at the pore-scale are assumed to be negligible. Doing so greatly reduces computational costs but may introduce a small amount of error.

The topology and geometry for the pore networks used in this study is extracted from a 3-dimensional voxel image using PoreSpy’s SNOW algorithm [48]. PoreSpy is an image analysis software that like OpenPNM is also developed by PMEAL and run on Python.

Chapter 3 Model Development

3.1 Generation of Pore Networks

3.1.1 Generation of Two-Dimensional Micromodel

Lacking essential geometric and topological features, two-dimensional pore networks are clearly unable to produce results representative of actual porous media. Before extracting a three-dimensional network from a simulated packing of spheres, however, it is instructive to illustrate key aspects of the simulation using a pore network that represents a simple two-dimensional 50 x 50 square lattice. This network is useful for illustrating how modeling of irreversible adsorption of NPs is done and for revealing the main features of simulation output.

The spacing between pores in this 50 x 50 network is $5x10^{-4}m$, resulting in a $2.5\ x\ 2.5\ cm$ matrix with $n_P = 2500$ and $n_t = 4802$ as seen in Figure 4. The throats connecting inlet (outlet) pores to other inlet (outlet) pores are trimmed, since for constant-pressure boundaries there is no pressure drop and, consequently, no flow along these throats. The 50 x 50 network is built using OpenPNM's "StickAndBall" geometry package, which assigns pore diameters between zero and the largest possible size that does not intersect with its nearest neighbor. This geometry package assigns as throat diameter the diameter of the smallest connected pore.

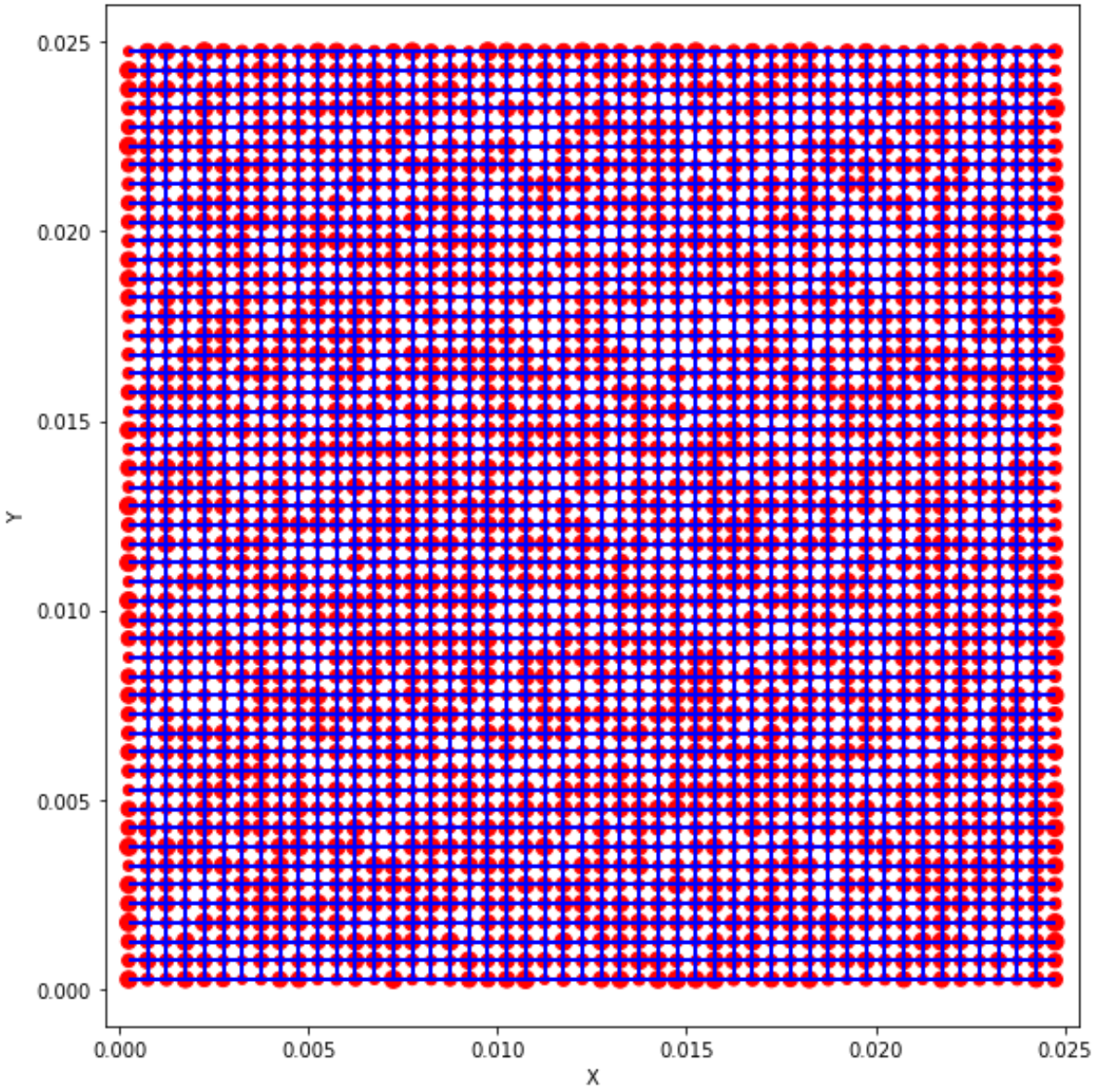


Figure 4: 50x50 square lattice network with pores scaled based on their diameter which are based on OpenPNM's StickAndBall geometry.

3.1.2 Column Packing Generation and Network Extraction

All the simulations run in Chapter 4 and Chapter 5 are done on a pore network extracted from a voxel image representing a column packed with spherical beads. Dispersion experiments on colloidal particles were performed by [21] using glass columns with a diameter of d_c = 2.5 cm and length of either L_c = 15 or 30 cm were filled with glass spheres with a diameter of d_b = 2 mm. The porosity ε of these columns was found to range between 0.39 and 0.43, variation which is indicative of the extent of reproducibility of sphere packing. The pore networks used in this study are based on the packed glass columns used by [21], and in Section 4.3 the results from [21] are simulated and analyzed. Pore networks representing columns.

An artificial voxel image representing the packed column was generated using PoreSpy's "pseudo_gravity_packing" image generator [48]. The voxel resolution of this image is defined in terms of the radius of one of the spheres, which in this case is 28 voxels (1 mm = 28 voxels). The resulting simulated packing matching the dimensions of the column used in [21] has a diameter of d_c = 700 voxels and a length L_c= 4200 voxels. A total of 10916 spheres were added to the column, which results in a porosity of $\varepsilon = 0.390$ after subtraction of the solid volume from the total volume as seen in Figure 5(a).

Table 1: Data for two packed columns

	Column #1	Column #2
Bead Count	10916	10918
ε	0.390	0.390
n_p	38816	38683
n_t	124408	123980

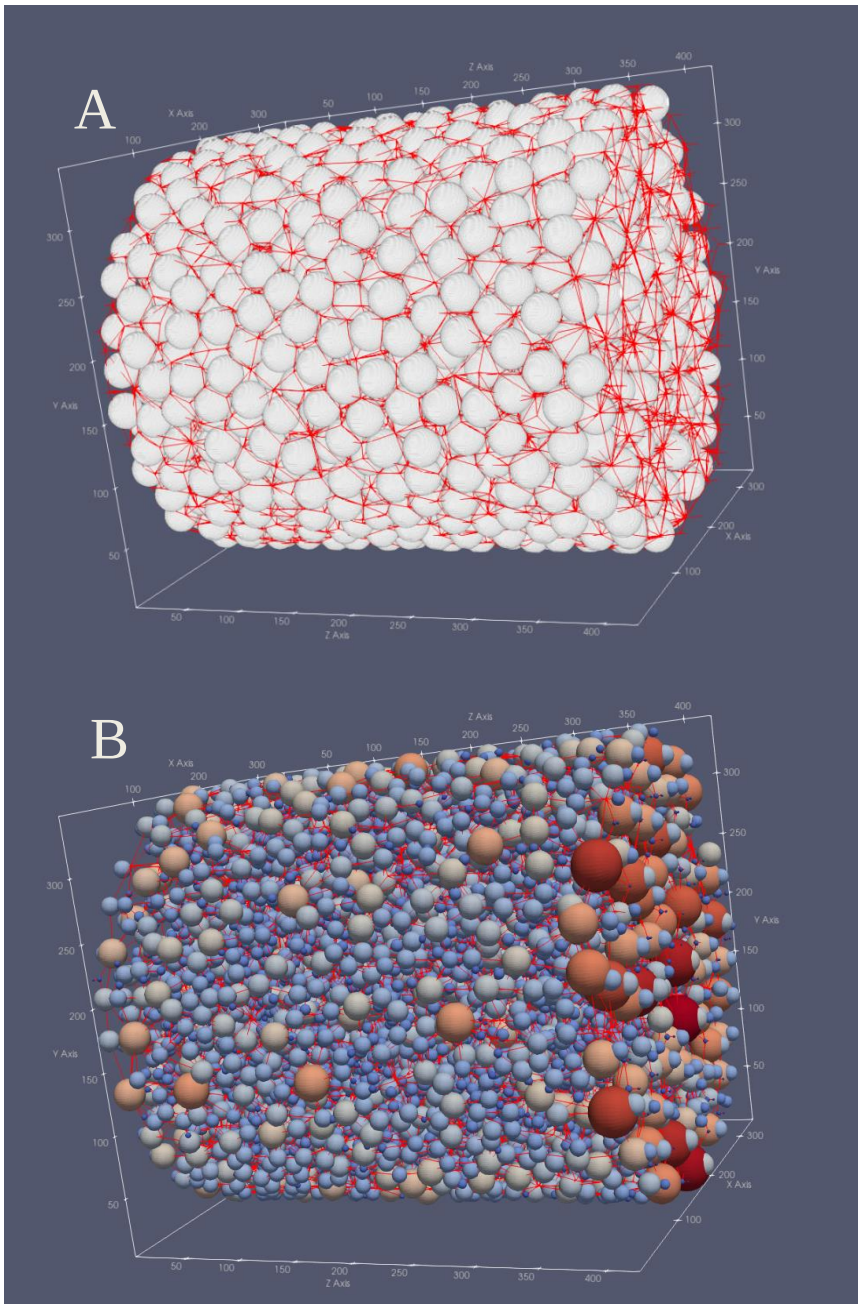


Figure 5: (a) Voxel image of column with a radius of r=28 voxels with throat connectivity represented in red (b) extracted network from the above voxel image with pores represented as spheres and color-coded for size. The pores at the end of the z-axis are noticeably larger because of the voids created when no more beads can be added to the column.

Once the voxel image of the sphere packing is generated, the pore network is extracted from the voxel image using PoreSpy's SNOW algorithm, according to which the connectivity between the pores and throats is defined in the topology object, and the sizes of the pores and throats are defined in the geometry object. A required input for the SNOW algorithm is the voxel size. Table 1 presents data for two different voxel images generated using the same parameters (*i.e.* two different realizations of the packing). Most simulations done in this paper will use Column #1, and Column #2's only purpose is to verify that the random placement of beads does not play a significant role in simulations. A visualization of a shorter spherical packing is presented in Figure 5a along with the extracted pore network in Figure 5(b). In Figure 5(b) the pores are shown to be spheres, but this is simply due to limitations with Paraview. There are several shapes that the pore bodies can be defined as, and in this work, it was found that treating them as truncated pyramids and the throats as cuboids yields the most representative results of real-world packed columns.

3.2 Hydraulic and Diffusive Conductance for Truncated Pyramids and Cuboids

In this section it is shown how to compute the hydraulic and diffusive conductance for truncated pyramids and cuboids. A similar derivation for cylinders and spheres is simpler and is omitted for brevity. In Section 4.1 the suitability of using truncated pyramids to represent pores and cuboids to represent throats is justified by comparing simulated results for permeability, tortuosity, and breakthrough pressure to known correlations for packed columns.

The values for both the diffusive and hydraulic conductance are determined using functions for the cross-sectional area of the ½ pore-throat-½ pore conduit. Figure 6 presents this assembly for when truncated pyramids are used for each ½ pore and cuboids are used for the throat.

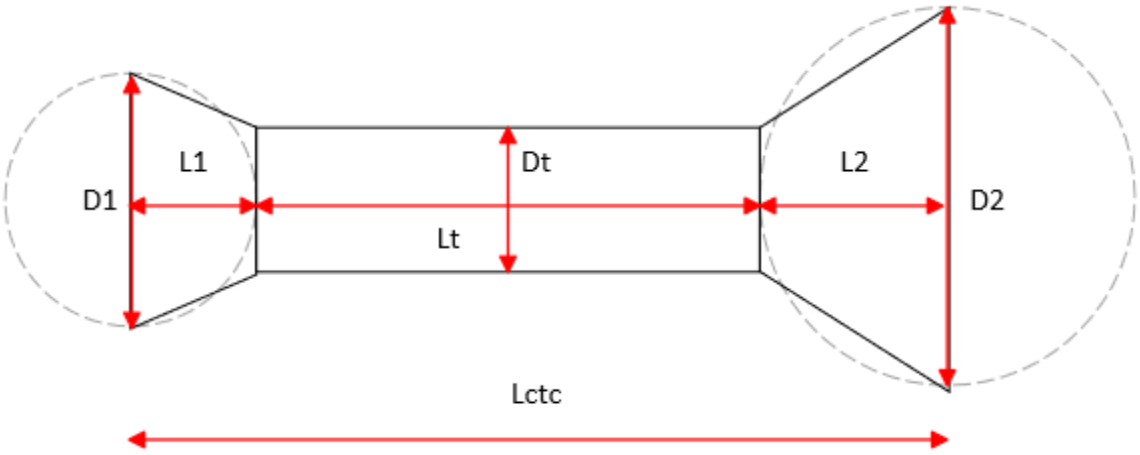


Figure 6: ½ pore-throat- ½ pore conduit for truncated pyramids as pores and cuboids as throats

The conduit-to-conduit length L_{ctc} and the pore and throat diameters are defined in the SNOW extraction. The conduit lengths then can be determined using

$$L_1 = \frac{D_1}{2}$$

$$L_2 = \frac{D_2}{2} \qquad \textbf{(3.1)}$$

$$L_t = L_{ctc} - (L_1 + L_2)$$

The cross-sectional area of the cuboid section of the assembly is found with $A_t = D_t^2$. The cross-sectional area of the truncated pyramid parts of the assembly can be found using

$$r_i(x) = \frac{D_i}{2} - \left(\frac{x}{L_i}\right)\frac{D_i - Dt}{2}$$

$$A_i(x) = \left(2r_i(x)\right)^2 \qquad \textbf{(3.2)}$$

where $x = 0$ at the intersection of the pore and throat, and $x = L_i$ at the center of the pore.

The hydraulic conductance for each segment of the conduit can be derived from Eqn.(3.3 from [49].

$$g^h = \frac{\Delta p}{Q} = 16\pi^2\mu \int_{x_1}^{x_2} \frac{I_p^*}{A^2} dx \qquad \textbf{(3.3)}$$

where I_p^* is the specific polar moment of cross-sectional inertia [50] which can be determined using $I_p^* = I_p/A^2$, where $I_p = \int_A (y^2 + z^2) dA$. It can be shown that $I_p^* = 1/6$ when the cross-section is a square, which is conveniently what the cross-section is for all points along the x-axis for both truncated pyramids and for cuboids. Eqn.(3.3 is thus simplified to

$$g^h = \frac{16}{6}\pi^2\mu F \qquad \textbf{(3.4)}$$

where

$$F = \int_{x_1}^{x_2} \frac{1}{A^2} dx \qquad \textbf{(3.5)}$$

The integral is then solved for the two ½-pore segments and the throat segments using Eqns. (3.1 and (3.2:

$$F_1 = \frac{(D_1^2 + D_1 D_t + D_t^2) L_1}{3(D_1 D_t)^3}$$

$$F_2 = \frac{(D_2^2 + D_2 D_t + D_t^2) L_2}{3(D_2 D_t)^3} \qquad \textbf{(3.6)}$$

$$F_t = \frac{L_t}{D_t^4}$$

The values for hydraulic conductance can be found by combining Eqns. (3.4 and (3.6.

Solving for the diffusive conductance is much simpler since the integral to compute for *F* only has the area in the denominator, as opposed to area squared. The diffusive conductance and size factors are solved using Fick's law

$$g^d = \frac{D_{eff}}{F} \qquad \textbf{(3.7)}$$

where:

$$F = \int_0^{L_i} \frac{dx}{A(x)}$$

$$F_1 = \frac{L_1}{D_1 D_t}$$

$$F_2 = \frac{L_2}{D_2 D_t} \qquad \textbf{(3.8)}$$

$$F_t = \frac{L_t}{D_t^2}$$

3.3 Steady Fickian Diffusion and Stokes Flow in Pore Networks

For steady incompressible flow of a Newtonian fluid, a mass conservation equation for each pore *i* in the network is solved for the unknown fluid pressures,

$$\sum_{j=1}^{n_i} G_{ij}^h (p_i - p_j) = 0, i = 1,2, \dots, n_p \qquad \textbf{(3.9)}$$

where *i* is the pore of interest, *j* are all neighboring pores, n_i is the number of neighbor pores to pore *i*, n_p is the total number of pores in the network, p_i is the pressure in pore *i*. Furthermore, G_{ij}^h is the hydraulic conductance of the pore-throat-pore assembly, also known as the hydraulic conduit conductance, which can be found by

$$G_{ij}^h = \left(\frac{1}{g_i^h} + \frac{1}{g_{ij}^h} + \frac{1}{g_j^h} \right)^{-1} \qquad \textbf{(3.10)}$$

where g_i^h is the hydraulic conductance of pore *i*, g_j^h is the hydraulic conductance of pore *j*, and g_{ij}^h is the hydraulic conductance of the throat connecting pores *i* and *j*. Following solution of the linear algebraic equation system for pore pressure, the flow rate across any throat q_{ij} can be computed using $q_{ij} = G_{ij}^h (p_i - pj)$. It is also necessary to determine diffusive conduit conductance to solve advection-diffusion problems. The conduit conductance appropriate for diffusion, G_{ij}^d, is computed in a similar fashion.

$$G_{ij}^{d} = \left(\frac{1}{g_i^d} + \frac{1}{g_{ij}^d} + \frac{1}{g_j^d} \right)^{-1} \quad \textbf{(3.11)}$$

The solutions for hydraulic and diffusive conductance for individual pores and throats depends on the defined shapes of the pores and throats.

Once the hydraulic and diffusive conductance are known it is possible to determine the net mass flow rate across the pore network, which is done by appropriately summing mass fluxes, $\dot{m}_{ij}$, through individual throats. The species mass balance at steady state on any given pore can be defined as follows:

$$\sum_{j=1}^{n_i} \dot{m}_{ij} = \sum_{j=1}^{n_i} G_{ij}^{AD} C_i - \sum_{j=1}^{n_i} G_{ji}^{AD} C_j = 0, i = 1,2, \dots, n_p \quad \textbf{(3.12)}$$

where G_{ij}^{AD} and G_{ji}^{AD} are the two advective-diffusive conductance values for a given throat ij, and C_i and C_j are the concentrations of pores i and j. This material balance is obviously applicable when there is no advection, such that $G_{ij}^{AD} = G_{ji}^{AD} = G_{ij}^{d}$. When both advection and diffusion take place, the advective-diffusive conductance will be different depending on whether the flux being considered is in the same direction of advective flow or against it. Thus, G_{ij}^{AD} refers to the advective-diffusive conductance in the same direction as hydraulic flux and G_{ji}^{AD} is when it is in the opposite direction of hydraulic flux.

OpenPNM offers 4 different discretization schemes to solve for G_{ij}^{AD} and G_{ji}^{AD}: upwind, hybrid, power-law, and exponential. These 4 schemes were analyzed in [51], and it was found that the power-law and exponential schemes yielded the most accurate results. However, in this work the hybrid scheme was selected due to convergence issues with the exponential and power-law schemes in the presence of reaction. In [51] it was found that the hybrid scheme can result in moderate amounts of error but is more significant in smaller networks. It was also found that relative error is only significant when the length of the network along the direction of flow is less than 20 pores. The number of pores in the networks used in this study is greater than 10000, thus the potential error from using the hybrid scheme is assumed to be negligible. The upwind scheme to solve for the two advective-diffusive conductance at steady state is presented in Eqn.(3.12.

$$G_{ij}^{AD} = \max\left[q_{ij}, \left(G_{ij}^{d} + \frac{q_{ij}}{2}\right), 0\right]$$
$$G_{ji}^{AD} = \max\left[-q_{ij}, \left(G_{ij}^{d} - \frac{q_{ij}}{2}\right), 0\right] \quad \textbf{(3.13)}$$

To solve transient advection-diffusion problems, Eqn. (3.2 is discretized using the Implicit scheme, which is also known as the Backwards Euler scheme, which can be found in Section 3.4.

3.4 Implicit Discretization

In Section 3.3, a mass balance for the advection-diffusion problem was written for steady state conditions, as given by Eqn. (3.12). In this section, Eqn. (3.12 is discretized to solve Transient Advection-Diffusion (TAD) problems. This discretization is done for simulations with no "sink" terms, which would be a necessary term if a reaction or and an adsorption term were desired. In section 3.7.1 a term accounting for the adsorption term is added to the discretization.

At steady state, the species conservation for the advection-diffusion can be described using the following equation:

$$\sum_{j=1}^{n_i} G_{ij}^{AD}\, C_i - \sum_{j=1}^{n_i} G_{ji}^{AD}\, C_j = 0 \quad \textbf{(3.12)}$$

where G_{ij}^{AD} is the advective-diffusive conductance of the given pore-throat-pore formation, and the subscript *ij* refers to the direction in which the conductance of mass transport is being considered. In transient conditions however, the net flow of material in each pore will vary and result in a net accumulation of R_i for every time step such that

$$\sum_{j=1}^{n_i} G_{ij}^{AD}\, C_i - \sum_{j=1}^{n_i} G_{ji}^{AD}\, C_j = R_i \quad \textbf{(3.14)}$$

where the accumulation rate R_i can be expanded as $R_i = V_i \Delta C_i / \Delta t$ when there is no reaction, where V_i is the volume of the pore in question. The concentration at the next time step, $t + \Delta t$, is

$$V_i \frac{C_i^{t+\Delta t} - C_i^t}{\Delta t} = \sum_{j=1}^{n_i} G_{ij}^{AD} C_i^{t+\Delta t} - \sum_{j=1}^{n_i} G_{ji}^{AD} C_j^{t+\Delta t} \quad \textbf{(3.15)}$$

Rearranging to place the known values on the right-hand side yields:

$$C_i^{t+\Delta t} - \frac{\Delta t}{V_i}\left(\sum_{j=1}^{n_i} G_{ij}^{AD} C_i^{t+\Delta t} - \sum_{j=1}^{n_i} G_{ji}^{AD} C_j^{t+\Delta t}\right) = C_i^t \quad \textbf{(3.16)}$$

This equation has the following general form:

$$C^{t+\Delta t} - \frac{\Delta t}{V_i} A C^{t+\Delta t} = C^{t+\Delta t}\left(I - \frac{\Delta t}{V_i} A\right) = A_I C^{t+\Delta t} = C^t \quad \textbf{(3.17)}$$

Next, $C^{t+\Delta t}$ can be expressed as a system of linear equations

$$C^{t+\Delta t} = A_I^{-1} C^t \quad \textbf{(3.18)}$$

To illustrate the solution of the transient advection-diffusion problem using the implicit scheme, a simple pore network is presented in Figure 7

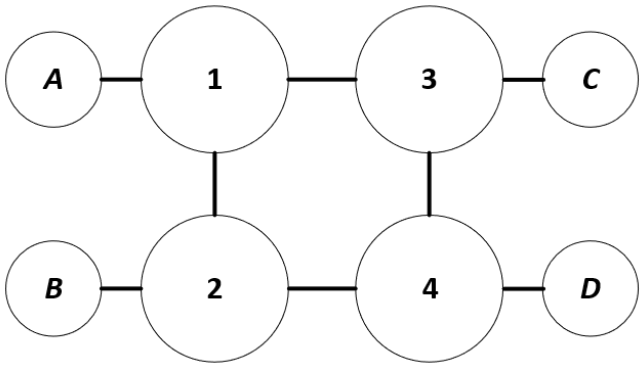


Figure 7: Sample pore network with 4 body pores, 2 inlet pores (*A* and *B*), and 2 outlet pores (*C* and *D*).

To solve the transient advection-diffusion problem, a mass balance equation is written for each pore using Eqn. (3.16:

For pore 1:

$$C_1^{t+\Delta t}\left(\frac{V_1}{\Delta t}-G_{1A}^{AD}-G_{12}^{AD}-G_{13}^{AD}\right)+C_A^{t+\Delta t}G_{A1}^{AD}+C_2^{t+\Delta t}G_{21}^{AD}+C_3^{t+\Delta t}G_{31}^{AD}=C_1^t\frac{V_1}{\Delta t}$$

For pore 2:

$$C_2^{t+\Delta t}\left(\frac{V_2}{\Delta t}-G_{2B}^{AD}-G_{21}^{AD}-G_{24}^{AD}\right)+C_B^{t+\Delta t}G_{B2}^{AD}+C_1^{t+\Delta t}G_{12}^{AD}+C_4^{t+\Delta t}G_{42}^{AD}=C_2^t\frac{V_2}{\Delta t}$$

For pore 3:

$$C_3^{t+\Delta t}\left(\frac{V_3}{\Delta t}-G_{31}^{AD}-G_{34}^{AD}-G_{3C}^{AD}\right)+C_1^{t+\Delta t}G_{13}^{AD}+C_4^{t+\Delta t}G_{43}^{AD}+C_C^{t+\Delta t}G_{C3}^{AD}=C_3^t\frac{V_3}{\Delta t}$$

For pore 4:

$$C_4^{t+\Delta t}\left(\frac{V_4}{\Delta t}-G_{42}^{AD}-G_{43}^{AD}-G_{4D}^{AD}\right)+C_2^{t+\Delta t}G_{24}^{AD}+C_3^{t+\Delta t}G_{34}^{AD}+C_D^{t+\Delta t}G_{D4}^{AD}=C_4^t\frac{V_4}{\Delta t}$$

The resulting coefficient matrix for the network presented in Table 2:

Table 2: Coefficient Matrix for pore network presented in Figure 7

C_1	C_2	C_3	C_4	C_A	C_B	C_C	C_D	b
$\frac{V_1}{\Delta t} - G_{1A}^{AD} - G_{12}^{AD} - G_{13}^{AD}$	G_{21}^{AD}	G_{31}^{AD}	0	G_{A1}^{AD}	0	0	0	$C_1^t \frac{V_1}{\Delta t}$
G_{12}^{AD}	$\frac{V_2}{\Delta t} - G_{2B}^{AD} - G_{21}^{AD} - G_{24}^{AD}$	0	G_{42}^{AD}	0	G_{B2}^{AD}	0	0	$C_2^t \frac{V_2}{\Delta t}$
G_{13}^{AD}	0	$\frac{V_3}{\Delta t} - G_{31}^{AD} - G_{34}^{AD} - G_{3C}^{AD}$	G_{43}^{AD}	0	0	G_{C3}^{AD}	0	$C_3^t \frac{V_3}{\Delta t}$
0	G_{24}^{AD}	G_{34}^{AD}	$\frac{V_4}{\Delta t} - G_{42}^{AD} - G_{43}^{AD} - G_{4D}^{AD}$	0	0	0	G_{D4}^{AD}	$C_4^t \frac{V_4}{\Delta t}$
0	0	0	0	1	0	0	0	C_A
0	0	0	0	0	1	0	0	C_B
0	0	0	0	0	0	1	0	C_C
0	0	0	0	0	0	0	1	C_D

3.5 Dispersion Simulations

3.5.1 Analytical Solutions for Instantaneous and Step-Change Injections

The longitudinal dispersion coefficient, D_L, for a water saturated porous medium is determined by injecting a tracer and fitting an appropriate analytical solution to the resulting elution curve. In this study, two types of tracer simulations were performed: step-change simulations, where a plane at the inlet is maintained at a concentration of $C = C_0$, and pulse simulations where a determined mass of NPs or tracer are injected as an "instantaneous" pulse. Both types of simulations are defined to be one-directional, and colloids are considered a conservative tracer (adsorption on interfaces is ignored). The transport for one-dimensional transport in saturated porous media have been derived by [52]:

$$D_L \frac{\partial^2 C}{\partial^2 x} = U \frac{\partial C}{\partial x} + \frac{\partial C}{\partial t} \qquad \textbf{(3.19)}$$

The boundary conditions for a step-change transport simulation are

$$\begin{aligned} C(x,0) &= 0, \qquad x \geq 0 \\ C(0,t) &= C_0, \qquad t \geq 0 \\ C(\infty,t) &= 0, \qquad t \geq 0 \end{aligned} \qquad \textbf{(3.20)}$$

The resulting analytical solution for the given boundary conditions is

$$\frac{C}{C_0} = \frac{1}{2} erfc\left(\frac{x - Ut}{2\sqrt{D_L t}}\right) + \frac{1}{2} exp\left(\frac{Ux}{D_L}\right) erfc\left(\frac{x + Ut}{2\sqrt{D_L t}}\right) \qquad \textbf{(3.21)}$$

where *erfc* is the complementary error function, *x* is the point along the column where the elution curve is being solved for, and *U* is the interstitial velocity. The analytical solution for an instantaneous pulse injection has been derived by [53] The boundary conditions for an instantaneous pulse are:

$$\begin{aligned} C(x,0) &= 0, \qquad x \geq 0 \\ C(0,t) &= \begin{cases} C_0, & 0 < t \leq t_0 \\ 0, & t > t_0 \end{cases} \\ C(\infty,t) &= 0, \qquad t \geq 0 \end{aligned} \qquad \textbf{(3.22)}$$

where t_0 is the time length of the injection, $C_0 * Q * t_0 = M_{inj}$. The resulting analytical solution for an instantaneous pulse is:

$$C_A = \frac{M_{inj}}{A\epsilon(4\pi D_L t)^{0.5}} exp\left(-\frac{(x-Ut)^2}{4D_L t}\right) \quad \textbf{(3.23)}$$

3.5.2 Fitting Procedure

Fitting was carried out using both Microsoft Excel's Solver tool and scipy's curve_fit function in Python. It was found that both options gave the exact same result. The residual sum of squares, RSS, for the average eluted concentration was determined using the least squares method:

$$RSS = \sum_{i=1}^{n} \left(C_e^{sim} - C_e^{fit}\right)^2 \quad \textbf{(3.24)}$$

where *n* is the number of time-steps run in the simulation, C_e^{sim} is the average eluted concentration found in the simulation at the n^{th} time step, and C_e^{fit} is the concentration found using the analytical solution at the n^{th} time step.

In all simulations, both D_L and *U* were determined by fitting the analytical solution to the elution curve found in the simulation. When a pulse was fitted, by fitting the interstitial velocity, *U*, the effect that including the velocity profile exclusion effect has on the model can be determined by comparing the input interstitial velocity, $U^{input} = U_{Darcy}/\varepsilon$, to the fitted interstitial velocity, U^{fit}. It is expected that if the particle is sufficiently large in relation to the bead size, then U^{fit} will be greater than U^{input} because of the enhanced velocity that the NPs experience due to the VPE effect. Another way to think of the U^{input} and U^{fit} terms is that U^{input} refers to the interstitial velocity of the solvent and U^{fit} refers to the interstitial velocity of the NPs.

3.5.3 Running Dispersion Simulations

The concentration of particles injected for both step-change and pulse injections will have no effect on the final dispersion coefficient. For simulations, however, the volume fraction was set to $\Phi_V = 0.0001$.

The input velocity is specified in terms of the Darcy velocity, U_{Darcy}, even though the velocity used to fit Eqns. (3.21 and (3.23 is the interstitial velocity.

In Figure 8 a flowchart of the process of running TAD simulations to solve for D_L and U_{fitted} is presented. The first step in running these simulations is to set the inputs for the simulation which include:

- Particle size, d_p
- Target Darcy Velocity, U_{Darcy}
- Phase Data. For all simulations in this paper the "water" phase which is already defined in OpenPNM is used
- Geometry and Topology data. For the packed columns the data was loaded from the columns extracted in Section 3.1.2
- Number of time-steps. For all packed column simulations 1000 steps are used.

The first step in running the dispersion simulation is to determine the permeability which would then be used to achieve the desired pressure gradient to achieve the target velocity. This is done by running a Stokes flow algorithm with an arbitrary pressure gradient across the pore network. This algorithm would then yield a corresponding flow rate for the arbitrary pressure gradient, and then using Darcy's law in Eqn (2.2 is used to calculate the simulated permeability k_s. Once the simulated permeability is determined another Stokes flow algorithm is run with the necessary pressure gradient to achieve the target velocity.

It is necessary to determine the pressure gradient (from knowledge of the permeability) to achieve the desired U^{input} prior to adjusting the diffusivity and conductance values to account for HD and VPE effects since accounting for VPE effects require the velocity of all throats as outlined in 2.2.2. After making the adjustments to account for the HD and VPE effects, the Stokes flow algorithm is executed again, since the velocity will now be different given the adjusted hydraulic conductance values.

OpenPNM requires knowledge of steady flow velocities as input to transient advection-diffusion (TAD) computations. Some additional inputs that are used in the TAD algorithm are:

- Instantaneous pulse or step change injection

- The length of each timestep, which is calculated by dividing the number of timesteps by the time it would take to flow 1.5 PV
- The discretization scheme, as outlined in Section 3.4

A total of 1.5 pore volumes are injected for the entire simulation, and 1000 time-steps are used for the TAD. The eluted concentration is determined for each time step using

$$C_e = \frac{\sum_{n_{ot}} C_t \, q_t}{\sum_{n_{ot}} q_t} \qquad \textbf{(3.25)}$$

where q_t is the flow rate of the given throat, C_t is the concentration of the given throat, and this is summed for n_{ot}, all outlet throats. Once C_e is determined for all time steps, either Eqn. (3.21 or (3.23 is fitted to the elution curve using the method described in Section 3.5.2 to determine D_L and U_{fitted}.

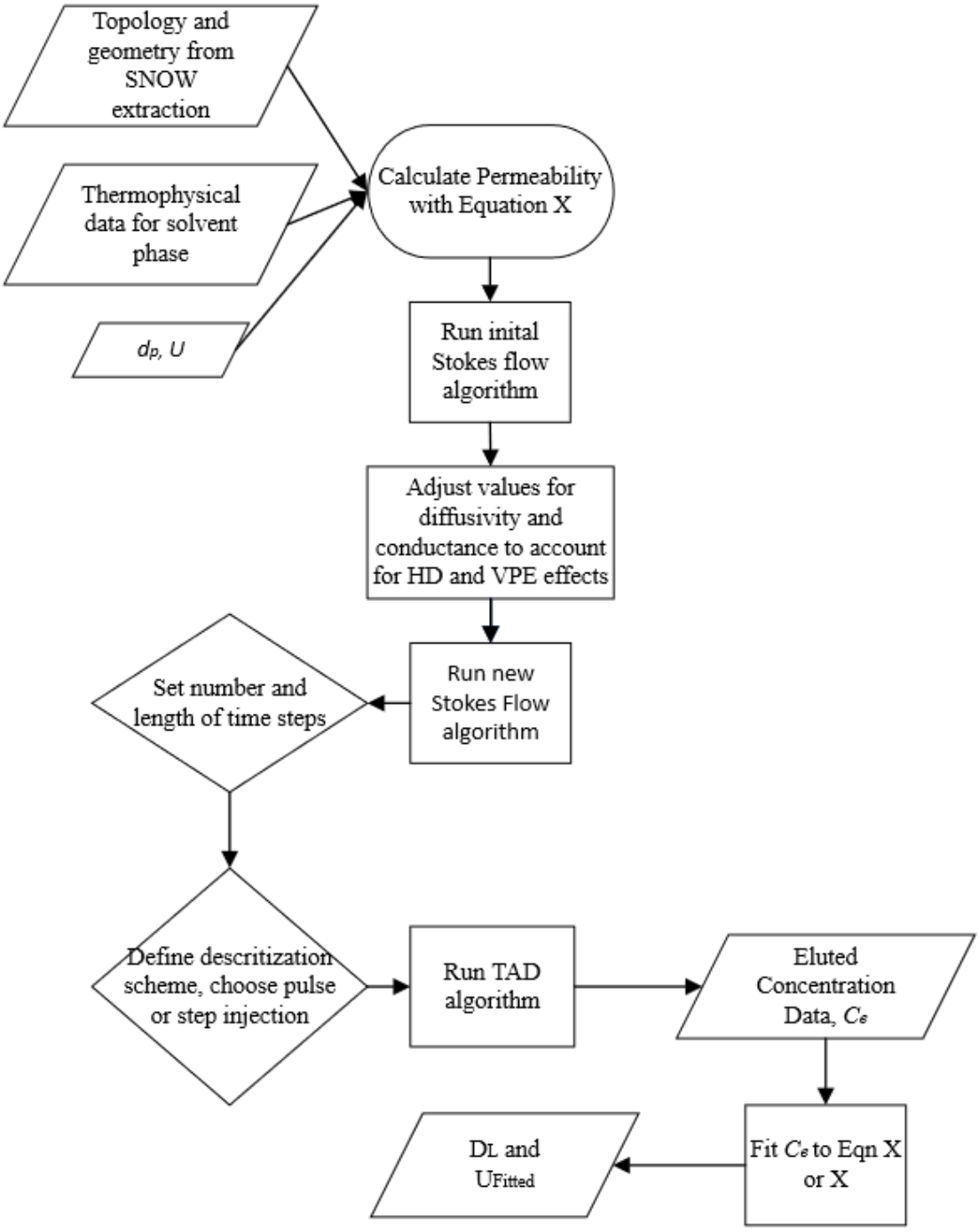


Figure 8: Flowchart for running dispersion simulations

3.6 Creation of Immobile Non-Wetting Phase

In this work we consider the water phase as the only mobile phase in the presence of a second immobile (trapped) non-wetting phase (oil or air). To simulate an immobile oil phase, blobs of oil are generated using OpenPNM's "InvasionPercolation" (IP) algorithm, where a specified number of pores are selected, and the oil phase is allowed to invade the network until a specified saturation level is achieved. The pores and throats successfully invaded by the oil phase are recorded in the model, and the physics are adjusted such that there is no flow through these pores and throats.

3.6.1 Invasion Percolation

The first step in running an IP simulation is to define the invading phase. In this thesis, the only invading phase used is a custom "Oil" phase. OpenPNM has predefined properties for "Air", "Water", and "Mercury" phases, but to create an "Oil" phase the thermophysical properties must be defined by the user. In this study, since the non-wetting phase (NWP) is immobile, the thermophysical properties are of little significance. This new NWP has a property called "occupancy" which will be "True" if the specified pore or throat contains the NWP, and "False" if it still contains the "Water" phase.

Once the thermophysical properties of the invading phase are defined, the desired pores from where the invasion is to start must be specified. The user provides the number of oil blobs desired by specifying the number of inlet pores where the invasion takes place. For all simulations preformed on the three-dimensional packed column, 100 pores are randomly selected. After running the IP algorithm, the NWP saturation, S_{NWP}, is defined by the user and the pore occupancy is updated to achieve the desired saturation. Next, all trapped water pores are converted to oil pores. Thus, the actual value for non-wetting phase saturation is slightly higher than the input saturation. Finally, all throats connecting two oil pores are set to contain oil as well.

Figure 9 presents a sample result of the IP algorithm run on the 50×50 pore network presented in Figure 4 with a $S_{NWP} = 0.10$, and 10 randomly selected pores to act as inlet sites. Of these 10 inlet pores, 2 of them did not contribute to any additional pores when running the IP algorithm. This is a result of the connecting throats to these pores not being sufficiently large enough to result in any fluid invasion.

Since particle adsorption on water-NWP interfaces is assumed to take place only in NWP-occupied pores which are connected to water saturated pores - these pores were designated as "NWP surface pores". To

quantify the extent of particle adsorption throughout the entire network, the total surface area of all surface pores can be summed to find $A_{NWP,surf}$, which is found by summing the surface area of all "NWP surface pores" in the network.

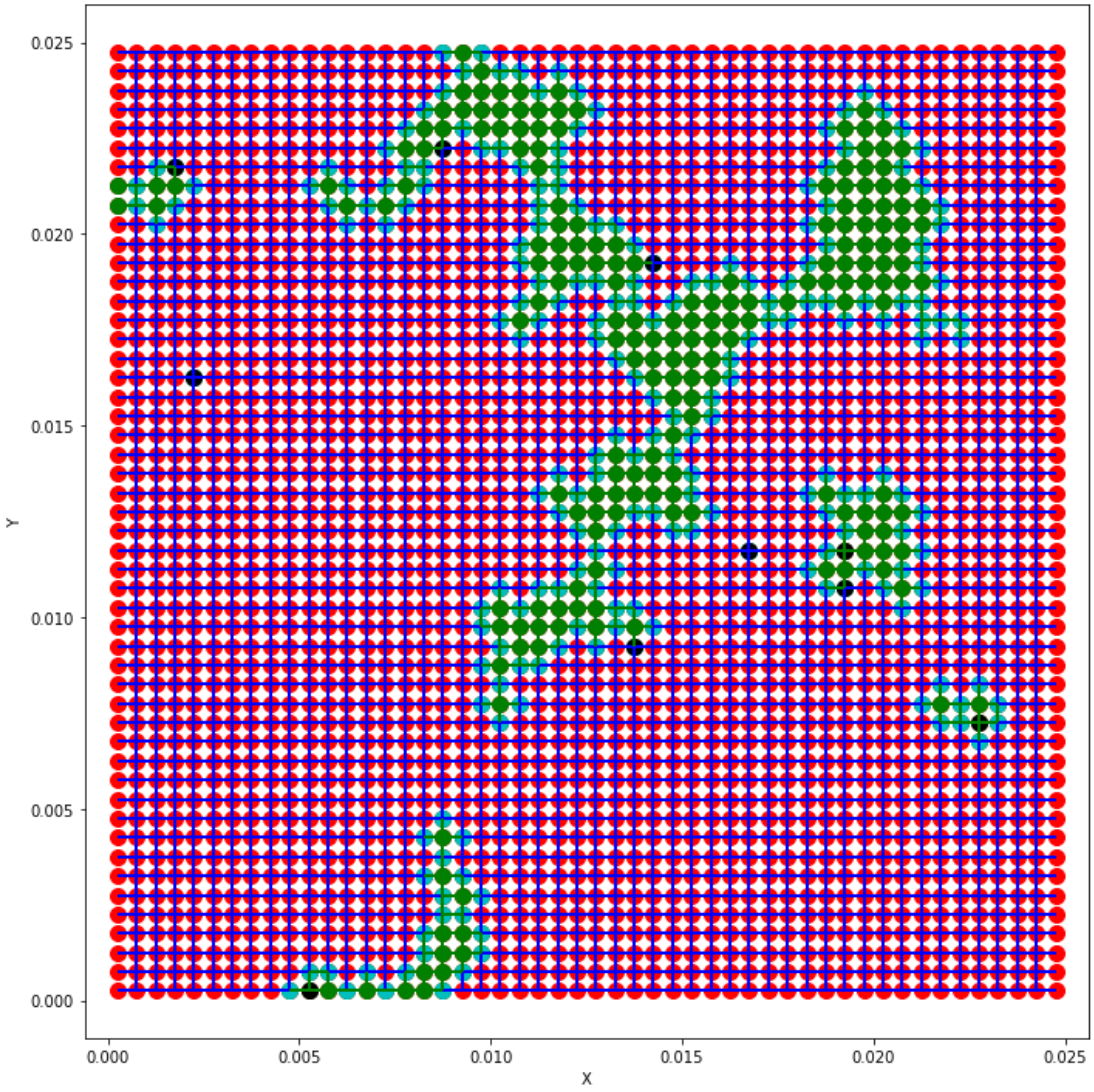


Figure 9: 50x50 square lattice pore network with $S_{NWP} = 0.10$ and 10 randomly selected pores as inlet pores for the IP algorithm. Water pores are presented as red, water throats are in blue, trapped oil pores are in green, surface oil pores are in cyan, and IP inlet pores as black.

3.6.2 Accounting for the Presence of Non-Wetting Phase

Prior to running any Stokes-flow or advection-diffusion simulations, the values for hydraulic, diffusive, and advective-diffusive conductance are updated. Clearly, it is desired for there to be no advection or diffusion in pores or throats where there is a NWP, so all three conductance terms are set to zero. Additionally, if a throat is connected to a water-filled pore on one end and an NWP-filled pore on the other end, there would be no advection, and all transport is assumed to be diffusive. Thus, the diffusive conductance is kept as is, the hydraulic conductance is set to zero (rather, a very small value 10^{-50}), and the advective-diffusive conductance is re-calculated using Eqn. (3.13.

The quantity of particles in each pore is stored as concentration in units of particles per cubic meter in the "water" *dict* in these simulations. This quantity can generally be non-zero for "surface oil pores". This is because the only way for the solute to end up in a NWP pore is through diffusion through a water saturated throat connected to the NWP pore. Therefore, NWP pores that are not connected to water throats have no possible ways to get a non-zero concentration of solute. It is also possible for the solute to diffuse back out through a different water throat, which is why it was important to convert "trapped water" pores to oil pores prior to doing any simulations.

It may seem odd to refer to the concentration of particles in a NWP pore when the concentration is determined from the volume of the entire pore, but this assumption is necessary in preforming these simulations. It has been shown that a film of water will typically form between the solid and the NWP, but this would only represent a small fraction of the total volume of the pore.

Figure 10 presents the same pore network and oil characterization as Figure 9 with a Stokes-Flow algorithm that is generated with a pressure gradient of $\Delta P = 1.0\ Pa$ across the entire network. The flow path is clearly greatly influenced by the presence of oil filled pores and a significant part of the network experiences slow-flow or near-stagnation conditions.

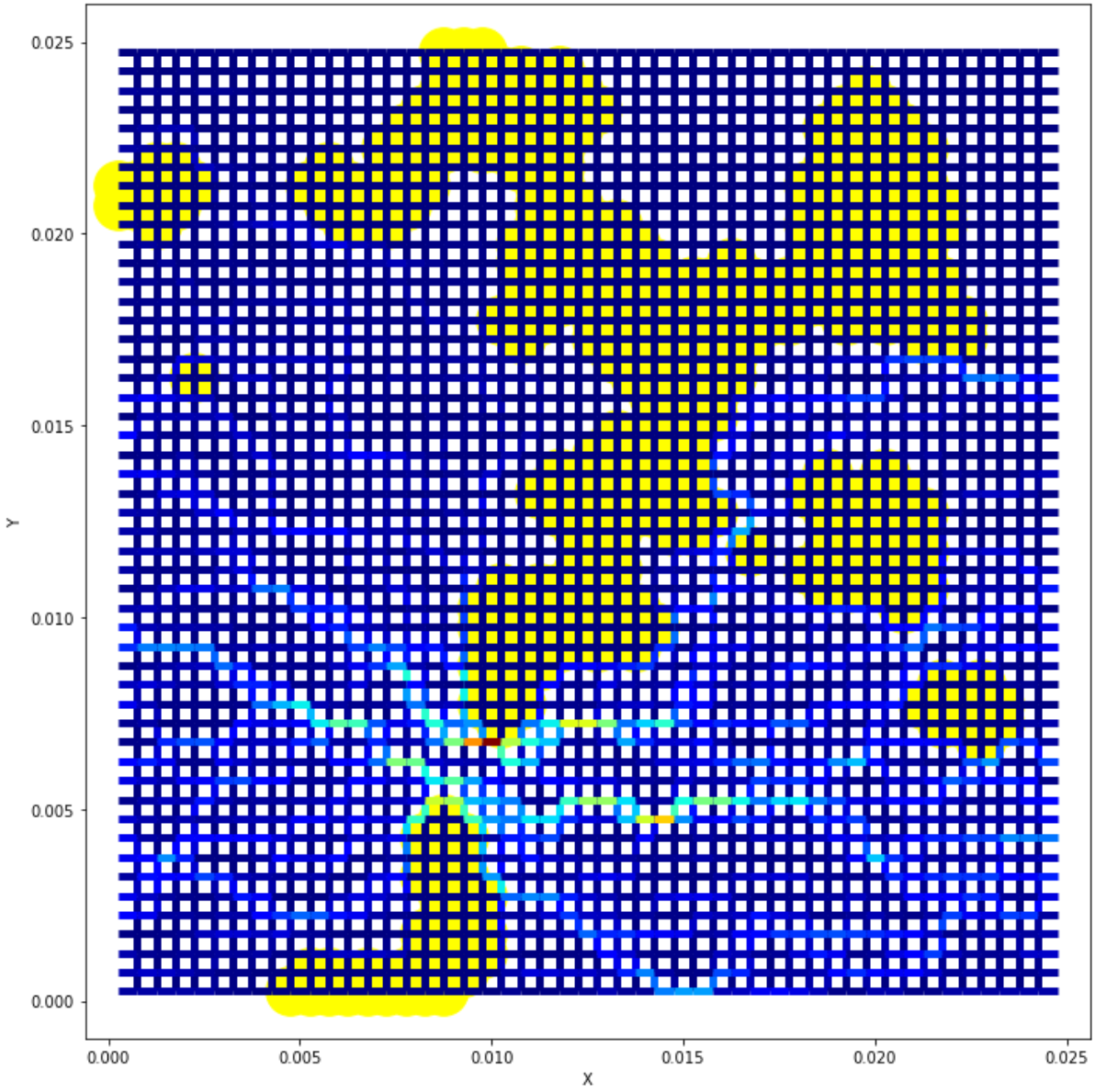


Figure 10: 50x50 two-dimensional network with same oil characterization as Figure 9. Oil pores are shown as yellow blobs, and all throats are shown with their color based on the velocity.

3.7 Modeling Nanoparticle Adsorption onto Trapped Non-Wetting Phase Ganglia

3.7.1 Discretization of Adsorption Equations

The simulations for modeling particle adsorption onto trapped NWP ganglia are like simulations to solve for the longitudinal dispersion coefficient in that they both use OpenPNM's TransientAdvectionDiffusion algorithm to simulate the transport of particles through a porous medium. The key difference in the two simulations is accounting for adsorption through an additional term added to Eqn. (3.14, which results in

$$\sum_{j=1}^{N_i} G_{ij}^{AD} C_i - \sum_{j=1}^{N_i} G_{ji}^{AD} C_j - r_i^{ads} = R_i \qquad \textbf{(3.26)}$$

The discretization of this equation is effectively the same as in Section 3.4 only with the term r_{ads} added, which refers to the rate of particles removed from the bulk concentration due to adsorption for an individual pore. Eqn.(3.27)is solved for all pores in the network for each time step, but r_i^{ads} will be equal to zero for all non-oil pores. r_i^{ads} is solved using

$$r_i^{ads} = \frac{dn_{ads}}{dt} = A_s k_a \bar{B}(\theta) C_0 \qquad \textbf{(3.27)}$$

where A_s is the total surface area of the pore where adsorption is taking place, k_a is the reaction constant that depends on the particle diameter, $\bar{B}(\theta)$ is the blocking function calculated using Eqn. (2.17, and C_0 is the concentration of particles in solute in the pore. The fractional coverage of a pore, θ, is found using Eqn. (2.16. Although the pores are treated as cuboids to calculate hydraulic and diffusive conductance in Section 3.2, the surface area is calculated as if the pores are spheres such that $A_s = \pi d_{pore}^2$. k_a is determined using Eqn. (2.18 and will be analyzed later in section 5.3.2.

For each time step, the number of particles removed from the solute for each pore is recorded and saved. The number of particles adsorbed for an individual time step can be calculated simply by $\Delta N_{ads} = r_{ads} \Delta t$ where Δt is one time-step. The total number of particles adsorbed is then solved by

$$n_{ads}^{t} = \Delta n_{ads} + n_{ads}^{t+\Delta t} \qquad \textbf{(3.28)}$$

The total number of adsorbed particles is then used to calculate a new value, θ, which is then used to calculate a new $\bar{B}(\theta)$, and then the process is repeated until the final step.

3.7.2 Adsorption Simulation with Two-Dimensional Model

In this subsection a sample adsorption simulation on an NWP with advection-diffusion is performed on the oil ganglia seen in Figure 9. A flowchart for the process of running adsorption simulations is presented in Figure 11. There are several variables to be selected for the simulation including:

- Network's geometry and topology
- NWP characterization (S_{NWP}, Location of seeds, number of seeds)
- Particle size
- Velocity
- Volume fraction of particles
- Instantaneous pulse, slug, or step change injection
- Number and duration of each time step

It was decided to define the concentration of injected particles injected in terms of the volume fraction of particles and then to convert it to the concentration of particles per cubic meter using $C = \Phi_V / V_p$, where Φ_V is the volume fraction of particles. In simulations on the packed column, the velocity can be defined as a variable by using Darcy's law, but for this two-dimensional simulation only the pressure gradient across the network is defined. The addition of a Stokes Flow algorithm and inclusion of HD and VPE effects are the same as was done with dispersion simulations as outlined in Section 3.5.3. Adsorption simulations also use a TAD algorithm similarly to dispersion simulations, but with a key difference being that at each time step in the TAD algorithm the physics is updated to account for the blocking of the interface.

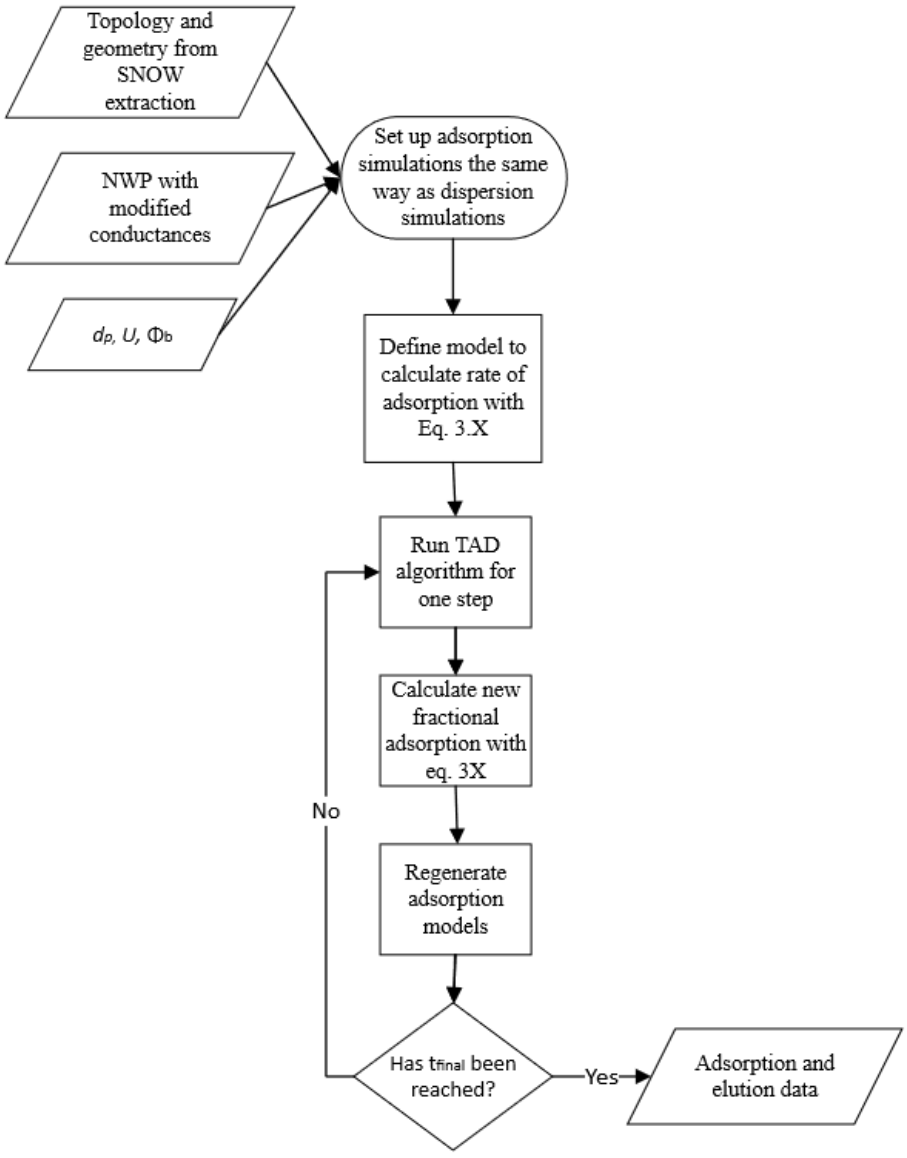


Figure 11: Flowchart for adsorption simulations

A sample simulation was preformed using $20\ nm$ NPs with a pressure gradient of $\Delta p = 1.0\ Pa$. The dimensionless energy barrier, ϕ_b/k_bT, was taken to be equal to 8. This yields an adsorption constant of $k_a = 1.52x10^{-6}m/s$, which is of the order ($O(10^{-6}m/s)$) of adsorption constants found in similar experiments [38, 54].

The concentration of particles at the injection are defined in terms of volume fraction of particles, but during the simulations they are calculated with the units of particles per cubic meter. In this simulation, the volume fraction of particles *is* $\Phi_V = 0.001$ The conversion of volume fraction to particles per cubic meter is calculated using

$$C = \frac{\Phi_V}{\frac{\pi}{6} d_p^3} \quad \textbf{(3.29)}$$

A slug with total of 20 μg of particles is injected. The conversion of particle concentration to mass concentration can be found by multiplying the concentration of particles by the volume and density of particles. For this simulation, the density of ethyl cellulose NPs ($\rho_{EC} = 1140 g/L$) was used [38]. The TAD algorithm is then preformed for 100 steps at $t_{step} = 200s$.

Figure 12 presents color maps of the particle concentration at nine different time steps, while Figure 13 presents color maps of the fractional coverage on the oil pores at the same 9 different time steps. Comparing Figure 12 and Figure 10 it is clear that oil pores that are near to the main flow path expirence adsorption quicker than oil pores further away. Figure 13 further makes it clear that water-NWP interfaces closer to main flow fields will experience adsorption quickly, where pores that are further away will still experience adsorption at a slower rate. Figure 15 presents the fractional adsorption of all oil pores as a function of time. Some of the pores approach fully blocked coverage ($\theta_{max} = 0.91$) at the start of the injection quickly, while other pores do so quickly after some time. The amount of time it takes for the pores to experience blocking depends on their distance from the inlet of the network. It can further be observed that the rate that pores approach maximal blocking is quite different. The pores that have a faster rate of adsorption are the ones that are closest to the primary flow field, while the pores that experience adsorption at a slower rate must have the particles diffuse towards them which takes quite a bit longer.

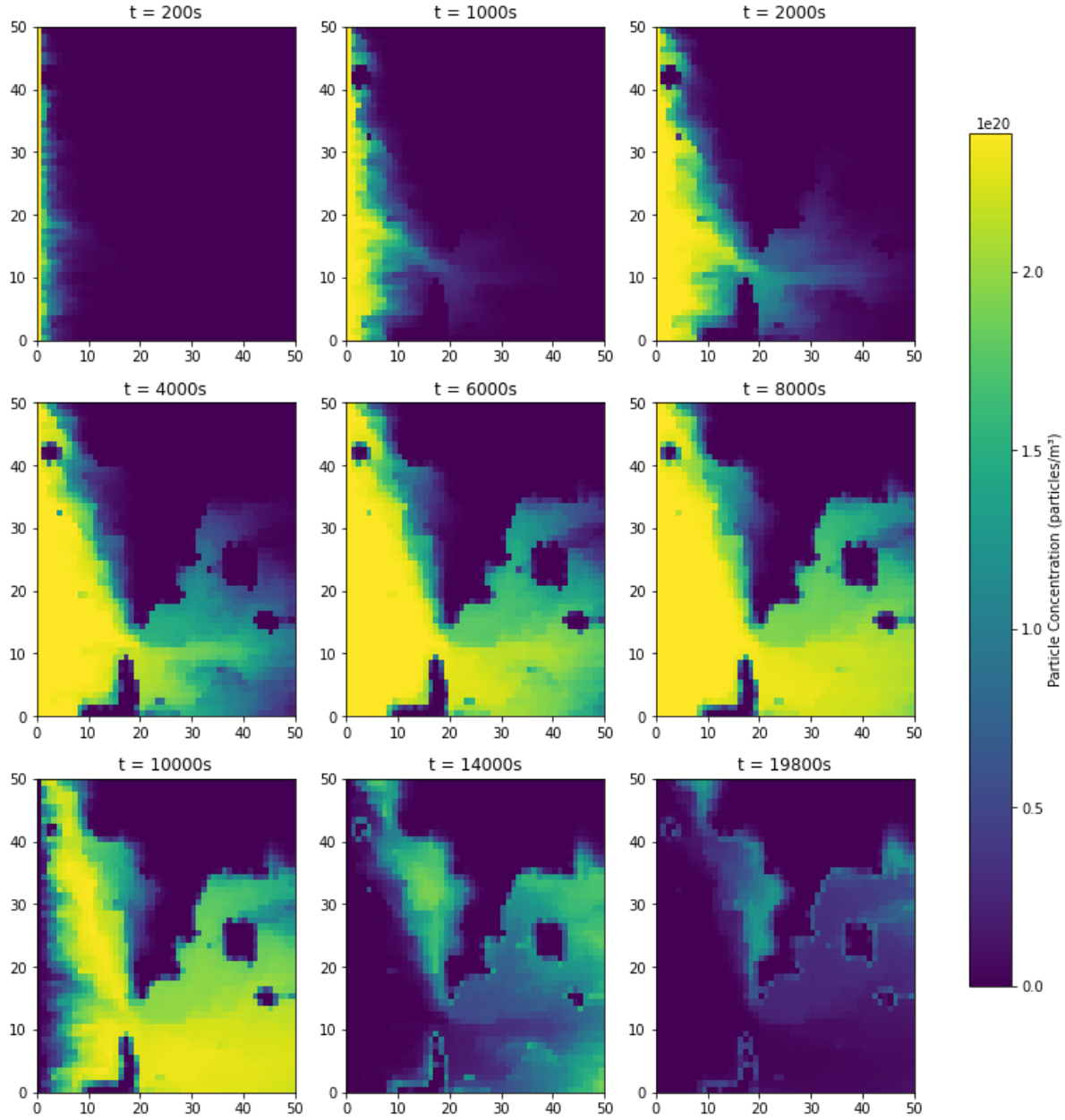


Figure 12 Particle concentration heatmap with oil blob at 9 different time steps for 50x50 2-D network with NWP characterization from Figure 9

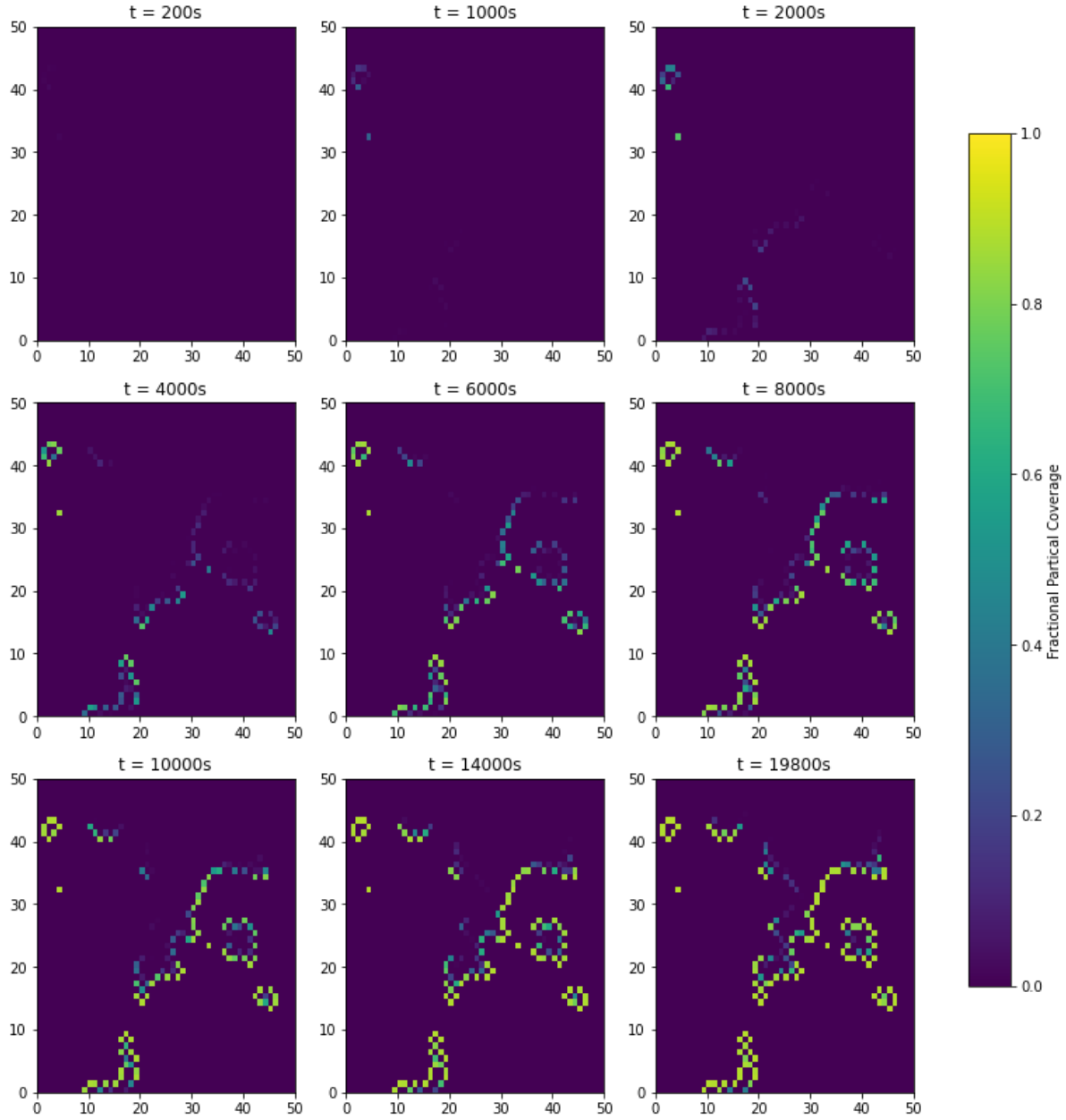


Figure 13: Fractional coverage heatmap with oil blob at 9 different time steps for 50x50 2-D network with NWP characterization from Figure 9

In Figure 14 a mass balance for the simulation at all time steps is presented. There are three different states that injected particles can be in: solute, adsorbed, and eluted. Since the total injected particles always

equals the sum of these three states the mass balance is verified. In Figure 12 and Figure 14 it is seen that there are several particles left in the solute when the simulation is finished. These particles left over are mostly in water-saturated pores that are outside of the main flow field. If the simulation were to continue running these particles would either continue to diffuse until they end up attached to a water-NWP interface or eluted from the network. When comparing the solute + eluted line to the solute + eluted + particles line only ~1% of injected particles end up being adsorbed.

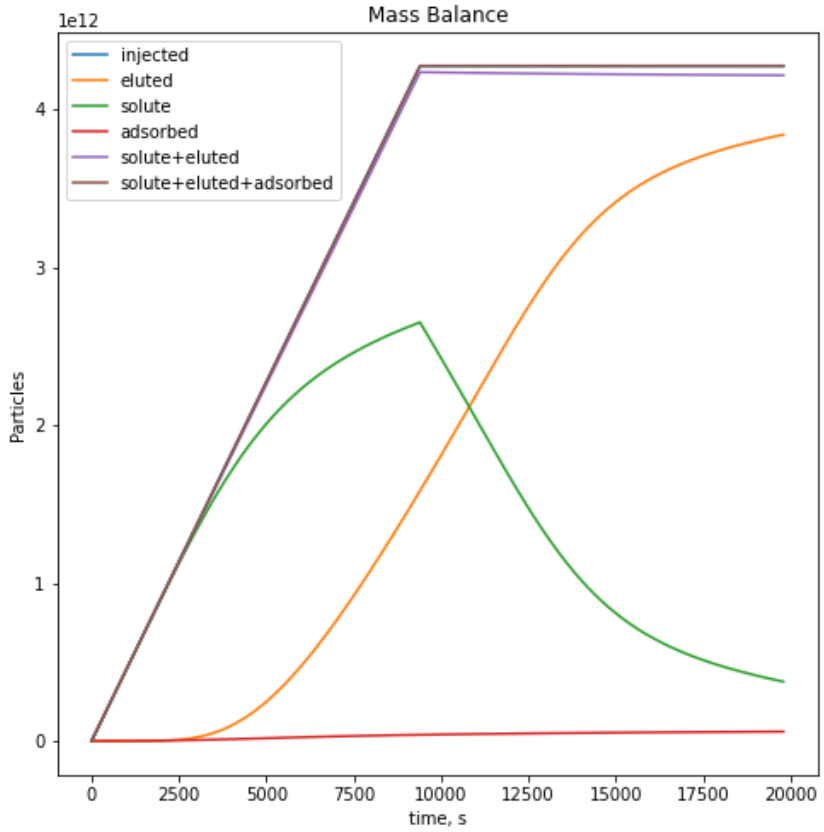


Figure 14: Mass balance of the 50x50 2-D network

In Figure 15 the fractional coverage θ of specific pores shown in Figure 16 is presented. Pore B is right next to a primary flow path and is close to the inlet, and thus reaches complete jamming ($\theta \rightarrow \theta_{max}$) quickly. Pores A and C take longer to approach complete jamming because the NPs must reach these pores through diffusion, as opposed to advection with pore B. Pore E barely experiences any adsorption since it is far away from any main flow paths, thus any Ns that would come in contact with this pore must diffuse

quite a distance to reach it. In Figure 12 there is still a sizable concentration of particles in the solute phase near this pore, so this pore would likely continue to experience dispersion if the simulation were to continue to run.

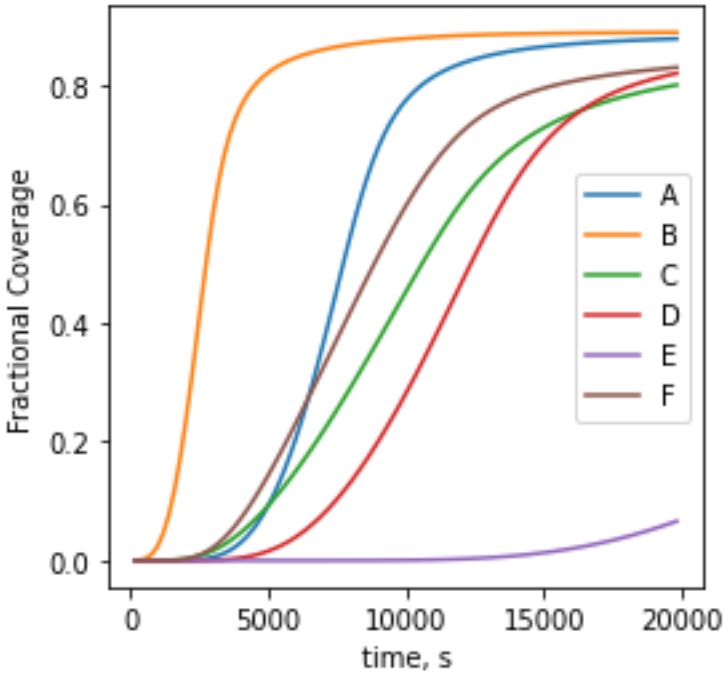


Figure 15: Fractional coverage, θ, of all oil pores as a function of time

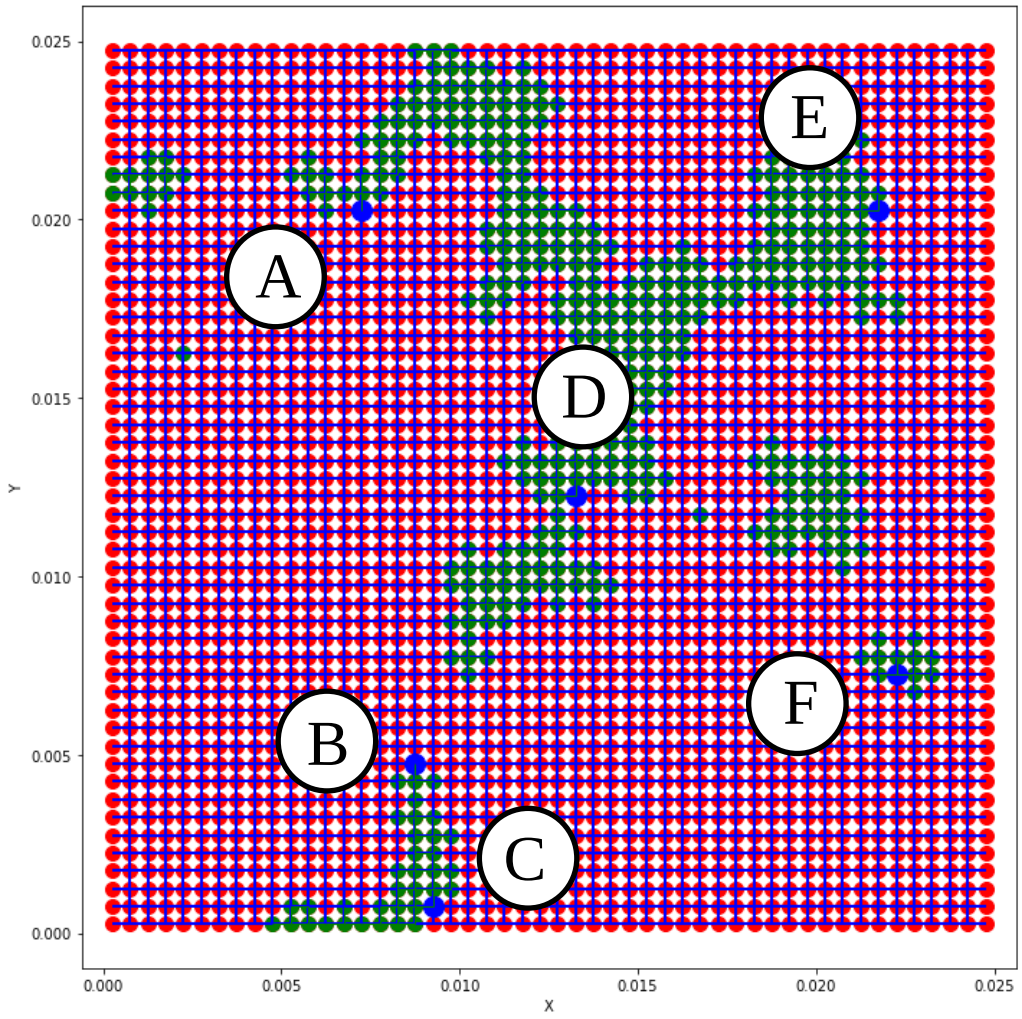


Figure 16: Pores represented in Figure 15

Chapter 4 Nanoparticle Effects on Longitudinal Dispersion

Typical groundwater velocities can range from as high as 1 foot per day ($3.5x10^{-6}\,m/s$) to as low as 1 foot per year ($9.7x10^{-9}\,m/s$) [55] or even as low as 1 foot per decade in certain cases. In the experiments motivating the present investigation [21], the interstitial velocities ranged from $1.7x10^{-4}\,m/s$ to $5.2x10^{-4}\,m/s$. Other experiments analyzing NPs transport in the typical velocity ranges expected in groundwater are lacking. Since it is possible to set the velocity to any desired value in the model developed in this study, the Darcy velocities used in this study range from $1.0x10^{-3}\,m/s$ to $1.0x10^{-9}\,m/s$, which will encompass velocities ranging from the upper range of lab-scale velocities to the lower range of velocities expected in groundwater. In this manner, a number of hypotheses made in [21] to explain experimental observations can be tested under realistic conditions.

4.1 Verification and Discussion of Extracted Pore Networks

The extracted pore network is necessarily a geometric simplification of the sphere packing. For example, pores may be represented by truncated pyramids or spheres, and throats by cuboids or cylinders. In Section 3.2 the derivation of hydraulic and diffusive conductance was presented, but in this section the use of these shapes is validated by comparing several measurable parameters to known correlations. The parameters used to verify the pore network in this study are the permeability, the tortuosity, and the breakthrough capillary pressure. The permeability of the extracted network is determined by simulating Stokes flow to find the flow rate for a given pressure gradient across the pore network, and then invoking Darcy's law in Eqn. (2.2. This permeability is then compared to the permeability of the sphere packing which is predicted with good accuracy by the Kozeny-Carman equation [40].

$$k_{KC} = \frac{\varepsilon^3 {d_b}^2}{180(1-\varepsilon)^2} \qquad \textbf{(4.1)}$$

The permeability of the extracted pore network is 4071.83 D, whereas the permeability of the sphere packing from which the network was extracted is 3598.32 D, a difference of 13.16%.

The tortuosity of the extracted pore network is 1.43. Several models exist to predict the tortuosity of porous media [56] that give values ranging from $\tau =$ 1.3 to 1.7 for a packed column with a porosity of $\varepsilon =$ 0.39. One such model appropriate for granular media [56] is

$$\tau = \frac{\varepsilon}{1 - (1 - \varepsilon)^{2/3}} \quad \textbf{(4.2)}$$

which yields a tortuosity of $\tau = 1.39$.

Additional verification consists of comparing the breakthrough capillary pressure for drainage in the extracted network to the breakthrough capillary pressure expected for a packing of uniform spheres. In [57] it was found that for a column packed with spherical beads the breakthrough pressure during drainage of water by air, p_c°, can be reliably predicted by the following equation

$$p_c^\circ = \frac{2\gamma_{air-water}}{\langle R_t \rangle} \quad \textbf{(4.3)}$$

where the air-water surface tension, $\gamma_{air-water} = 0.072\ N/m$, and $\langle R_t \rangle$ is a characteristic throat radius given by $\langle R_t \rangle = 0.21 d_b$ [58]. For a bead size of $d_b = 2\ mm$, a characteristic throat radius of $\langle R_t \rangle = 0.42\ mm$ is predicted. A simulation of drainage of water by air was preformed using OpenPNM's OrdinaryPercolation algorithm and is presented in Figure 17 as a relationship of non-wetting phase (air) saturation to capillary pressure. The condition of breakthrough of the non-wetting phase is identified with the inflection point in this graph. The value predicted from Eqn. (4.3 is clearly in very good agreement with this expectation, a further verification of the validity of the extracted pore network. Furthermore, a histogram of the throat diameters is shown in Figure 18, and the characteristic throat radius ($\langle R_t \rangle = 0.42\ mm$) is shown with the green line, which is larger than 95% of throats in the network.

Given the close agreement of permeability, tortuosity, and breakthrough capillary pressure as shown in Table 3, it may be concluded that the extracted pore network pores and throats represented by cuboids and truncated pyramids, respectively, is an acceptable approximation of the void space in the simulated sphere packing.

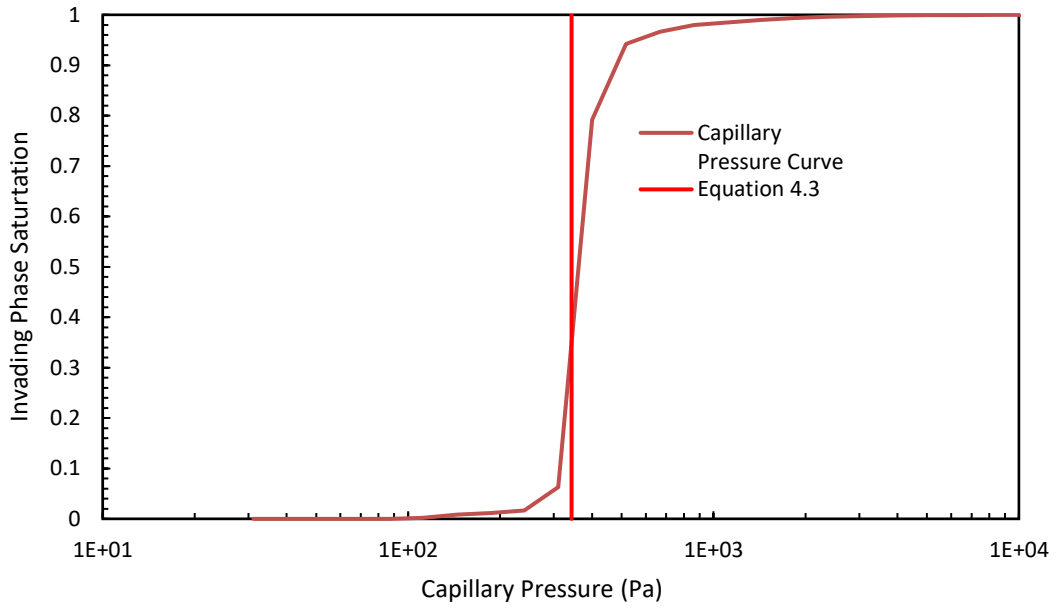


Figure 17: Capillary pressure curve found using OpenPNM's OrdinaryPercolation algorithm.

Table 3: Verification Data for extracted pore networks with L_c = 15cm, D_c= 2.5 cm, and D_b = 2mm

	Column #1	Column #2
Bead Count	10916	10918
ε	0.390	0.390
k_S *(D)*	3598	3607
k_{KC} *(D)*	4072	4173
τ	1.43	1.42

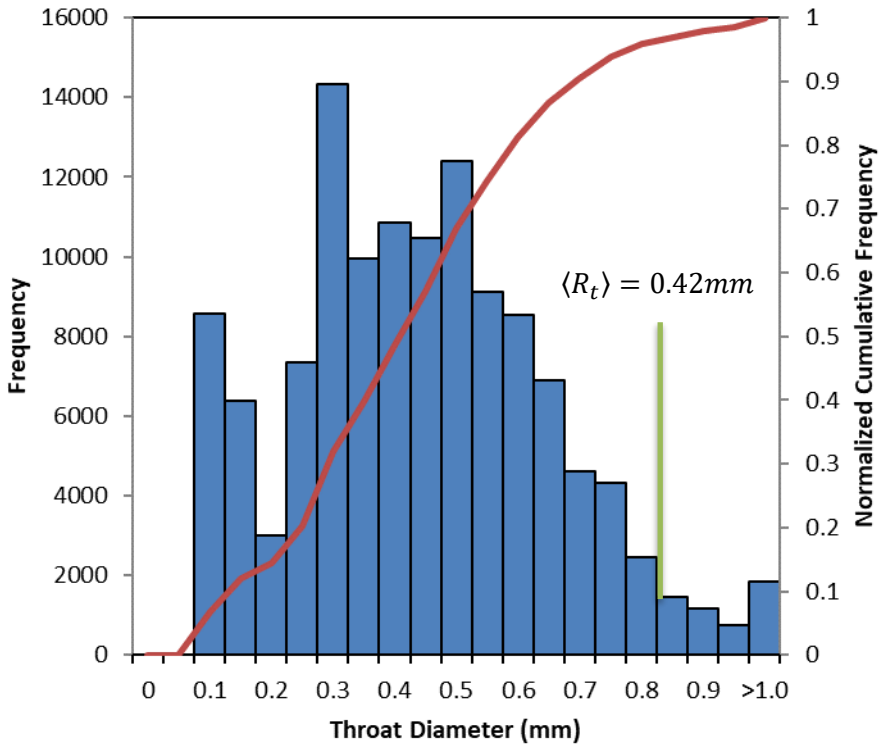


Figure 18: Histogram of throat diameters for Column #1 from Table 3 with characteristic throat radius denoted with a green line.

Table 4 lists the permeability values and bead diameters of 6 columns that are based on the voxel image shown in Figure 5(a). Typical permeabilities for unconsolidated deposits can range from $1x10^{5}D$ for gravel to $1x10^{-8}D$ for unweathered clay and shale [42]. Since columns packed with spherical beads with uniform size are the only porous media modeled in this study a smaller range of permeability is used, as seen in Table 4. The simulated permeability, k_s, is found using Darcy's law in Eqn. (2.2) All six columns presented in Table 4 are pore networks extracted from the same voxel image based on the column from Figure 5.

Table 4: Pore network models at different scales extracted from column #1 from Table 3.

	Ratio to Column #1	d_b (mm)	k_s (D)
Column #1	1	2	4072
Column #3	0.1	0.2	40.72
Column #4	0.04	0.08	6.51
Column #5	0.02	0.04	1.63
Column #6	0.01	0.02	0.407
Column #7	0.002	0.004	0.0163

4.2 Step change and pulse injection comparison

For the simulations run in this experiment D_L and U_{Fitted} are determined by fitting the analytical solution to the simulated elution curve using Eqn. (3.21 for step change simulations and Eqn. (3.23 for pulse injections. Figure 19 presents a typical elution curve for a step change injection for a column with L = 15cm, d_c = 2.5cm, and d_b = 2mm. The size of injected particles is d_p = 1000nm and the Darcy velocity was set to $U_{Darcy} = 10^{-5}$m/s. Figure 20 presents an elution curve using the same parameters as Figure 19 but with a pulse injection instead of a step change injection. The mass of particles injected was set to 20 μg and with a density of $\rho_p = 1050 g/L$. The resulting longitudinal dispersion coefficients for the simulations are $D_L = 5.29\ x\ 10^{-8} m^2/s$ for the step change simulation and $D_L = 5.52\ x\ 10^{-8} m^2/s$ for the pulse injection, and the resulting fitted interstitial velocities are $U_{fitted} = 2.54\ x\ 10^{-5} m/s$ for the step change simulation and , $U_{fitted} = 2.61\ x\ 10^{-5} m/s$ for the pulse. The calculated interstitial velocity $U = U_{Darcy}/\varepsilon$ is $2.53\ x\ 10^{-5} m/s$. At a particle to bead ratio of 0.0005 there is almost no effect from both the VPE and HD effects. Thus, the difference between the two fitted interstitial velocities and the calculated velocity is attributed to error in fitting the analytical solution to the simulated elution curve. Since the analytical model for a step change elution curve results in an error of 0.52% for U_{fitted} compared to 3.45% for the injected pulse, future experiments will be done using step-change simulation instead of with pulse injections. An added benefit of using step simulations is that it removes the need to define the injected mass, and elution curves can be normalized with the injected concentration.

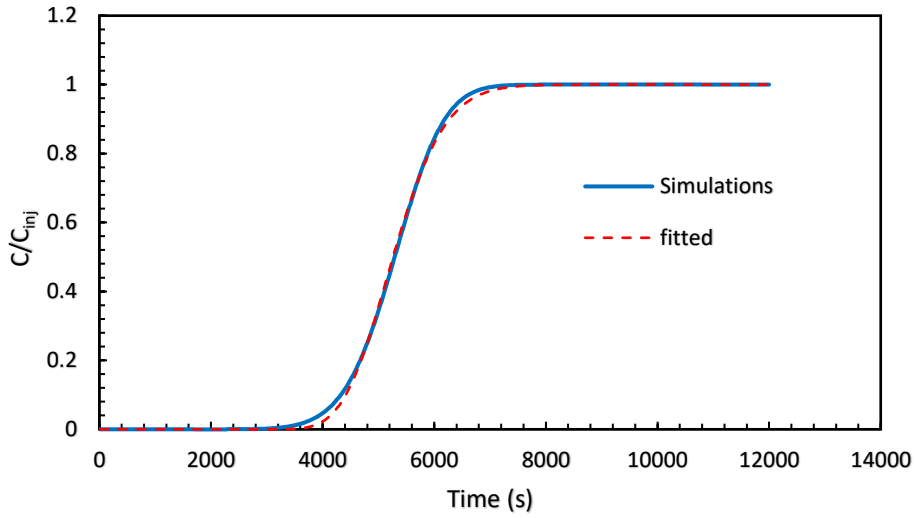


Figure 19: Elution curves for $d_p = 1000$nm particles with a Darcy velocity of $10^{-5}\ m/s$ with a bead size of 2.0mm represented with the blue curve for step change injection ($D_L = 5.29\ x\ 10^{-8} m^2/s, U_{fitted} = 2.54\ x\ 10^{-5} m/s$)

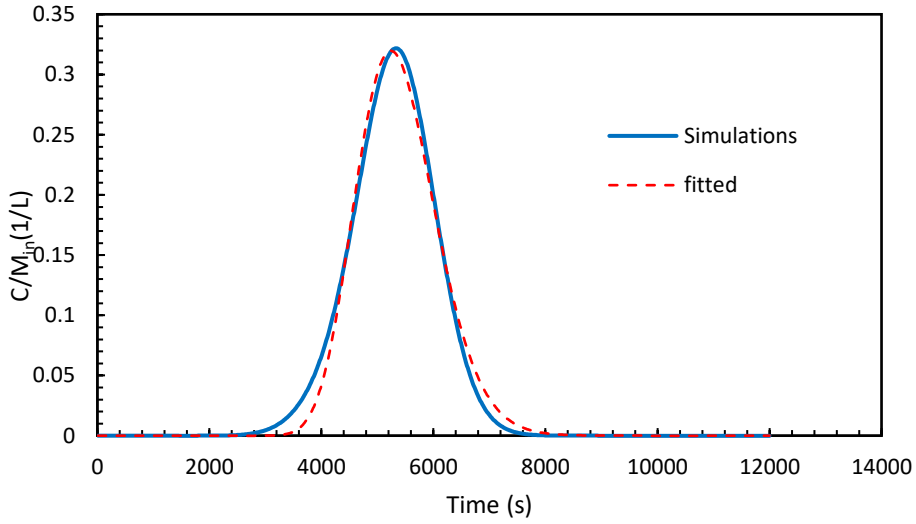


Figure 20: Elution curves for $d_p = 1000$nm particles with a Darcy velocity of $10^{-5}\ m/s$ with a bead size of $d_b = 2.0$ mm represented with the blue curve for pulse injection ($D_L = 5.52\ x\ 10^{-8}\ m^2/s, U_{fitted} = 2.61\ x\ 10^{-5}\ m/s$).

4.3 Comparison to experiments

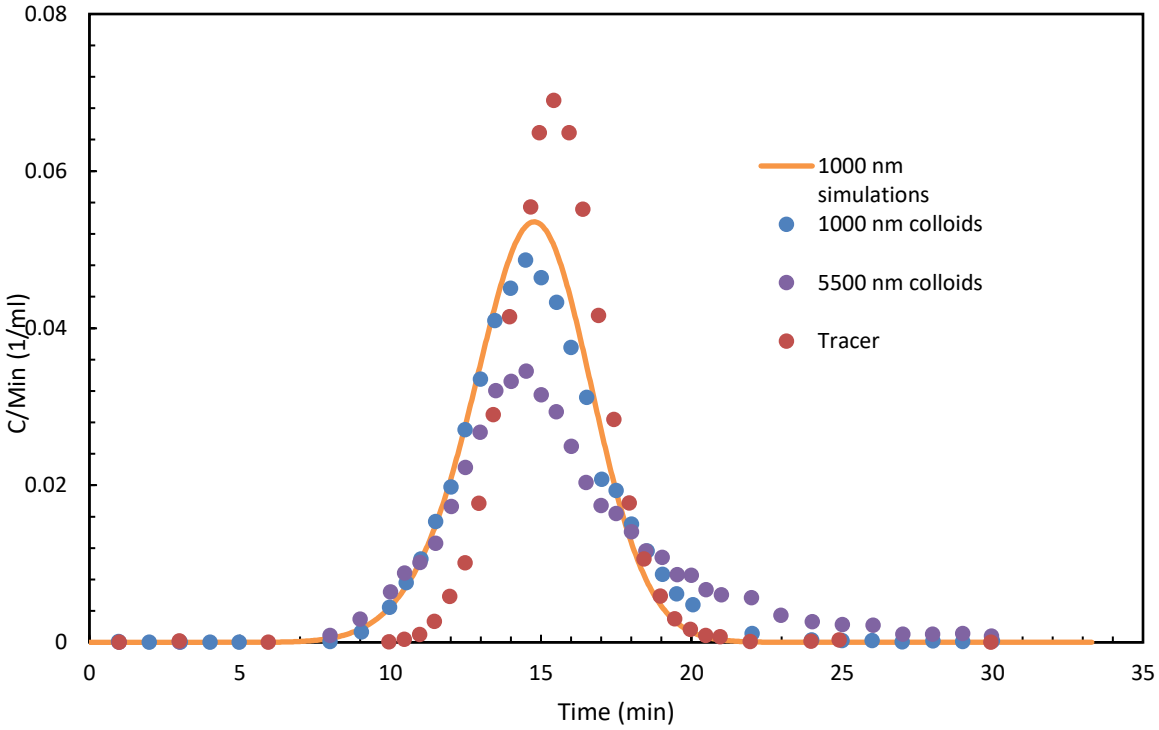


Figure 21: Comparison of breakthrough results from [21] experiment numbers 4 (orange), 37 (blue), and 52 (yellow) against a simulated 30cm packed column with 1000 nm particles.

Experimental data for breakthrough concentration C normalized with injected mass M_{in} versus time from Chrysikopoulos and Katzourakis (2015) [21] and data from a simulation is presented in Figure 21. The experimental data is all preformed with packed columns with $L_c = 30\ cm$, $d_c = 2.5\ cm$, $d_b = 2\ mm$, a flow rate of 4 mL/min, and the porosity of the columns range from ε = 0.39 to 0.41. The three breakthrough curves presented are for 1000nm particles, 5500 nm particles, and bromide tracer. In the cases in this paper where a solvent is used instead of NPs a diffusivity value of $D_{AB} = 1.85\ x\ 10^{-9}$ is assumed, which is the diffusivity for bromide that is found using the Wilke-Chang relationship [59]. The resulting longitudinal dispersion coefficients are $D_L = 1.18\ x\ 10^{-5},\ 1.97\ x\ 10^{-5},$ and $4.83\ x\ 10^{-5}\ m^2/s$ respectively.

The column used in these experiments was reproduced as a voxel image using Porespy's "pseudo_gravity_packer" algorithm and extracted using the SNOW algorithm. The radius of one bead in this column is 20 voxels, as opposed to 28 voxels in the 15 cm long columns, thus the resolution in the voxel image is lower than the image presented in Figure 5. It was necessary to use smaller voxel sizes due to file size limitations resulting from the longer column. The lower resolution of the image resulted in a smaller porosity and may influence the simulated longitudinal dispersion. The resulting porosity for this network is $\varepsilon = 0.372$, and the measured permeability and tortuosity found using Eqns. (2.2) and (2.4 are $k_s = 3509.09D$ and $\tau = 1.494$ respectively, and the predicted values found using Eqns. (4.1 and (4.2 are $k_c = 2933.7D$ and $\tau = 1.39$. Three simulations were performed on this generated column in OpenPNM with the same two particle sizes as was used in [21] and a tracer with a diffusion coefficient of 1.85×10^{-9} m^2/s. All three D_L were found to be between $9.38 \times 10^{-7}\ m^2/s$ and $9.39 \times 10^{-7}\ m^2/s$, and U_{fitted} was found to be between 3.42×10^{-4} m/s and 3.43×10^{-4} m/s. Clearly, when the model used in this study is run with the same parameters as was in the experiment done by [21], changes in particle size has no effect on either D_L or U_{fitted}. Since both HD and VPE effects are based on the ratio of particle size to the throat diameter it is logical that changes in dispersion will only occur when the ratio d_p/d_b is large enough. The largest particles used in [21] are 5500 *nm*, which only gives a particle to bead ratio of $d_p/d_b = 0.00275$.

In [21], it is observed that there is early breakthrough for larger particles compared to smaller particles or to a molecular tracer, and this observation is attributed to exclusion from lower velocity regions. However, this attribution is not supported by the simulations presented in this thesis. Specifically, the ratio of particle to bead size is much too small for there to be any noticeable effect, as is observed in the simulations used in this study, and the observed differences in dispersion for the different particle sizes are possibly due to different phenomena, such as a combination of reversible attachment and size exclusion. Size exclusion [60, 61] where larger particles can avoid narrower throats and thus have a more "streamlined" path was also suggested as a possible explanation for the earlier breakthrough for larger nanoparticles. It is unlikely that streamlines due to size exclusion played any effect in [21] because the largest particle to bead size ratio used is $d_p/d_b = 0.00275$, while [60] suggests streamlining only occurs when the ratio of the smallest throat size to particle size is 1.5. There have been several studies about reversible attachment of NPs in porous media [62] but the correlation between retardation and dispersion has rarely been studied.

4.4 Influence of Hindered Diffusion and Velocity Profile Exclusion Effects

The effect that including HD and VPE effects have on both the longitudinal dispersion coefficient D_L and effective particle velocity U_{fitted} is analyzed in this section. This is done by running dispersion simulations with and without the changes to the pore and throat diffusivities and to the throat hydraulic conductance. Figure 22 and Figure 24 present the change in longitudinal dispersion coefficient as a function of the dimensionless Peclet number

$$Pe_m \equiv \frac{U_{fitted} d_b}{D_{eff}} \qquad \textbf{(4.4)}$$

for simulations run in column #7 ($d_b = 0.004$mm) and #4 ($d_b = 0.02$mm) respectively from Table 4. The percent change in longitudinal dispersion represent simulations run with the HD effect, the VPE effect, and both effects compared to simulations done with none of the effects. Similarly, Figure 23 and Figure 25 presents data from the same simulations as Figure 22 and Figure 24, but the percent change in U_{fitted} is presented instead of D_L as a function of the Peclet number. Table 5 presents the NP sizes and the range of Darcy velocities used in these simulations.

When considering U_{fitted}, it was observed that the fitted value positively deviates from the inputted value when $Pe_m < 0.1$, and this deviation increases at the same rate as the Peclet number decreases. Thus, U_{fitted} values should not be considered to hold much meaning when at low Pe_m. Despite this, the values for D_L appear to be completely reasonable when found at these low Pe_m values.

When $Pe_m \gg 1.0$, there is no change in either D_L or U_{fitted} when HD is the only effect used. It can thus be concluded that the HD effect will only be significant in the diffusive transport regime. The effect that including HD has on dispersion can be seen as Pe_m decreases and approaches 10.0, wherein there is a slight jump just before the superposition regime ($0.5 \leq Pe_m \leq 5.0$). When comparing dispersion to the Peclet number, it is typical to see the maximum dispersive value occur in the superposition regime [63], thus this slight increase in dispersion caused by the HD effect can be attributed to the same effect in overall dispersion. In the dispersive regime ($Pe_m < 0.1$). After this jump, the percent change decreases until a certain point. Logically, all these effects can be seen to be greater as d_p/d_b increases. When only considering the HD effect, there is no effect on the percent change in U_{fitted}.

When the VPE effect is used an increase in both D_L and U_{fitted} is observed as particle size increases when $Pe_m \gg 10.0$. The percent change is effectively the same no matter how much higher Pe_m is. Thus, when there is a higher effective particle velocity that arises from the VPE effect, there will be an increase in D_L correspondingly. In the dispersive regime, the VPE effect results in a reduced D_L. Considering Eqn. (2.6, increasing the velocity will increase the mechanical dispersion, resulting in a higher D_L, thus the conclusion that the VPE effect results in enhanced dispersion is reasonable. In Figure 22 the change in dispersion from the VPE effect is at the lowest point at $Pe_m = 0.2$, and at lower values for Pe_m there is still a reduction in D_L, but this effect is decreasingly smaller.

When combining both the HD and VPE effect, only the VPE effect plays a significant role when $Pe_m > 10.0$. Below this point the two effects tend to enhance one another with regards to the percent change in D_L and U_{fitted}.

Table 5: Particle sizes and range of Darcy velocities used in simulations presented in Figure 22,Figure 23, Figure 24, and Figure 25

	d_p (nm)	d_p/d_b	$U_{Darcy,min}$ (m/s)	$U_{Darcy,max}$ (m/s)
Column #7	20	0.005	10^{-9}	10^{-3}
Column #7	50	0.0125	10^{-9}	10^{-3}
Column #7	100	0.025	10^{-9}	10^{-3}
Column #4	100	0.005	10^{-8}	10^{-3}
Column #4	200	0.01	10^{-8}	10^{-3}
Column #4	500	0.025	10^{-8}	10^{-3}

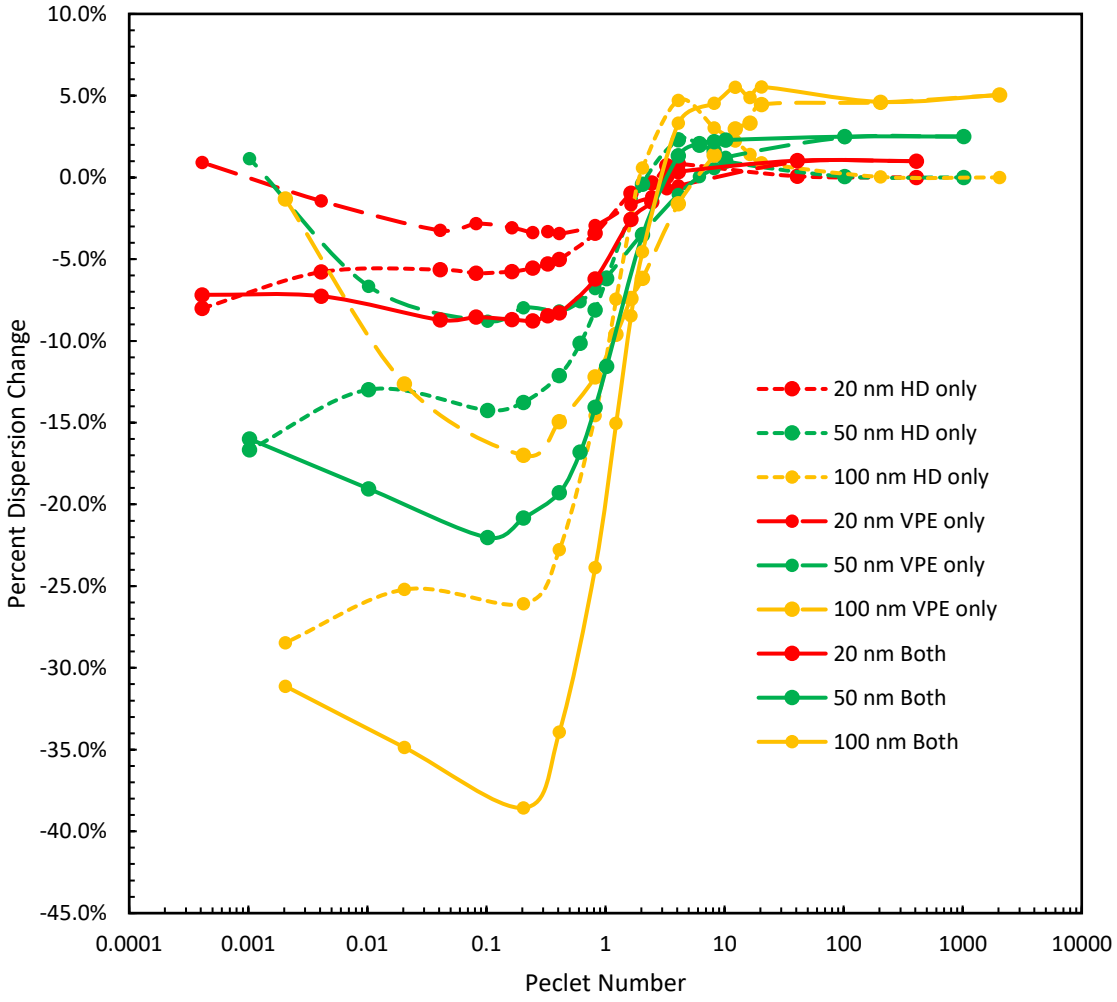


Figure 22: Percent change in D_L for simulations run with specified particle effects compared to simulations with no effects as a function of the Peclet number run in column #7 with $d_b = 0.004$ mm beads, for 20nm (red), 50nm (green), and 100nm (yellow). Small dashed lines are for when HD was included as an effect, large dashed lines are for when VPE is included as an effect, and solid lines are when both HD and VPE are included.

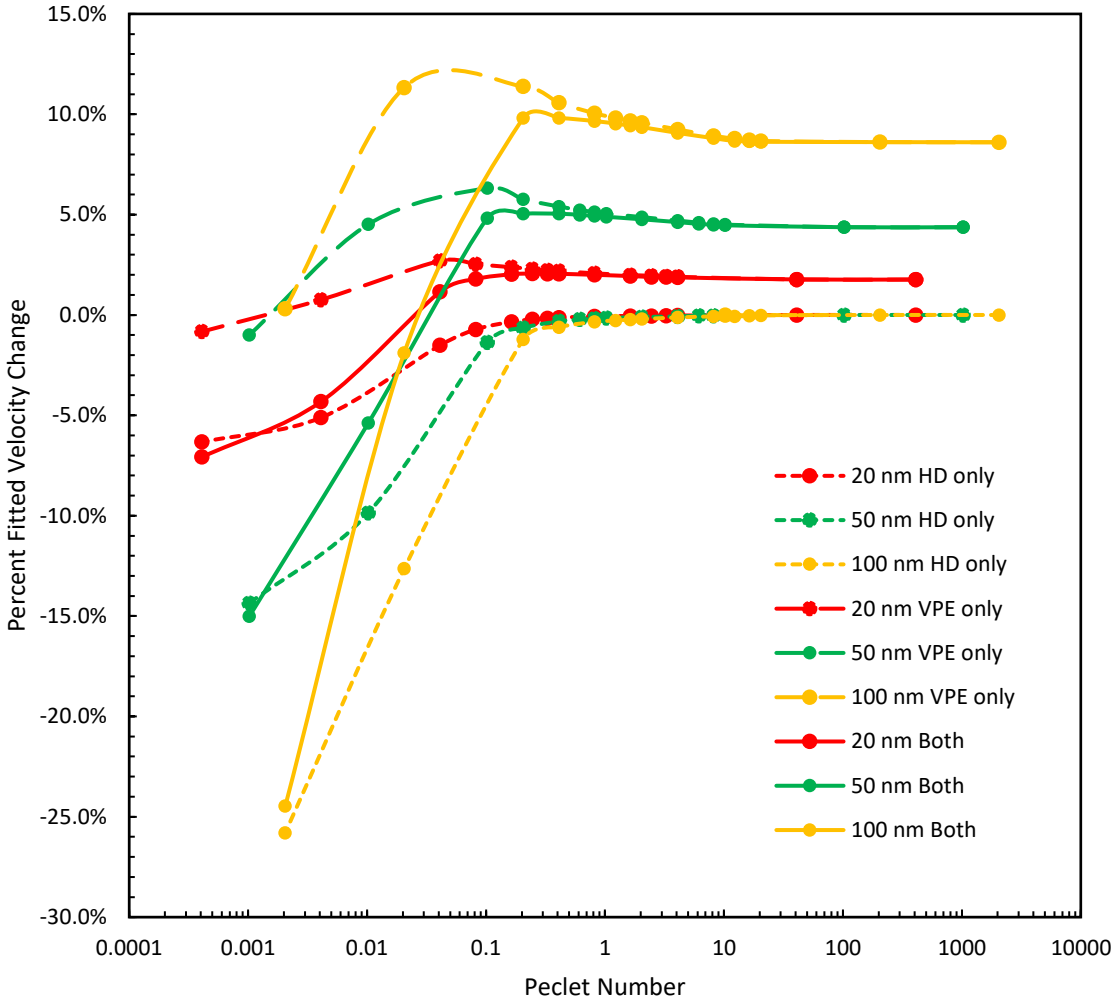


Figure 23: Percent change in U_{fitted} for simulations run with specified particle effects compared to simulations with no effects as a function of the Peclet number run in column #7 with $d_b = 0.004$ mm beads, for 20nm (red), 50nm (green), and 100nm (yellow). Small dashed lines are for when HD was included as an effect, large dashed lines are for when VPE is included as an effect, and solid lines are when both HD and VPE are included.

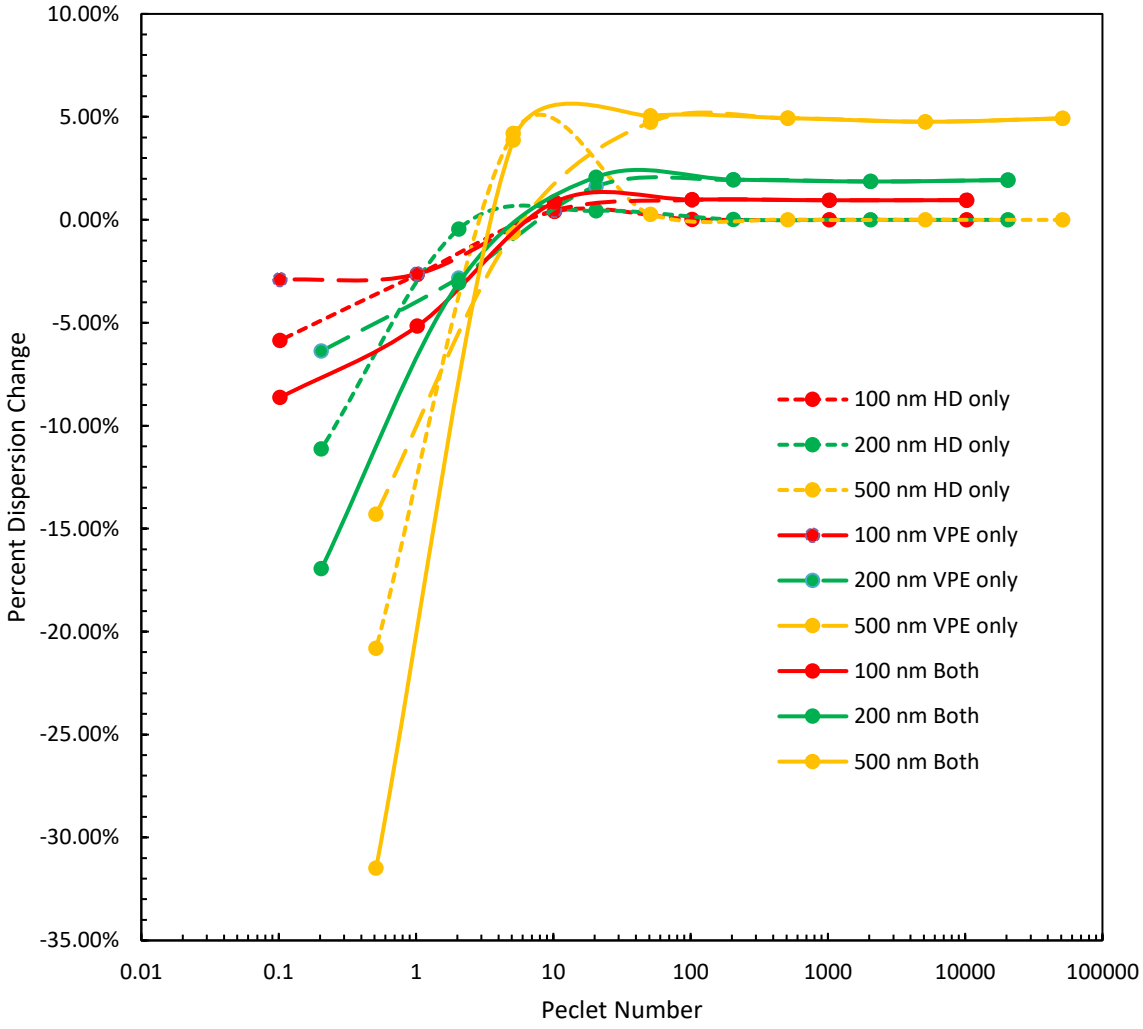


Figure 24: Percent change in D_L for simulations run with specified particle effects compared to simulations with no effects as a function of the Peclet number run in column #4 with $d_b = 0.02$ mm beads, for 20nm (red), 50nm (green), and 100nm (yellow). Small dashed lines are for when HD was included as an effect, large dashed lines are for when VPE is included as an effect, and solid lines are when both HD and VPE are included.

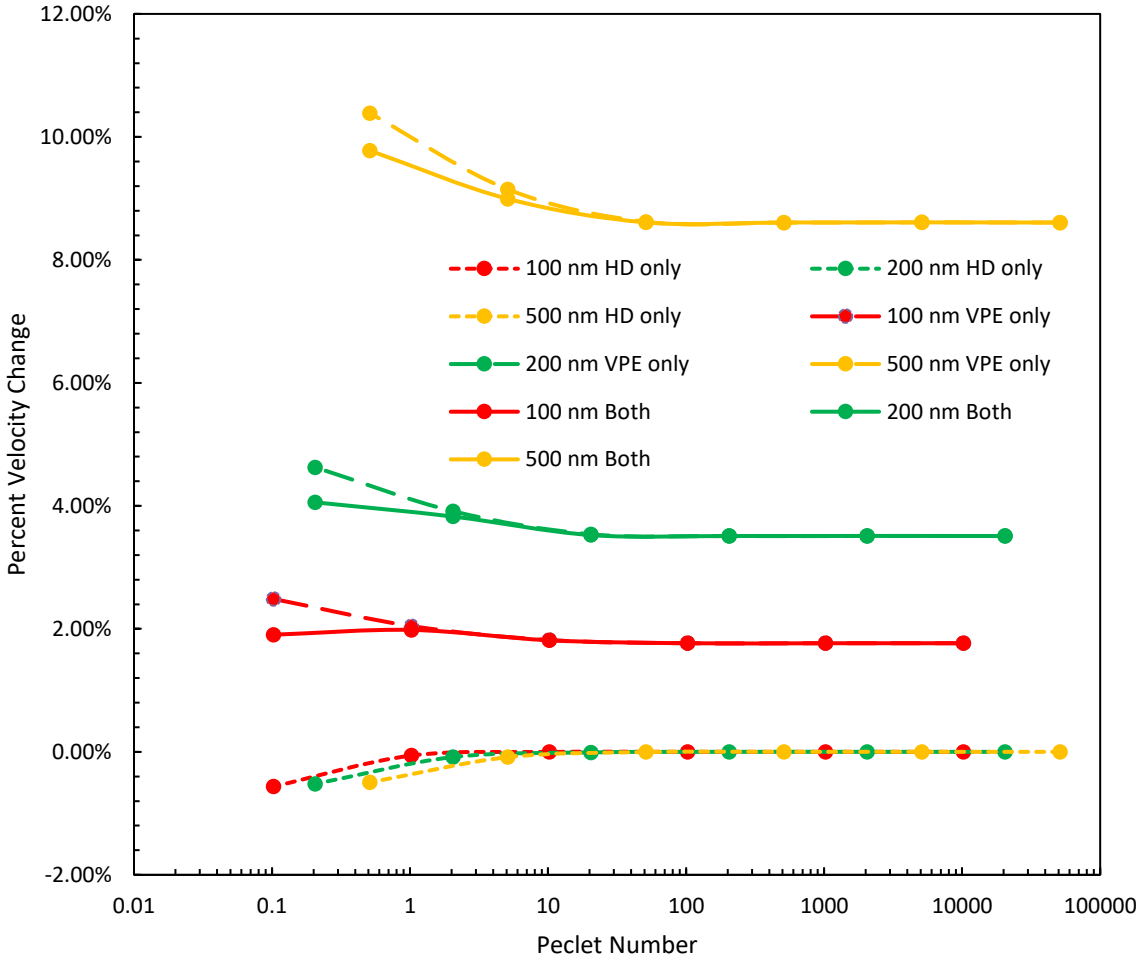


Figure 25: Percent change in U_{fitted} for simulations run with specified particle effects compared to simulations with no effects as a function of the Peclet number run in column #4 with $d_b = 0.02$ mm beads, for 20nm (red), 50nm (green), and 100nm (yellow). Small dashed lines are for when HD was included as an effect, large dashed lines are for when VPE is included as an effect, and solid lines are when both HD and VPE are included.

Early breakthrough of colloidal particles due to enhanced velocity is a widely observed phenomena that has been attributed to reduced porosity resulting from void volume that is not accessible to the particles, as well as exclusion from lower velocity regions within the homogeneous porous media. Figure 26 presents the percent increase in particle (fitted) velocity as a function of the ratio of NP size to bead size d_p/d_b for both columns #3 and #4 from Table 4 with $d_b = 0.2$ *mm* and 0.08 *mm* respectively. Both columns result in the same linear plot is a solution for effective velocity from the regression line in Figure 26. The line does not intercept at zero, which is attributed to the slight error when fitting the analytical solution to the simulation elution curve.

$$U_{eff} = U\left(3.4\frac{d_p}{d_b} + 0.9922\right) \qquad \textbf{(4.5)}$$

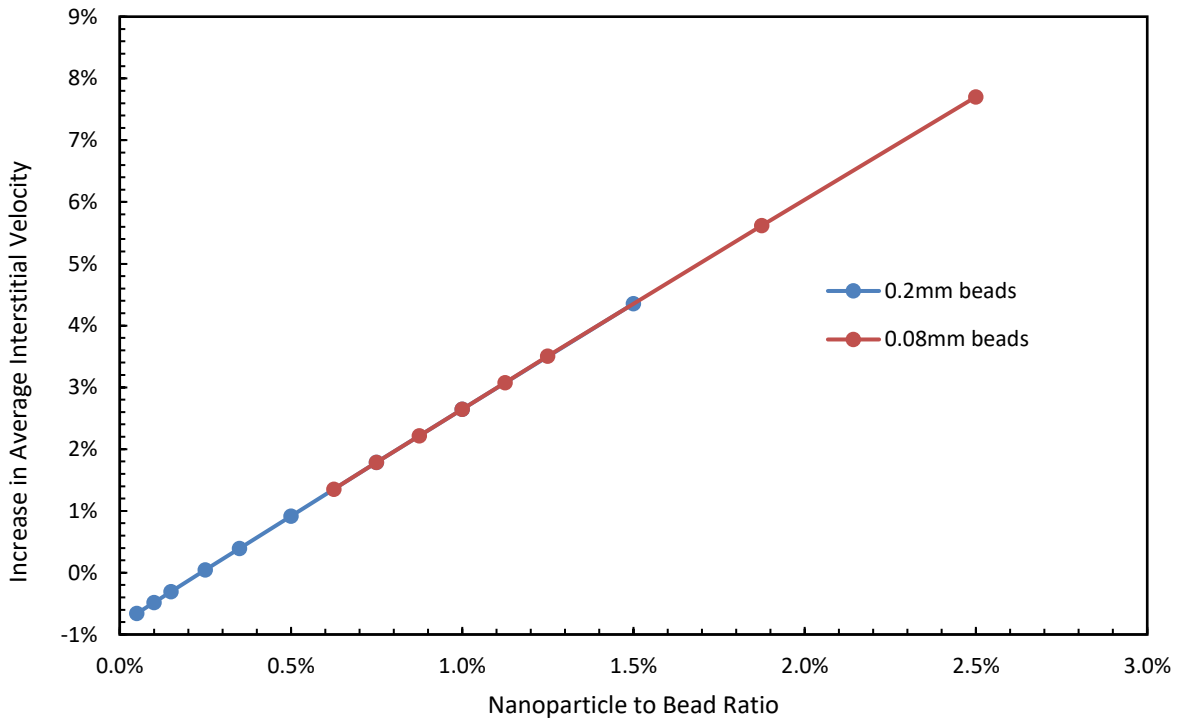


Figure 26: Percent increase in effective velocity as a function of increase in NP to bead ratio d_p/d_b for both columns #3 and #4 with a Darcy velocity of $U_{Darcy} = 0.001\ m/s$ for both columns.

4.5 Longitudinal Dispersion Correlations

Values for longitudinal dispersion can be accurately predicted using various equations corresponding to different dispersive regimes. For a pure molecular diffusion regime, the D_L has been shown to be predicted with Eqn. (4.6, while for dispersion with major mechanical dispersion, D_L can be predicted using Eqn. (4.7 [63].

$$\frac{D_L}{D_m} = \frac{1}{\tau}, \qquad \text{valid for } Pe_m < 0.3 \tag{4.6}$$

$$\frac{D_L}{D_m} = \frac{Pe_m}{6\tau}\left[ln\left(\frac{3}{2}Pe_m\right) - \frac{1}{4}\right], \qquad \text{valid for } \frac{Pe_m}{\tau} \gg 1 \tag{4.7}$$

Figure 27 presents data from columns #3 and #7 from Table 4, which were scaled to have 0.2 *mm* and 0.004 *mm* beads respectively for a wide range of NP sizes and additionally with a molecular solute. In section 4.4 it was shown that including HD and VPE effects can have a moderate effect on both D_L and U_{fitted}. However, in Figure 27 all particle sizes follow the same trend regardless of the simulation's velocity or size of the network. When compared to the correlations in Eqns. (4.6) and(4.7, they can predict the dispersion with high accuracy in the diffusive regime and reasonable accuracy in the regime with major mechanical dispersion.

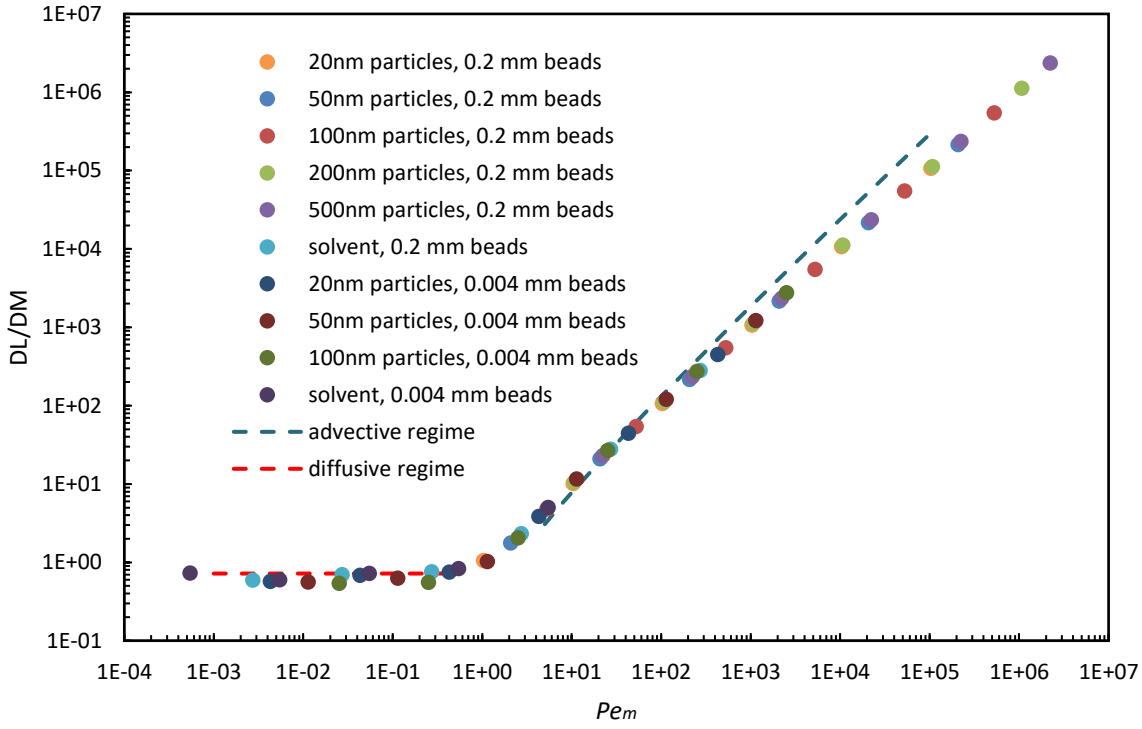


Figure 27: D_L/D_m versus Pe_m for different particle sizes in packed columns with $d_b = 0.2$mm and $d_b = 0.004\ mm$, along with predictive Eqn. (4.6 for the diffusive regime (Pe_m < 0.3) and Eqn. (4.7 advective regime ($Pe_m \gg 1$). All particle sizes had simulations run with the velocity being in range from $U_{Darcy} = 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\ m/s$.

Chapter 5 Irreversible Adsorption of Nanoparticles onto Fluid Interfaces

5.1 Characterization of non-wetting phase in packed column

In this section, a non-wetting phase (NWP) is added to previously generated pore networks, and the reproducibility and characteristics of these partially-saturated networks are analyzed. In Section 3.6.1 the total area of so-called surface NWP pores, $A_{NWP,surf}$, was defined. This term is only an approximation of the actual fluid-NWP interface area, A_o, in an actual packing of spheres with the same non-wetting phase saturation. The assumption that $A_{NWP,surf} = A_o$ is tested by comparing the simulated results to experiments. The validity of the networks is further tested by finding the tortuosity of the pore network as a function of the NWP saturation and comparing to existing correlations.

The same pore network extracted from column #1 as presented in Table 3 is also used in this section. A NWP is added to the pore network by executing an invasion-percolation (IP) algorithm using the process described in Section 3.6. We select 100 pores randomly as NWP inlets and set target saturations of $S_{NWP} = 0.01, 0.1, \text{and } 0.5$. Figure 28 shows pores saturated with water and NWP for the three selected S_{NWP} values. To assess the effect of multiple realizations (*i.e.*, different random choice of 100 "seed" pores in the IP algorithm) of the same value of NWP saturation, we carry out two additional IP simulations. Figure 29 presents three different realizations for $S_{NWP} = 0.1$.

The calculated NWP saturation, $S_{NWP,calc}$, effective porosity, ε_{eff}, water-NWP interface area, A_o, and specific surface area, a_o are listed in Table 6 for the three networks presented in Figure 28 and the two additional networks presented in Figure 29 (Figure 28.B and Figure 29.A depict the same pore network). The target saturation S_{NWP} is not exactly matched by the actual $S_{NWP,calc}$, which is calculated by finding the ratio of NWP volume to total pore volume. $S_{NWP,calc}$ is always somewhat greater than S_{NWP} because pores with water trapped during NWP invasion are converted to NWP-filled pores. The effective porosity, ε_{eff}, which is the porosity calculated using the volume of pores filled with the mobile aqueous phase to the bulk volume of the porous medium ($\varepsilon_{eff} = V_W/V_B$). The interfacial area, A_o, is calculated by summing the

surface area of all "surface NWP" pores in the network, and the specific interfacial area a_o is calculated by determining the ratio of interface area to the water volume V_W.

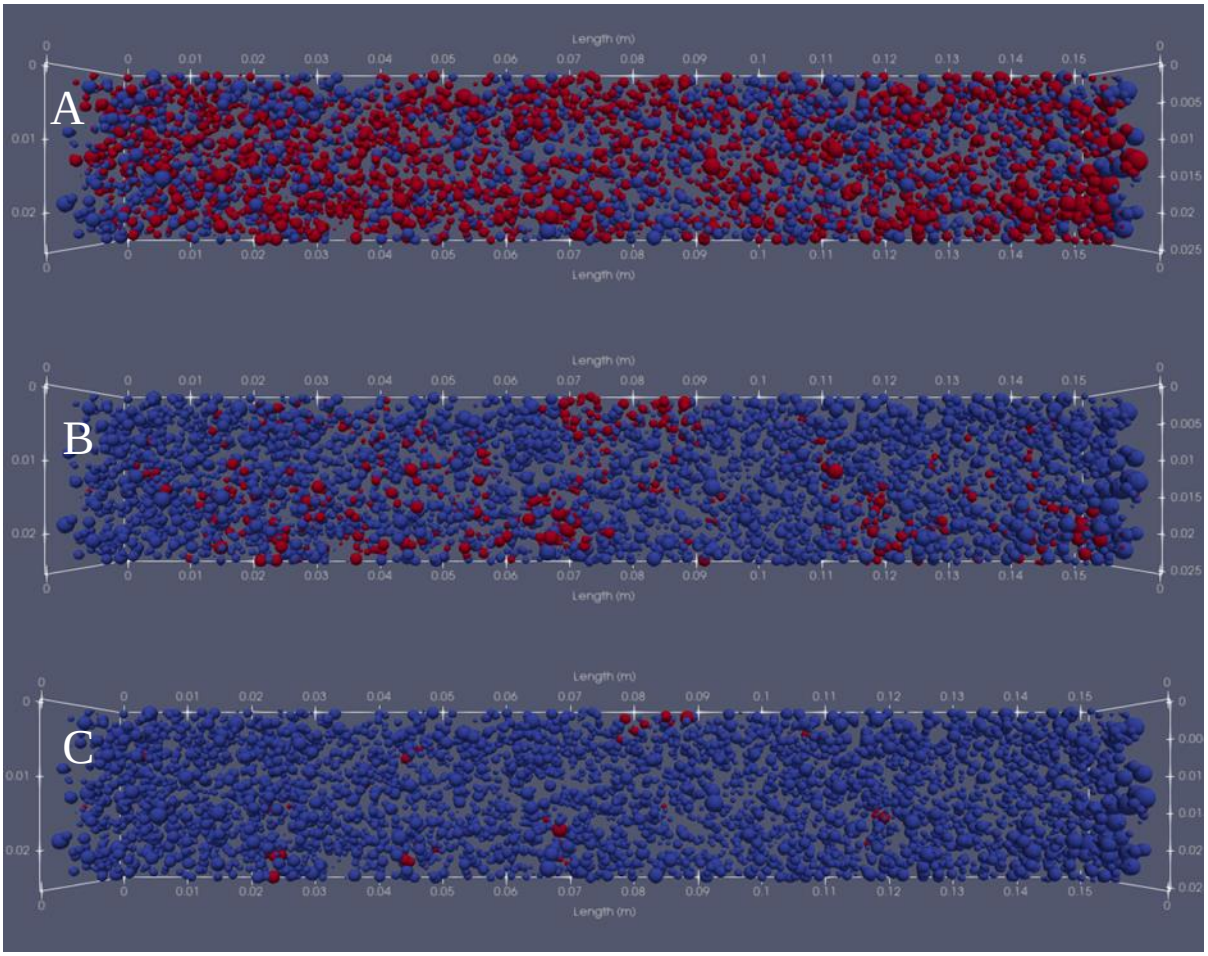


Figure 28: Pore network extracted from a 15-cm packed column with water phase shown in blue and NWP shown in red where $S_{NWP} = 0.5$ in (a), $S_{NWP} = 0.1$ in (b), and $S_{NWP} = 0.01$ in (c).

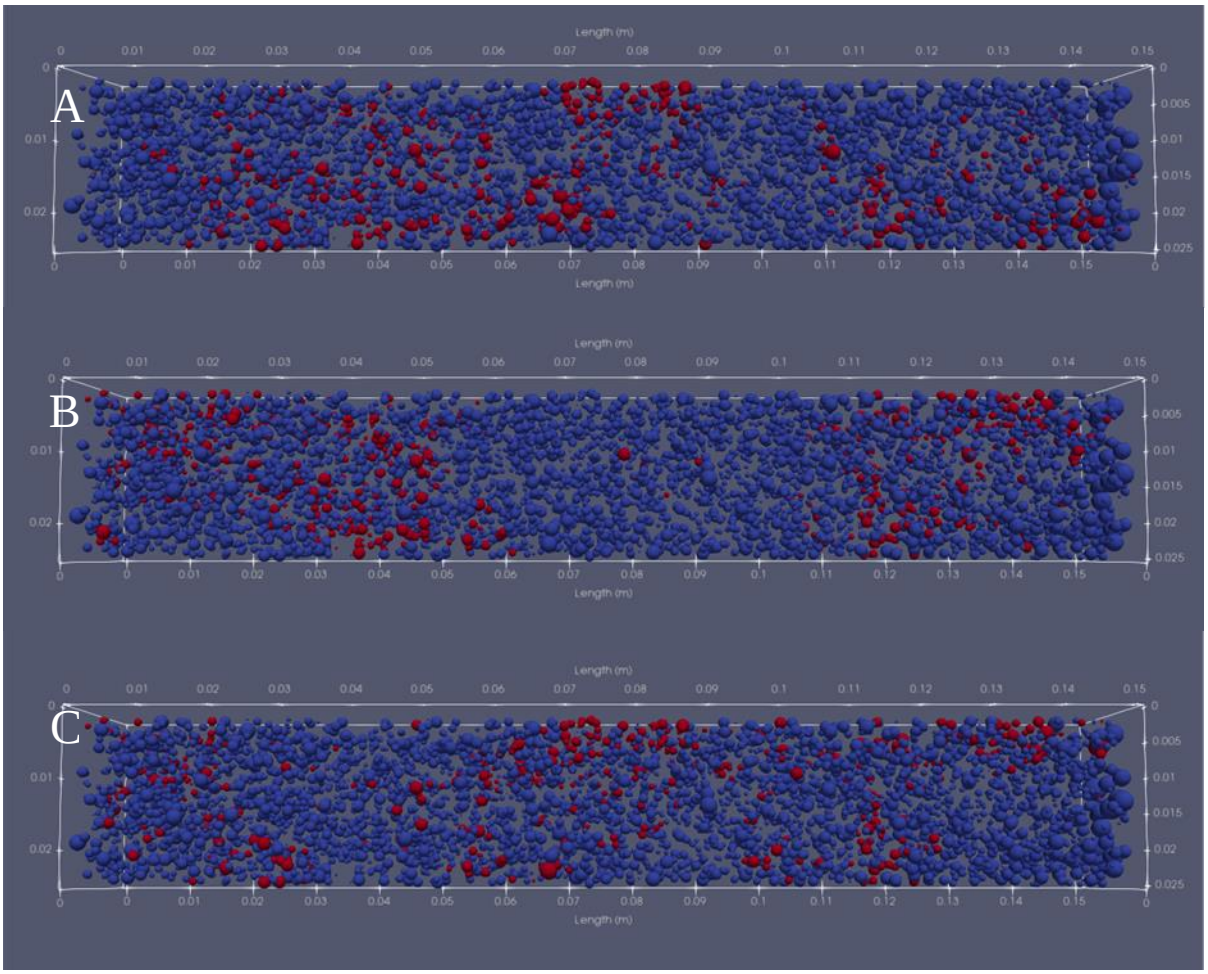


Figure 29: Pore network extracted from a 15-cm packed column with water phase shown in blue and oil phase shown in red. All three networks have an NWP saturation of $S_{NWP} = 0.1$, but with different randomly selected inlet pores.

Table 6: Saturation data for columns presented in Figure 28 and Figure 29

Pore Network	Figure 28.C	Figure 28.B	Figure 29.B	Figure 29.C	Figure 28.A
S_{NWP}	0.01	0.1	0.1	0.1	0.5
$S_{NWP,calc}$	0.013	0.115	0.115	0.115	0.588
ε_{eff}	0.391	0.350	0.350	0.350	0.163
$A_o\ (m^2)$	0.0018	0.0140	0.0139	0.0140	0.0561
$a_o(m^{-1})$	62.4	535.4	531.3	534.6	3850.6

A 3 PV slug injection was simulated on each of the networks presented in Figure 29, and the normalized elution curves are presented in Figure 30, where it is observed that the three elution curves are effectively the same. It is concluded that there is negligible variation between the results for multiple realizations of the same NWP saturation. The same conclusion is reached by inspecting the values listed in Table 6 for the three different realizations.

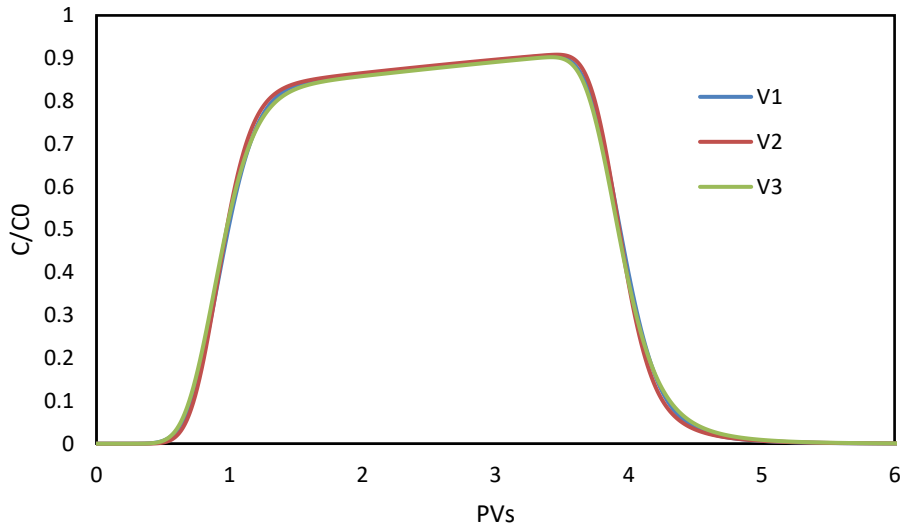


Figure 30: Breakthrough curve for simulations run on pore networks in Figure 29.

The specific interfacial area, a_o, has been empirically correlated to S_{NWP} [64-66] as follows

$$a_o = \alpha(S_{NWP})^{\beta} \quad \textbf{(5.1)}$$

where α and β are fitted parameters. In Figure 31 the specific surface area, a_o, is shown as a function of S_{NWP} along with least-squares regression using α and β as the fitted variables. The best-fit values of these parameters are $\alpha = 9433\ m^{-1}$ and $\beta = 1.29$. Clearly, this yields an excellent fit to the simulated data. In [38, 66] a value of $a_o = 3040 m^{-1}$ was determined for an NWP saturation of $S_{NWP} = 0.5$, which is quite close to the value of $a_o = 3850.6 m^{-1}$ found with the IP simulations. Although the method used to determine the interface area used in this study slightly overestimates a_o when compared to measured values determined in similar studies, it is still well within the expected order of magnitude. In Table 7 a comparison of experimental values for α and β are presented. Although the values for both α and β determined in this study are higher than any of the experimentally determined values, they still are well within reason, as there is much variation in experimentally determined values. In future studies the method of calculating the fluid-fluid interface area could be better calibrated to match experimentally determined values for a_o.

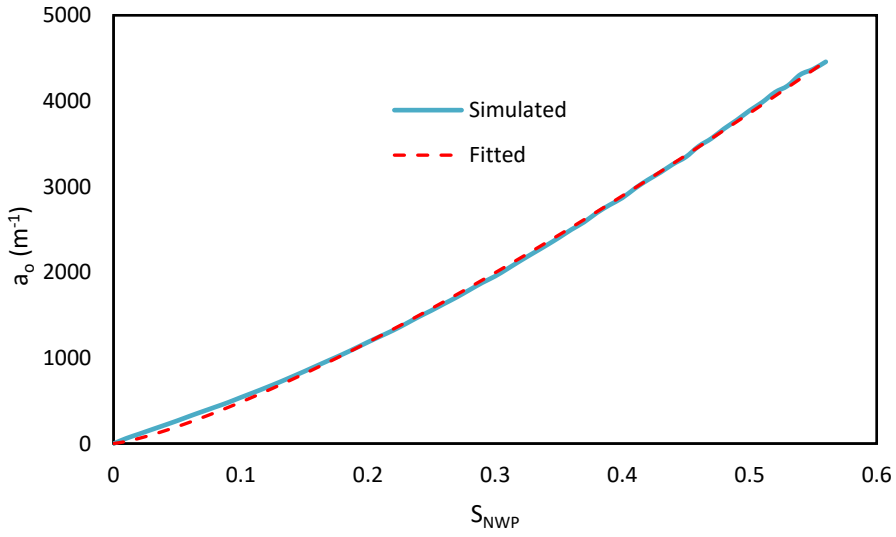


Figure 31: Specific surface area, $a_o(m^{-1})$, fitting for simulated data and fitted data with Eqn. (5.1 where $\alpha = 9433 m^{-1}$ and $\beta = 1.29$.

Table 7: Comparisons of values for α and β

Description	$\alpha(m^{-1})$	β	Reference
IP simulations	9433	1.29	This work
Total surface area of octanol blob using magnetic resonance imaging	3400-6900	0.97-1.00	[64]
Total interfacial area of octanol blob using magnetic resonance imaging	1600-3800	0.92-0.96	[64]
NAPL dissolution experiments	2433-9333	0.38-0.46	[65]

The tortuosity, τ, is found using the method outlined in Section 4.1. Figure 32 presents the tortuosity as a function of S_{NWP} for the three different inlet pore configurations in Figure 29. In [67] it was suggested that τ can be expressed as a function of S_{NWP} using

$$\frac{\tau|_{S_{NWP}}}{\tau|_{S_{NWP=0}}} = (S_{NWP})^{-\lambda} \quad \textbf{(5.2)}$$

where λ is a constant that was found to range anywhere from $\lambda = 0.5$ to more than 2.0. In Figure 32, the predicted tortuosity found using Eqn. (5.2 for $\lambda = 0.1,\ 0.5,$ and 2.0. No matter the value of λ selected, Eqn. (5.2 yields a poor fit to the simulated tortuosity values for NWP saturation values greater than 0.3.

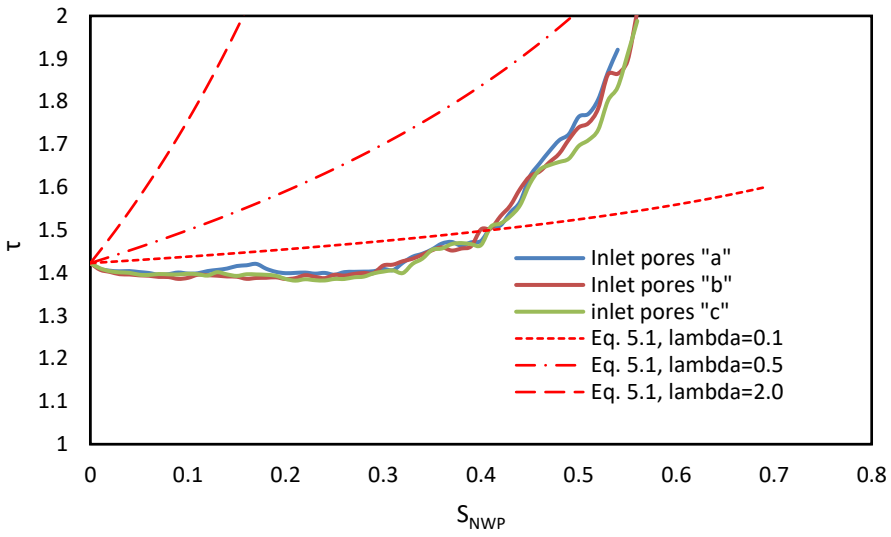


Figure 32: Tortuosity as a function of oil saturation for three different inlet selections, along with predicted curves using Eqn. (5.2 where $\lambda = 0.1, 0.5, \text{and } 2.0$.

5.2 Breakthrough Curve Profiles

There are several mechanisms that can be inferred about the shape of breakthrough curves both at the aquifer and laboratory scale [68]. A total reduction of area underneath the breakthrough curve can indicate either irreversible adsorption or retardation (reversible adsorption), while an ascending plateau indicates reduced deposition, likely resulting from the blocking of the interface by adsorbed particles. Figure 33 presents schematic representation the profiles of elution curves experiencing (a) unlimited irreversible deposition, (b) reduced deposition, (c) enhanced deposition, and (d) acceleration and retardation. An elution curve with a descending plateau is possible when there is either enhanced deposition (such as particles clogging a path) or certain NP behaviors such as size exclusion [23] or velocity profile exclusion [20]. Because the particles used in this simulation are much smaller than the throat diameters a shift towards earlier times is not expected.

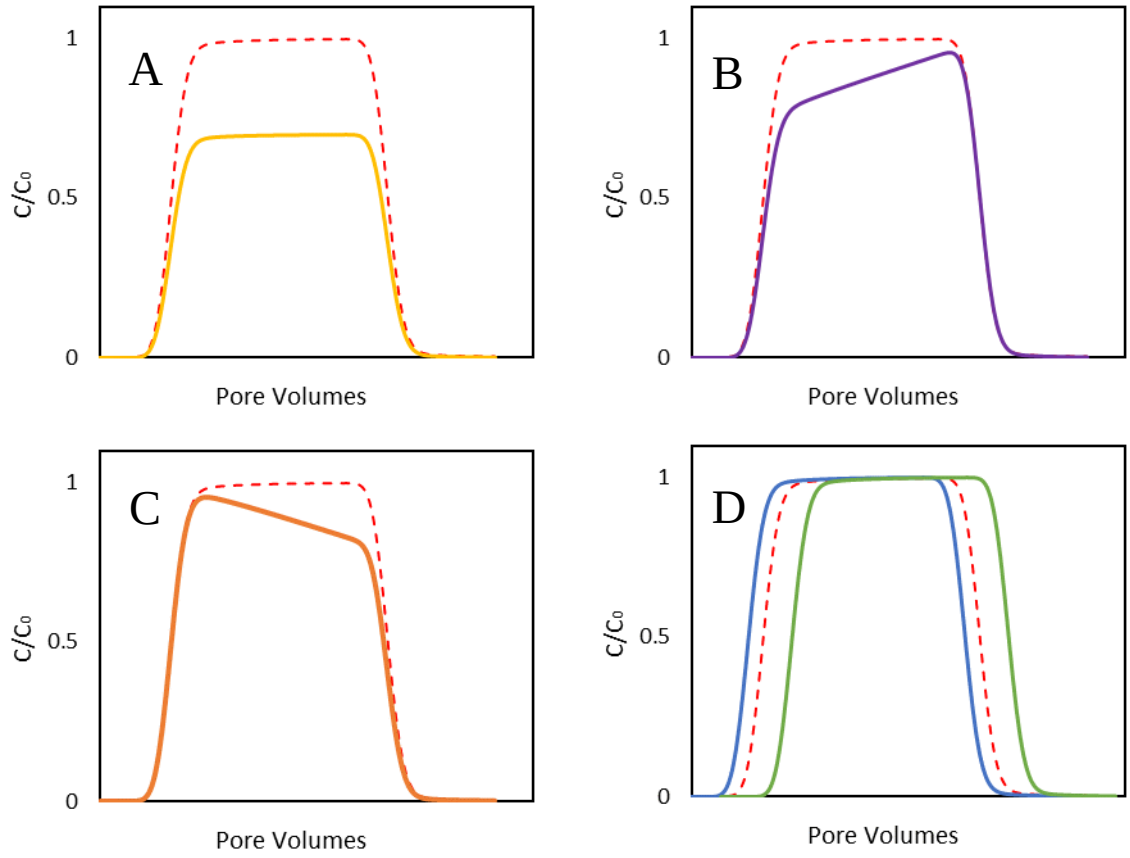


Figure 33: Normalized breakthrough curves of NPs compared to conservative tracer (dashed line) for (a) unlimited irreversible deposition, (b) reduced deposition (blocking), (c) enhanced deposition, (d) acceleration (blue) and retardation (green).

We performed three adsorption simulations where 3 PV of 20 nm particles were injected into columns with $S_{NWP} = 0.01$, 0.1, and 0.5 from Figure 28. The Darcy velocity in these simulations is set to $U_{Darcy} = 10^{-6}(m/s)$, and the simulations were performed for 2000 steps with $t_{step} = 200s$. In two of the simulations the dimensionless adsorption barrier is ϕ_b/k_bT = 8, which results in an adsorption constant of $k_a = 1.52x10^{-6}(m/s)$, whereas in the third simulation ϕ_b/k_bT = 50 which effectively corresponds to negligible adsorption ($k_a \approx 0$). The volumetric fraction of particles in the slug was set to $\Phi_V = 10^{-4}$ for the simulation without adsorption and one of the simulations with adsorption, whereas the second simulation with adsorption had the volumetric fraction of particles set to $\Phi_V = 10^{-5}$.

The extent of total adsorption is quantified by defining the term θ_{total}, which is found using

$$\theta_{total} = \frac{S \sum n_{ads}}{A_{NWP,surf}} \quad \textbf{(5.3)}$$

where S is the cross sectional of a NP, $\sum n_{ads}$ is the total number of adsorbed NPs in the entire network, and $A_{NWP,surf}$ is the total surface area of NWP surface pores. Just like θ, θ_{total} is theoretically bounded by unity and θ_{max}.

The total amount of particles eluted, M_r, and the total fractional coverage, θ_{total}, at the end of the simulations are presented in Table 8. For the simulations without adsorption the volume fraction is $\Phi_V = 10^{-4}$, but the normalized elution curves would be the same no matter what the volume fraction of injected particles is, thus it is appropriate to compare the simulations with no adsorption to the simulation with a different concentration of particles injected.

Table 8: Experimental conditions and results for 3 pore-volume injections

Simulation Number	S_{NWP}	ϕ_b/k_bT	Φ_V	M_r	θ_{total}	Referred to as
1	0.01	50	10^{-4}	1.000	0.000	$S_L A_N P_H$
2	0.01	8	10^{-4}	0.999	0.889	$S_L A_Y P_H$
3	0.01	8	10^{-5}	0.980	0.772	$S_L A_Y P_L$
4	0.1	50	10^{-4}	0.998	0.000	$S_M A_N P_H$
5	0.1	8	10^{-4}	0.978	0.875	$S_M A_Y P_H$
6	0.1	8	10^{-5}	0.871	0.530	$S_M A_Y P_L$
7	0.5	50	10^{-4}	0.947	0.000	$S_H A_N P_H$
8	0.5	8	10^{-4}	0.856	0.613	$S_H A_Y P_H$
9	0.5	8	10^{-5}	0.684	0.150	$S_H A_Y P_L$

Figure 34 presents the elution curves for simulations when $S_{NWP} = 0.01$. There is no visible difference between the elution curves for simulations #1($S_L A_N P_H$) and #2($S_L A_Y P_H$) because in simulation #2($S_L A_Y P_H$) the amount of particles injected is much more than enough to fully cover the water-NWP interfaces with particles, with little change in the concentration of particles in the solute as inferred by $M_r = 0.999$ for simulation #2($S_L A_Y P_H$). Simulation #3 only had a tenth of the particles injected as simulation #2($S_L A_Y P_H$), and the total fractional coverage was still $\theta_{total} = 0.772$. The elution curve for simulation #3($S_L A_Y P_L$) linearly increases in relative concentration from 1 PV to 4 PVs, which is an indicator of blocking as seen in Figure 33.

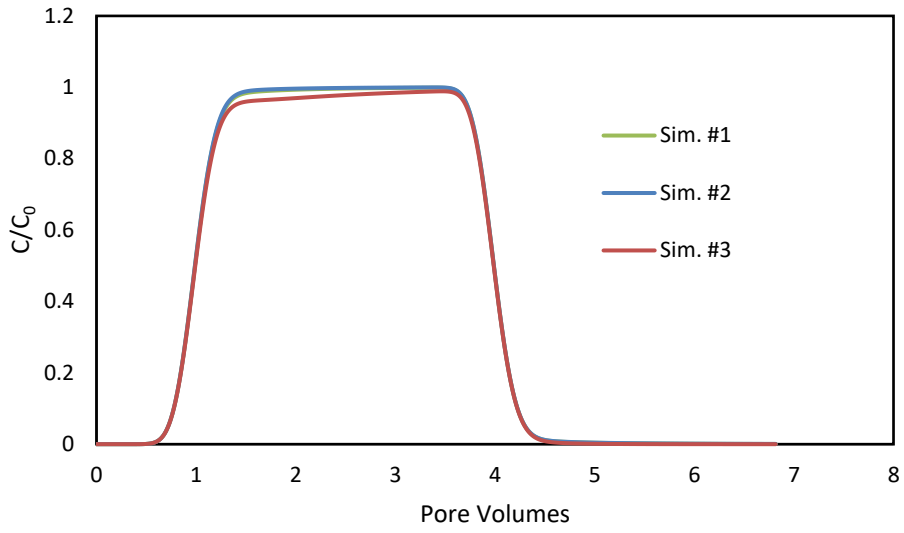


Figure 34: Breakthrough elution curves for simulations #1($S_L A_N P_H$), #2($S_L A_Y P_H$), and #3($S_L A_Y P_L$)

Figure 35 presents the elution data for simulations #4($S_M A_N P_H$), #5($S_M A_Y P_H$), and #6($S_M A_Y P_L$), where $S_{NWP} = 0.1$, while Figure 36 presents the elution curves for simulations #7($S_H A_N P_H$), #8($S_H A_Y P_H$), and #9($S_H A_Y P_L$). In simulation #1($S_L A_N P_H$), the normalized concentration reaches C/C_0 shortly after 1 PV is injected and stays at the same level until 4 PV's are injected. Comparatively in simulation #4 the elution curve continues to increase well after 1 PV is injected. This effect may be confused with blocking, but this is clearly not the case since adsorption is negligible. This increase in concentration is a result of the increase in mechanical dispersion in columns with a higher S_{NWP}. This phenomenon of the shape of the elution curve resembling reduced deposition is even more pronounced in simulation #7($S_H A_N P_H$) where C/C_0 does not even reach unity before the slug is eluted. In simulations #4($S_M A_N P_H$) and #7($S_H A_N P_H$) M_r does not reach unity as a result of particles getting trapped in pores with no flow. Some, but not all of these pores contain fluid-fluid interfaces ("surface NWP pores"). If the simulation were to continue to run M_r would continue to increase until these trapped particles slowly diffuse out of the "dead-end" and no-flow pores and be eluted.

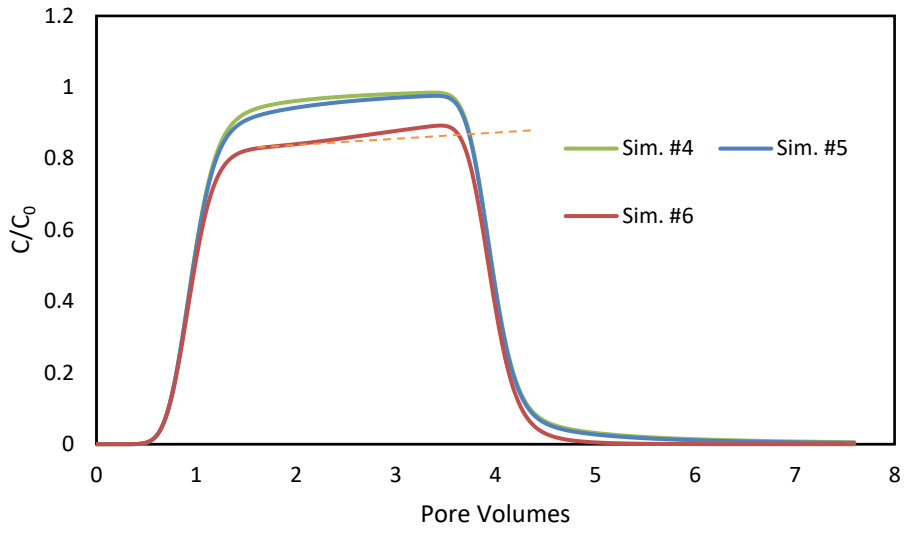


Figure 35: Breakthrough elution curves for simulations #4($S_M A_N P_H$), #5($S_M A_Y P_H$), and #6($S_M A_Y P_L$)

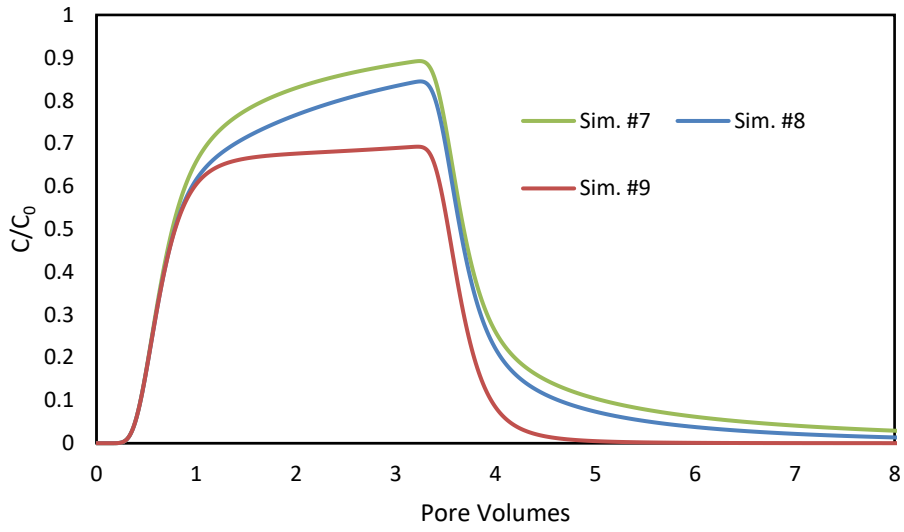


Figure 36: Breakthrough elution curves for simulations #7($S_H A_N P_H$), #8($S_H A_Y P_H$), and #9($S_H A_Y P_L$)

Just like simulation #2($S_L A_Y P_H$), simulation #5($S_M A_Y P_H$) nearly reaches maximum θ_{total} with M_r being close to unity, although there is slight difference between simulations #4($S_M A_N P_H$) and #5($S_M A_Y P_H$). In simulation #6 the recovered mass is 0.871, but the total fractional coverage is $\theta_{total} = 0.530$. The elution curve for simulation #6($S_M A_Y P_L$) (see Figure 35) similarly indicates blocking just like in simulation #3($S_L A_Y P_L$). In simulation #8 it is not clear if there is any blocking, as the elution curve has a shape similar to simulation #7*($S_H A_N P_H$)*. The most noticeable difference between these two curves is that the normalized concentration is consistently lower for simulation #8*($S_H A_Y P_H$)* than simulation #7*($S_H A_N P_H$)*.

In Figure 35, a dashed orange line is presented on the plateau of simulation #6($S_M A_Y P_L$). The purpose of the dashed orange line is to highlight a subtle increase in the slope of this simulation's plateau. This increase is attributed to the blocking having an increasing effect at later times, resulting in a slower rate of adsorption.

A large fraction of the particles injected in simulation #9*($S_H A_Y P_L$)* end up being adsorbed since $M_r = 0.684$, but the total fractional coverage is only $\theta_{total} = 0.150$. There does not appear to be any reduced deposition on this curve, which is a result of only a small fraction of the interface area being covered. Simulations #7*($S_H A_N P_H$)* and #8*($S_H A_Y P_H$)* both have a distinct "tail" at the end of the elution curves, while simulation #9 does not. This difference is because particles that meander into "dead-ends" will eventually diffuse back out of the "dead-ends" in simulations #7*($S_H A_N P_H$)* and #8*($S_H A_Y P_H$)*, whereas in simulation #9 particles that meander into these "dead-ends" are much more likely to simply adsorb onto the water-NWP interface as opposed to diffusing back out of the "dead-ends".

Since the Darcy velocity for these simulations is set to $U_{Darcy} = 10^{-6}(m/s)$, the time to perform a real experiment under the same conditions would take nearly 4 days, which is demanding though not impossible. To observe any noticeable characteristics of the elution curve it was necessary to run the simulations at such low velocities and concentrations. It is possible to achieve high coverage with faster velocities as will be discussed in Section 5.3, but this requires a higher concentration of injected particles. If a high concentration of particles is injected, then there is little observable difference in the elution curves.

5.3 Adsorption and Damkholer Number Analysis

In order to relate the rate of adsorption to the transport rate, two different definitions of the Damkholer numbers are considered. The first Damkholer number, Da_h, compares the rate of particle adsorption at the NWP interface to the rate of transport at the continuum scale (hydrodynamic dispersion)

$$Da_h \equiv \frac{k_a L^2 a_o}{D_L} \quad \textbf{(5.4)}$$

where k_a is the reaction constant found using Eqn. (2.18, L is the column length, a_o is the specific surface area of the water-NWP interface, and D_L is the hydrodynamic dispersion, which is predicted using Eqns.(4.6 and (4.7. A second, pore-level, Damkholer number compares the rate of particle adsorption at the NWP interface to the rate of particle transport to a NWP interface by diffusion

$$Da_d \equiv \frac{k_a \langle d_L \rangle}{D_{AB}} \quad \textbf{(5.5)}$$

where $\langle d_L \rangle$ is the characteristic throat length of the pore network, which was simply taken as the average throat length ($\langle d_L \rangle = 2.37x10^{-4}\, m$).

5.3.1 Effects of Particle Size

It is predicted that as the size of injected NPs decreases, the rate of increase of the total fractional coverage should increase due to three different effects:

- Smaller particles have a higher diffusivity as predicted by the Stokes-Einstein equation in Eqn. (2.7. A higher diffusivity is predicted to result in faster transport of the particles to the water-NWP interface
- The adsorption constant increases with decreasing particle size as a result of a reduction in the barrier to adsorption according to Eqn. (2.18.
- When comparing the total area two different particle sizes can cover, for the same volume of particles is injected, the smaller particles are able to cover a larger total surface area than the larger particles.

In Figure 37, adsorption simulations were performed for 5 different NP sizes with the same volume concentration of $\Phi_V = 0.001$ of particles. All simulations in this section were performed with column A from Figure 28 where $S_{NWP} = 0.1$. A total of 9.48 PVs were injected, which was achieved by running the simulations for 500 steps where $t_{step} = 100s$. The Darcy velocity is $U_{Darcy} = 10^{-5}\, m/s$, and the dimensionless energy barrier is $\phi_b/k_bT = 8$. As predicted, the smaller particles approach the maximum total fractional coverage ($\theta_{max} = 0.91$) at a faster rate than the larger particles.

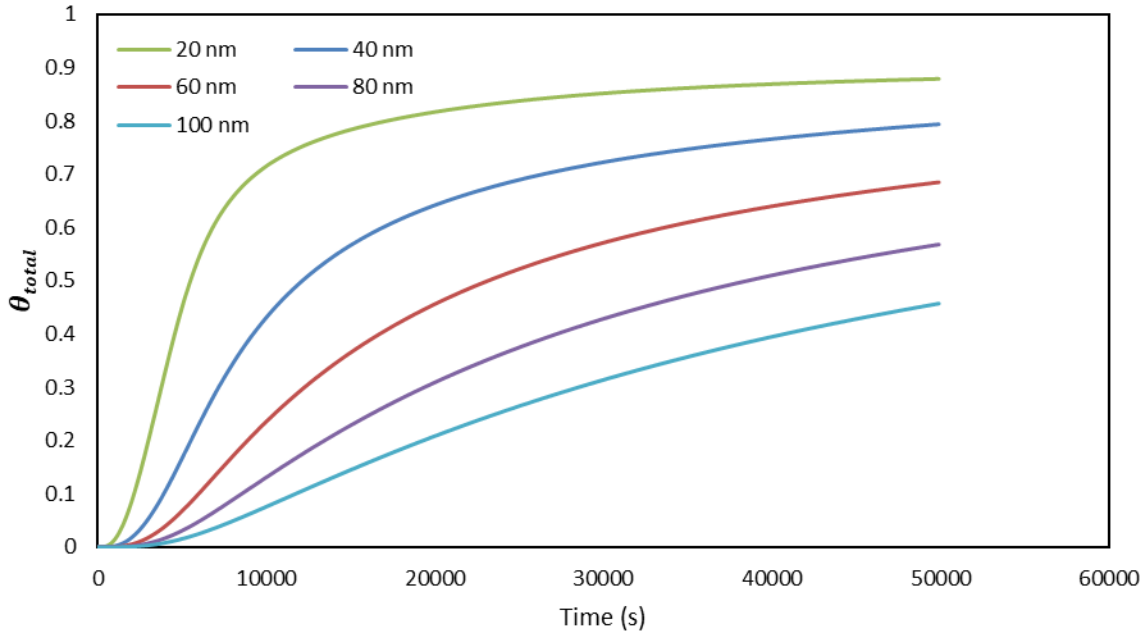


Figure 37: Total fractional coverage as a function of time for $d_p = 20, 40, 60, 80,$ and 100 *nm*. Volume fraction of particles is $\Phi_V = 0.001$.

In order to more accurately compare the effect changing the particle size has on the rate of interfacial coverage, we consider particle concentrations, C, that correspond to the same total particle cross-sectional area per unit volume, C_{area}, for particles of different size, d_p.

$$C = \frac{C_{area}}{\frac{\pi}{4} d_p^2} \qquad \textbf{(5.6)}$$

For the simulation from Figure 37 with $d_p = 20nm$ and $\Phi_V = 0.001$, $C_{area} = 75000m^2/m^3$. In Figure 38, five simulations were run with the same parameters as Figure 37, except with the same C_{area} instead of the same Φ_V. The particle cross-sectional area concentration was taken to be $C_{area} = 75000m^2/m^3$, which is what the C_{area} is for the simulation where $d_p = 20\ nm$ in Figure 37. Even though there is a higher concentration of larger particles than smaller particles, the smaller particles still result in a faster rate of coverage. The change in rate observed in Figure 38 is attributed to the higher diffusivity and higher k_a that

smaller particles possess in relation to larger particles. Furthermore, the effect where a given concentration of smaller particles can cover a greater surface area than the same concentration of larger particles can be observed when comparing Figure 37 and Figure 38.

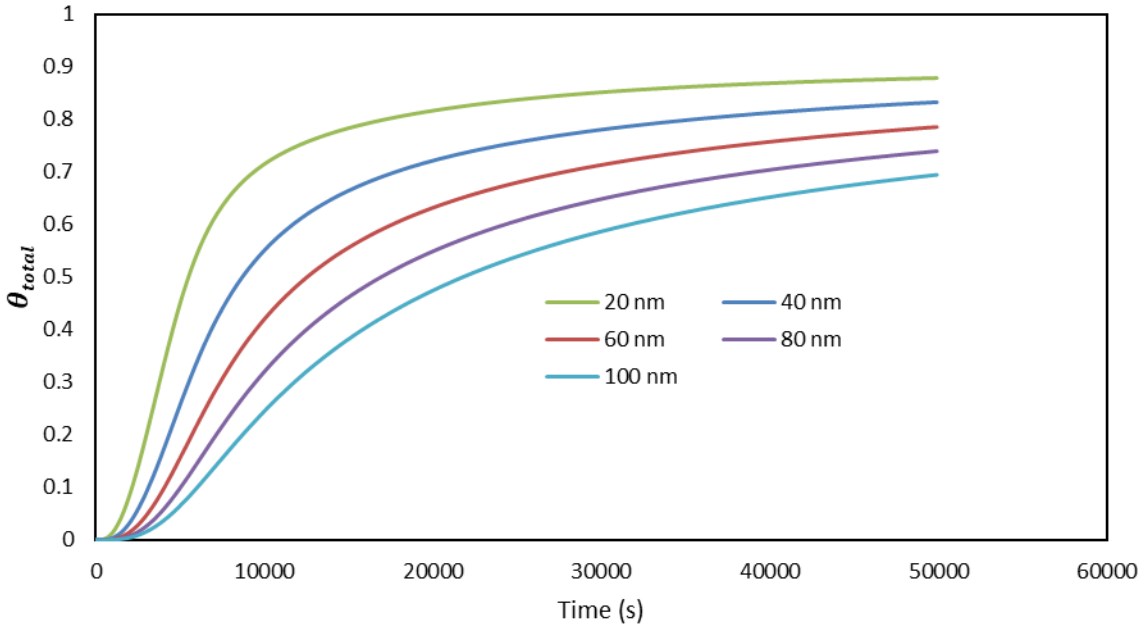


Figure 38: Total fractional coverage, θ_{total}, as a function of time for $d_p = 20, 40, 60, 80, \text{and } 100$ *nm*. Total surface area concentration is $C_{area} = 75000 m^2/m^3$.

Table 9: Damkholer numbers found with Eqns. (5.4 and (5.5 and Peclet numbers for simulations in Figure 37 and Figure 38

$d_p(nm)$	20	40	60	80	100
Da_h	235.7	54.5	23.2	12.7	7.9
Da_d	14.7	7.4	4.9	3.7	2.9
Pe	2317.9	4636.2	6954.8	9273.9	11593.3

It is assumed that HD and VPE will have no significant effect on any of the adsorption simulations run in this section as the largest NPs tested are only $d_P = 100\,nm$, which results in a particle to bead ratio of $d_p/d_b = 5x10^{-5}$. If simulations were performed in packed columns with smaller beads such that d_p/d_b were larger, then adsorption would likely be reduced if HD were to be accounted for. It is not clear what effect VPE would have on adsorption simulations, but it is likely that it would also reduce adsorption.

5.3.2 Effect of Changing Adsorption Barrier and Velocity

The effect that adjusting the adsorption barrier has on the adsorption simulation is tested in this section by running simulations at different values of ϕ_b/k_bT, ranging from 1 to 24. Once again, all simulations were performed using column A with $S_{NWP} = 0.1$ from Figure 28. Simulations were done at a Darcy velocity of $U_{Darcy} = 10^{-5}m/s$ until a total of 9.48 PVs was injected. The time step for these simulations was $t_{step} = 100s$ for a total of 500 steps. Figure 39 presents simulations run with $d_p = 20nm$ run at $\phi_b/k_bT = 1, 2, 4, 8, 12, 16, 18, 20$, and Figure 40 presents simulations with $d_p = 80nm$ and $\phi_b/k_bT = 1, 2, 4, 8, 12, 14, 16, 18$. Table 10 and Table 11 present Da_h and Da_d from the simulations preformed in Figure 39 and Figure 40 respectively.

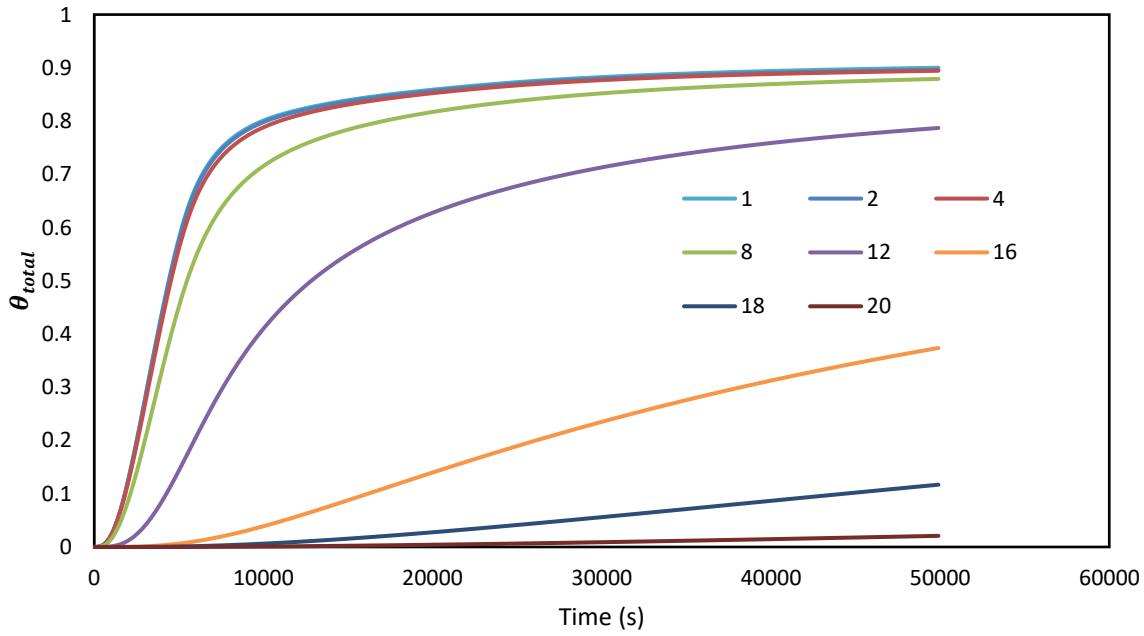


Figure 39: Total fractional coverage, θ_{total}, as a function of time for different values of ϕ_b/k_bT with $d_p = 20\ nm$

Table 10: Data for simulations in Figure 39

ϕ_b/k_bT	Da_h	Da_d	Final θ_{total}
1	91391	5700	0.900
2	47547	2966	0.896
4	9100	568	0.894
8	236	14.7	0.879
12	5.29	0.330	0.787
16	0.112	6.97E-03	0.374
18	1.61E-02	1.00E-03	0.117
20	2.29E-03	1.43E-04	0.021

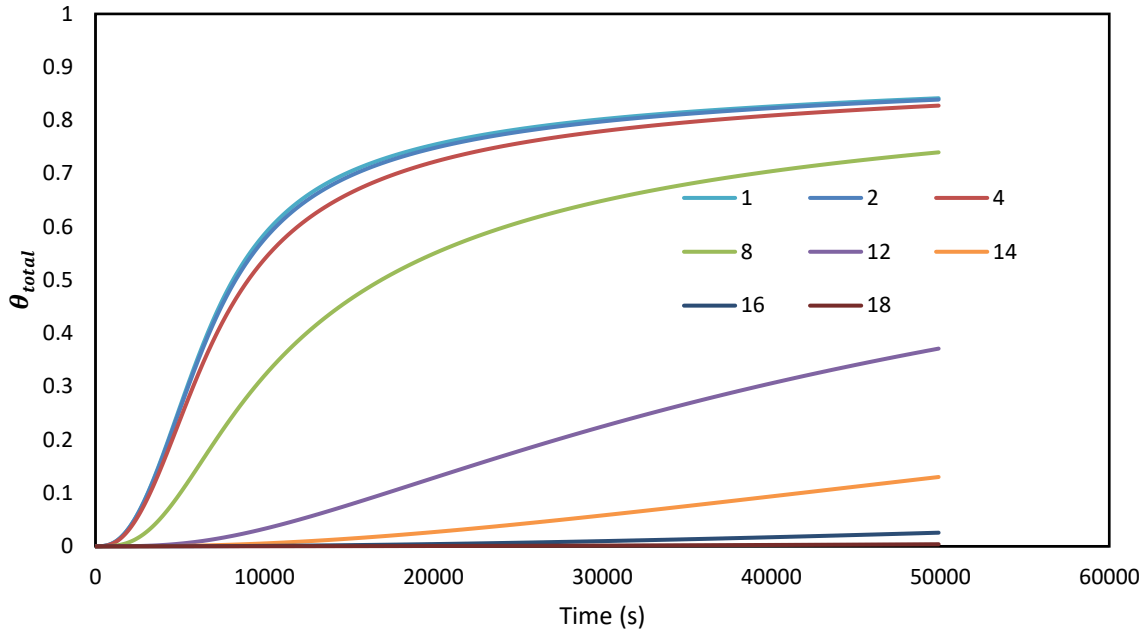


Figure 40: Total fractional coverage, θ_{total}, as a function of time for different values of ϕ_b/k_bT with $d_p = 80\ nm$

Table 11: Data for simulations in Figure 40

ϕ_b/k_bT	Da_h	Da_d	Final θ_{total}
1	4910	1425	0.842
2	2554	742	0.839
4	489	142	0.828
8	12.7	3.68	0.740
12	0.284	8.25E-02	0.371
14	4.15E-02	1.21E-02	0.130
16	6.01E-03	1.74E-03	0.026
18	8.62E-04	2.50E-04	0.0039

The Damkholer numbers can only predict the final adsorption to a certain extent, since they do not depend on the number of PVs injected and the concentration of particles injected. However, when $Da_h >> 1.0$ (transport-limited adsorption) then near complete coverage is achieved, and when $Da_h << 1.0$ (kinetically limited adsorption), θ_{total} is generally less than 0.10.

In Figure 41 adsorption simulations were performed at $U_{Darcy} = 10^{-7}$, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, $10^{-2}\ m/s$ for 9.48 PVs. These simulations had a volume fraction of $\Phi_V = 0.001$, and the adsorption barrier is $\phi_b/k_bT = 8$. All simulations were performed for 500 steps, and the appropriate time step was calculated to achieve the desired final injected pore volume 9.48 PVs. Note that in these simulations the pore-level Damkohler number is unchanged. Results for these simulations are presented in Table 12.

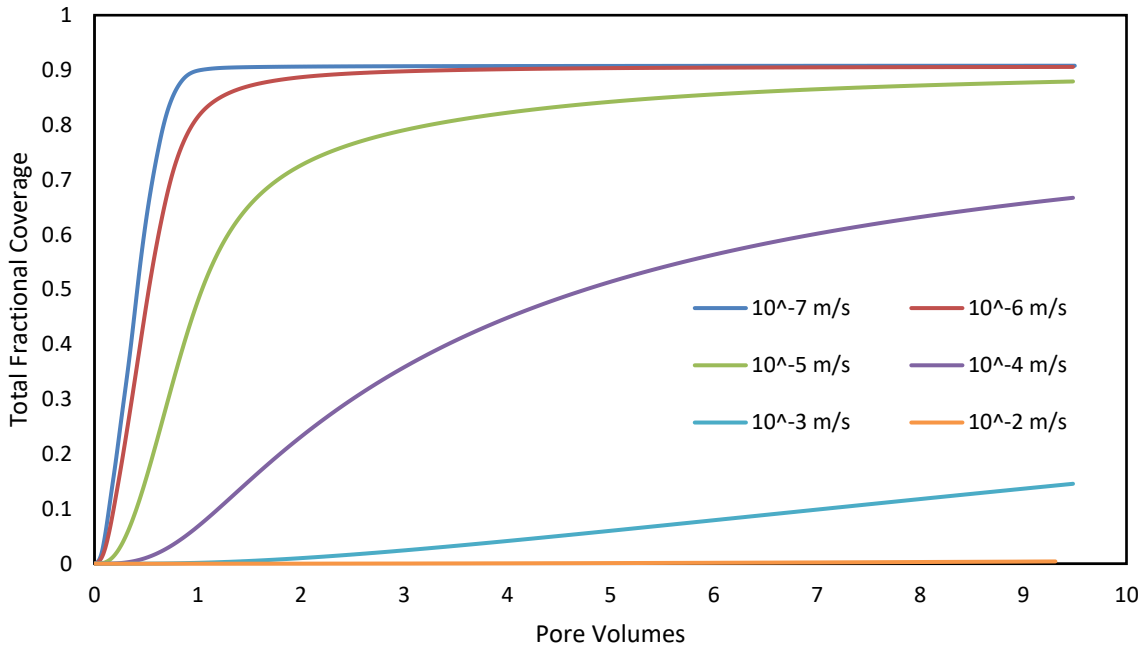


Figure 41: Total fractional coverage, θ_{total}, as a function of time for different Darcy velocities, all with 9.48 pore volumes injected

Table 12: Data for simulations run in Figure 41

$U_{Darcy}(m/s)$	Pe	Da_h	Da_d	Final θ_{total}
10^{-7}	23.2	51589.8	14.7	0.908
10^{-6}	231.8	3234.8	14.7	0.906
10^{-5}	2317.9	235.7	14.7	0.879
10^{-4}	23179.1	18.54	14.7	0.667
10^{-3}	231788.4	1.53	14.7	0.146
10^{-2}	2315507.4	0.130	14.7	0.004

Clearly, transport at both the pore and continuum scale can limit the rate of coverage of fluid interfaces by adsorbed particles.

Chapter 6 Conclusions and Recommendations

6.1 Summary and conclusions

In this work, transport of nanoparticles (NPs) with diameters in the range 20-1000 nm was simulated in pore networks extracted from computer-generated packed columns. The dispersion of NPs at a wide range of velocities and the irreversible adsorption onto a water-non wetting phase (NWP) interface was simulated. The Eulerian approach was taken, which described the transport of NPs in terms of concentration of all particles, as opposed to the Lagrangian approach which describes the trajectory of individual particles typically using random walk particle tracking. The model was developed using OpenPNM, and the porous medium that the network is based on is an unconsolidated column packed with spherical beads. This pore network was created by generating a voxel image of a packed column and then extracting a pore network from the voxel image using PoreSpy's SNOW2 algorithm. This network used pores in the shape of truncated pyramids and throats in the shape of cuboids. The pore network using these shapes was verified by simulating the permeability, tortuosity, and the breakthrough capillary pressure and comparing the results to known correlations.

A model that can determine the longitudinal dispersion coefficient and effective particle velocity for a given NP size and Darcy velocity for the given packed column was built. This was done by preforming transient advection-diffusion simulations where either a step-change or instantaneous pulse is injected at the inlet of the packing, and the analytical solution of the advection-diffusion equation is fitted to the elution. The study focused specifically on the effects of hindered diffusion (HD) and velocity profile exclusion (VPE) on the dispersion coefficient.

A model for simulating the irreversible adsorption of NPs onto a water-NWP interface was also built. The NWP was added to the water-saturated pore network by randomly selecting pores to act as the inlet for an invasion-percolation algorithm, and this allowed a second NWP to be added at the desired saturation. The particles were transported through the pore network using the same transient-advection-diffusion algorithm as before, and adsorption was simulated by adding a sink term to any pores where the NWP was present. This sink term was based on an energy barrier to adsorption, and a blocking term was added that would reduce the rate of adsorption based on the number of NPs already adsorbed onto the interface.

In the dispersion simulations, it was concluded that step-change simulations give more accurate results than pulse simulations, although both generally are adequate. The results from a real-world experiment were simulated, and it was determined that the effects of HD and VPE are insufficient to explain the reported differences NPs experience compared to conservative tracers. The influence of HD and VPE was then analyzed, and it was found that both effects play a significant role when the ratio of particle to bead size is sufficiently high ($d_p/d_b > 0.05$). HD will only have a significant effect when the Peclet number is less than 10, but VPE will occur in all flow regimes.

The prospect of predicting properties of the transport of particles based on their breakthrough curve was tested, and it was determined that effects such as reduced attachment (blocking) and total adsorption are observable when the breakthrough curve is compared to a conservative tracer. However, these effects were only observed when the concentration of particles is low, and the velocity is low. The influence that particle size, barrier to adsorption, and velocity has on adsorption was also tested. It was found that particle size has a significant effect on adsorption due to smaller particles being able to cover more surface area for the same volume of particles injected, smaller particles being able to diffuse towards the water-NWP interface at a greater rate, and smaller particles having a smaller barrier to adsorption. Finally, it was determined than when the Damkholer number is large (Da_h >>10.0) that the rate that the water-NWP interface is covered

is limited by diffusion, and when it is small (Da_h << 10.0) the rate that the water-NWP interface is covered is limited by the energy barrier to adsorption.

6.2 Future Recommendations

The following are recommended for future studies

- Preform real-world experiments to test the predictions of pore network simulations. This is possible using well-characterized NPs (e.g. ethyl cellulose NPs) and porous media (transparent glass micromodels).
- There are several features that could be added to both the dispersion and irreversible adsorption models. Accounting for additional effects such as retardation (via reversible adsorption) would be interesting, as would be particle attachment onto the solid surface. Additionally, the effect of variable particle size, e.g. by particle coagulation, would be interesting as it would modify the flow permeability.
- A more detailed account of the fluid-fluid interfacial area available for particle attachment would be desirable. Presently the pore network model ignores the presence of fluid interfaces in NWP-invaded pores.

Bibliography

[1] C.V. Chrysikopoulos and Y. Sim, "'One-dimensional virus transport in homogeneous porous media with time-dependent distribution coefficient," *Journal of Hydrology*, vol. 185, no. 1, pp. 199-219.

[2] M.Y. Corapcioglu and S. Jiang, "'Colloid-facilitated groundwater contaminant transport," *Water Resour.Res.*, vol. 29, no. 7, pp. 2215-2226.

[3] A. Abdel-Salam and C.V. Chrysikopoulos, "'Analysis of a model for contaminant transport in fractured media in the presence of colloids," *Journal of Hydrology*, vol. 165, no. 1, pp. 261-281.

[4] S.S. Patil, U.U. Shedbalkar, A. Truskewycz, B.A. Chopade and A.S. Ball, "'Nanoparticles for environmental clean-up: A review of potential risks and emerging solutions," *Environmental Technology & Innovation*, vol. 5, pp. 10-21.

[5] N. Wilson, "'Nanoparticles," *Bioscience*, vol. 68, no. 4, Apr 01, pp. 241-246.

[6] Y. Jin and M. Flury, "'Fate and Transport of Viruses in Porous Media," *Adv.Agron.*, vol. 77, pp. 39-102.

[7] J.C. Romero, "'The Movement of Bacteria and Viruses Through Porous Media," *Groundwater*, vol. 8, no. 2, pp. 37-48.

[8] A. Razak Ismail, T.C. Seong, N.A. Buang, W. Rosli and W. Sulaiman, "'Improve Performance of Water-based Drilling Fluids Using Nanoparticles," *Sriwijaya International Seminar on Energy and Environmental Science & Technology Palembang*.

[9] S. Ko and C. Huh, "'Use of nanoparticles for oil production applications," *Journal of Petroleum Science and Engineering*, vol. 172, pp. 97-114.

[10] M. Alaskar, "'In-Situ Multifunctional Nanosensors for Fractured Reservoir Characterization," *ProQuest Dissertations and Theses*.

[11] O.V. Salata, "'Applications of nanoparticles in biology and medicine," *Journal of Nanobiotechnology*, vol. 2, no. 1, pp. 3.

[12] M.G.L. Ntlailane and J. Wichmann, "'Effectiveness of N95 respirators for nanoparticle exposure control (2000–2016): a systematic review and meta-analysis," *Journal of Nanoparticle Research*, vol. 21, no. 8, pp. 170.

[13] J. Gostick, M. Aghighi, J. Hinebaugh, T. Tranter, M.A. Hoeh, |. Forschungszentrum, J.H. Day, B. Spellacy, M.H. Sharqawy, A. Burns, W. Lehnert, F. Jülich, R. Aachen and A. Putz, "'OpenPNM: A Pore Network Modeling Package," *Computing in Science & Engineering 18, no. 4*.

[14] G. Boccardo, T. Tosco, A. Fujisaki, F. Messina, A. Raoof, D.R. Aguilera, E. Crevacore, D.L. Marchisio and R. Sethi, "'A review of transport of nanoparticles in porous media," *Nanomaterials for the Detection and Removal of Wastewater Pollutants*, pp. 351.

[15] X. Meng and D. Yang, "'Pore-network modeling of particle dispersion in porous media," *Colloids and surfaces. A, Physicochemical and engineering aspects*, vol. 580, Nov 05, pp. 123768.

[16] D. Ding, D.A. Benson, A. Paster and D. Bolster, "'Modeling bimolecular reactions and transport in porous media via particle tracking," *Advances in water resources*, vol. 53, Mar, pp. 56-65.

[17] A. Paster, D. Bolster and D.A. Benson, "'Particle tracking and the diffusion-reaction equation," *Water Resour.Res.*, vol. 49, no. 1, pp. 1-6.

[18] E.K. Gnanapragasam, C. Yu, G. Whelan, W.B. Mills, J.P. McDonald, C.S. Lew, C.Y. Hung and D. Hoffmeyer, "'Comparison of multimedia model predictions for a contaminant plume migration scenario," *J.Contam.Hydrol.*, vol. 46, no. 1, pp. 17-38.

[19] J. Delgado, "'A critical review of dispersion in packed beds," *Heat Mass Transfer*, vol. 42, no. 4, Feb, pp. 279-310.

[20] S.C. James and C.V. Chrysikopoulos, "'Effective velocity and effective dispersion coefficient for finite-sized particles flowing in a uniform fracture," *Journal of Colloid and Interface Science*, vol. 263, no. 1, pp. 288.

[21] C.V. Chrysikopoulos and V.E. Katzourakis, "'Colloid particle size-dependent dispersivity," *Water resources research*, vol. 51, no. 6, pp. 4668-4683.

[22] K. He, S.T. Retterer, B.R. Srijanto, J.C. Conrad and R. Krishnamoorti, "'Transport and Dispersion of Nanoparticles in Periodic Nanopost Arrays," *ACS nano*, vol. 8, no. 5, May 27, pp. 4221-4227.

[23] S. Sirivithayapakorn and A. Keller, "'Transport of colloids in saturated porous media: A pore-scale observation of the size exclusion effect and colloid acceleration," *Water resources research*, vol. 39, no. 4, Apr 30, pp. 1109-n/a.

[24] S. Reich, A. Svidrytski, D. Hlushkou, D. Stoeckel, C. Kübel, A. Höltzel and U. Tallarek, "'Hindrance Factor Expression for Diffusion in Random Mesoporous Adsorbents Obtained from Pore-Scale Simulations in Physical Reconstructions," *Ind. Eng. Chem. Res.*, vol. 57, no. 8, pp. 3031.

[25] E.M. Renkin, "'Filtration, diffusion, and molecular sieving through porous cellulose membranes," *The Journal of general physiology*, vol. 38, no. 2, Nov 20, pp. 225-243.

[26] J.R. Welty, C.E. Wicks, R.E. Wilson and G.L. Rorrer, "'Fundamentals of Momentum, Heat, and Mass Transfer,", ed. 5th Edition, 2008.

[27] S.A. Bradford, S. Torkzaban and A. Shapiro, "'A Theoretical Analysis of Colloid Attachment and Straining in Chemically Heterogeneous Porous Media," *Langmuir*, vol. 29, no. 23, Jun 11, pp. 6944-6952.

[28] B. Ju and T. Fan, "'Experimental study and mathematical model of nanoparticle transport in porous media," *Powder technology*, vol. 192, no. 2, pp. 195-202.

[29] N. Bizmark, J. Schneider, R.D. Priestley and S.S. Datta, "'Multiscale dynamics of colloidal deposition and erosion in porous media," *Science advances*, vol. 6, no. 46, Nov 13,.

[30] P. Bedrikovetsky, F.D. Siqueira, C.A. Furtado and A.L.S. Souza, "'Modified Particle Detachment Model for Colloidal Transport in Porous Media," *Transp Porous Med*, vol. 86, no. 2, Aug 07, pp. 353-383.

[31] P. Bedrikovetsky, A. Zeinijahromi, F.D. Siqueira, C.A. Furtado and de Souza, Antonio Luiz S, "'Particle Detachment Under Velocity Alternation During Suspension Transport in Porous Media," *Transp Porous Med*, vol. 91, no. 1, Sep 14, pp. 173-197.

[32] D. Prédélus, L. Lassabatere, C. Louis, H. Gehan, T. Brichart, T. Winiarski and R. Angulo-Jaramillo, "'Nanoparticle transport in water-unsaturated porous media: effects of solution ionic strength and flow rate," *J Nanopart Res*, vol. 19, no. 3, Mar 10, pp. 1.

[33] B. Gao, T.S. Steenhuis, Y. Zevi, V.L. Morales, J.L. Nieber, B.K. Richards, J.F. McCarthy and J.-. Parlange, "'Capillary retention of colloids in unsaturated porous media," *Water resources research*, vol. 44, no. 4, Apr 22, pp. W04504-n/a.

[34] M.Y. Corapcioglu and H. Choi, "'Modeling Colloid Transport in Unsaturated Porous Media and Validation with Laboratory Column Data," *Water resources research*, vol. 32, no. 12, pp. 3437-3449.

[35] T. Cheng and J.E. Saiers, "'Mobilization and transport of in situ colloids during drainage and imbibition of partially saturated sediments," *Water resources research*, vol. 45, no. 8, Aug 11, pp. W08414-n/a.

[36] G. Chen and M. Flury, "'Retention of mineral colloids in unsaturated porous media as related to their surface properties," *Colloids and surfaces. A, Physicochemical and engineering aspects*, vol. 256, no. 2, pp. 207-216.

[37] Z. Adamczyk, "'Irreversible Adsorption of Particles,", pp. 251-374.

[38] N. Bizmark and M.A. Ioannidis, "'Ethyl Cellulose Nanoparticles at the Alkane–Water Interface and the Making of Pickering Emulsions," *Langmuir*, vol. 33, no. 40, Oct 10, pp. 10568-10576.

[39] A. Koponen, M. Kataja and J. Timonen, "'Permeability and effective porosity of porous media," *Physical review.E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, vol. 56, no. 3, pp. 3319-3325.

[40] Ravi Suman and D. Ruth, "'Formation factor and tortuosity of homogeneous porous media," *Transport in porous media*, vol. 12, no. 2, pp. 185-206.

[41] R.A. Greenkorn, "'Flow phenomena in porous media : fundamentals and applications in petroleum, water, and food production,", 1983.

[42] R.A. Freeze and J.A. Cherry, "'Groundwater,", vol. 6th ed, 1979.

[43] J. Bear and A.H.-. Cheng, "'Modeling Groundwater Flow and Contaminant Transport,", vol. 23, 2010.

[44] C.C. Miller, "'The Stokes-Einstein law for diffusion in solution," *Proceedings of the Royal Society of London.Series A, Containing Papers of a Mathematical and Physical Character*, vol. 106, no. 740, pp. 724-749.

[45] M. Auset and A.A. Keller, "'Pore-scale processes that control dispersion of colloids in saturated porous media," *Water resources research*, vol. 40, no. 3, Mar 05, pp. W03503-n/a.

[46] G.I.Taylor, "'Dispersion of soluble matter in solvent flowing slowly through a tube ,", pp. 186-203.

[47] X. Yang, Y. Mehmani, W.A. Perkins, A. Pasquali, M. Schönherr, K. Kim, M. Perego, M.L. Parks, N. Trask, M.T. Balhoff, M.C. Richmond, M. Geier, M. Krafczyk, L. Luo, A.M. Tartakovsky and T.D. Scheibe, "'Intercomparison of 3D pore-scale flow and solute transport simulation methods," *Advances in water resources*, vol. 95, no. C, Sep 01, pp. 176.

[48] J. Gostick, Z. Khan, T. Tranter, M. Kok, M. Agnaou, M. Sadeghi and R. Jervis, "'PoreSpy: A Python Toolkit for Quantitative Analysis of Porous Media Images," *Journal of open source software*, vol. 4, no. 37, May 01, pp. 1296.

[49] M. Akbari, D. Sinton and M. Bahrami, "'Viscous flow in variable cross-section microchannels of arbitrary shapes," *International journal of heat and mass transfer*, vol. 54, no. 17, pp. 3970-3978.

[50] M. Bahrami, M. Michael Yovanovich and J. Richard Culham, "'A novel solution for pressure drop in singly connected microchannels of arbitrary cross-section," *International journal of heat and mass transfer*, vol. 50, no. 13, pp. 2492-2502.

[51] M.A. Sadeghi, M. Agnaou, J. Barralet and J. Gostick, "'Dispersion modeling in pore networks: A comparison of common pore-scale models and alternative approaches," *Journal of contaminant hydrology*, vol. 228, Jan, pp. 103578.

[52] A. Ogata, "'Theory of Dispersion in a Granular Medium GEOLOGICAL SURVEY PROFESSIONAL PAPER 411-1," *GEOLOGICAL SURVEY PROFESSIONAL PAPER*.

[53] A.E. Scheidegger, "'General theory of dispersion in porous media," *J.Geophys.Res.*, vol. 66, no. 10, pp. 3273-3278.

[54] N. Bizmark, "'Dynamic Surface Tension as a Probe of Irreversible Adsorption of Nanoparticles at Fluid-Fluid Interfaces,", April 26,.

[55] W.M. Alley, T.E. Reilly and O.L. Franke, "'Sustainability of ground-water resources,".

[56] J. Fu, H.R. Thomas and C. Li, "'Tortuosity of porous media: Image analysis and physical simulation," *Earth-science reviews*, vol. 212, Jan, pp. 103439.

[57] M.A. Ioannidis, I. Chatzis, C. Lemaire and R. Perunarkilli, "'Unsaturated hydraulic conductivity from nuclear magnetic resonance measurements," *Water resources research*, vol. 42, no. 7, Jul 14, pp. W07201-n/a.

[58] K.M. Ng, H.T. Davis and L.E. Scriven, "'Visualization of blob mechanics in flow through porous media," *Chemical Engineering Science*, vol. 33, no. 8, pp. 1009.

[59] C.R. Wilke and P. Chang, "'Correlation of diffusion coefficients in dilute solutions," *AIChE J.*, vol. 1, no. 2, pp. 264-270.

[60] S. Sirivithayapakorn and A. Keller, "'Transport of colloids in saturated porous media: A pore-scale observation of the size exclusion effect and colloid acceleration," *Water resources research*, vol. 39, no. 4, Apr 30, pp. 1109-n/a.

[61] M. Auset and A.A. Keller, "'Pore-scale processes that control dispersion of colloids in saturated porous media," *Water resources research*, vol. 40, no. 3, Mar 05, pp. W03503-n/a.

[62] C. Bianco, J.E. Patiño Higuita, T. Tosco, A. Tiraferri and R. Sethi, "'Controlled deposition of particles in porous media for effective aquifer nanoremediation," *Scientific reports*, vol. 7, no. 1, Oct 11, pp. 12992-10.

[63] J. M. P. Q. Delgado, "'Longitudinal and transverse dispersion in porous media,", vol. 85, no. 9, pp. 1245.

[64] M.L. Johns and L.F. Gladden, "'Magnetic Resonance Imaging Study of the Dissolution Kinetics of Octanol in Porous Media," *Journal of Colloid and Interface Science*, vol. 210, no. 2, Feb 15, pp. 261-270.

[65] L. Zhong, A.S. Mayer and G.A. Pope, "'The effects of surfactant formulation on nonequilibrium NAPL solubilization," *Journal of Contaminant Hydrology*, vol. 60, no. 1-2, -01, pp. 55.

[66] R. Sharmin, M.A. Ioannidis and R.L. Legge, "'Effect of nonionic surfactant partitioning on the dissolution kinetics of residual perchloroethylene in a model porous medium," *Journal of contaminant hydrology*, vol. 82, no. 1, pp. 145-164.

[67] B. Ghanbarian, A.G. Hunt, R.P. Ewing and M. Sahimi, "'Tortuosity in Porous Media: A Critical Review," *Soil Science Society of America journal*, vol. 77, no. 5, Sep, pp. 1461-1477.

[68] P. Babakhani, J. Bridge, R. Doong and T. Phenrat, "'Continuum-based models and concepts for the transport of nanoparticles in saturated porous media: A state-of-the-science review," *Advances in colloid and interface science*, vol. 246, Aug, pp. 75-104.

Appendices

Appendix A

Cylinder Packer and Extractor Code

Here the code to generate a voxel image and extract the pore network is presented.

In [1]:

```
import porespy as ps
import openpnm as op
import numpy as np
import matplotlib.pyplot as plt
from edt import edt
from __sssp__ import pseudo_gravity_packing
```

Define desired dimensions for the packed cylinder, as well as the desired voxel radius for a single bead. I found that for a 15cm column that 28 voxels is the maximum radius before there is file size errors

In [2]:

```
sph_vox = 16
cyl_dia = 2.5e-2
cyl_len = 15e-2
sph_rad = 1e-3
voxel_size = (2*sph_rad)/sph_vox
voxel_volume = voxel_size**3
```

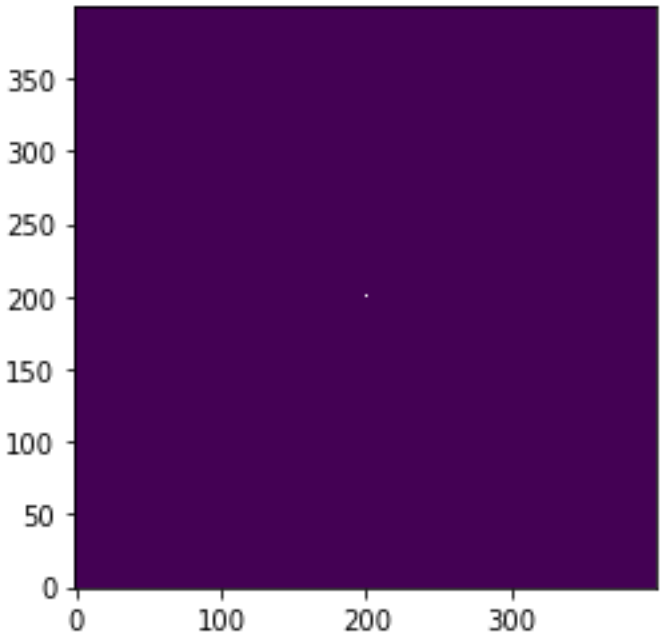
Firstly, an empty voxel image with the same dimensions as the cylinder is generated

In [3]:

```
cyl_lv = int(cyl_len*sph_vox/sph_rad)
cyl_dv = int(cyl_dia*sph_vox/sph_rad)
cyl_rv = int(cyl_dv/2)
temp = np.ones([cyl_lv, cyl_dv, cyl_dv], dtype=bool)
temp[:, cyl_rv, cyl_rv] = 0

ps.imshow(temp[100, ...]);
```

Out[3]:

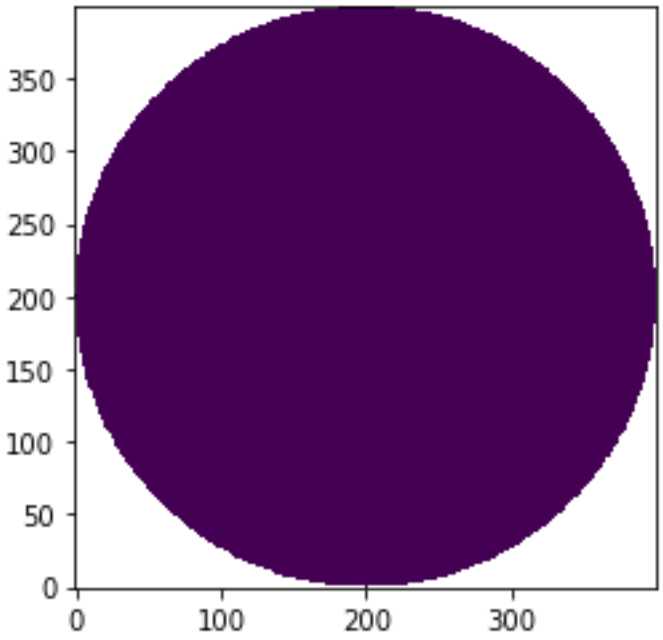
```
<AxesSubplot:>
```



Next, a cylinder is made using the Euclidean Distance Transform

In [4]:

```
cyl = edt(temp) <= cyl_rv
ps.imshow(cyl[100, ...]);
Vbulk = cyl.sum()
```



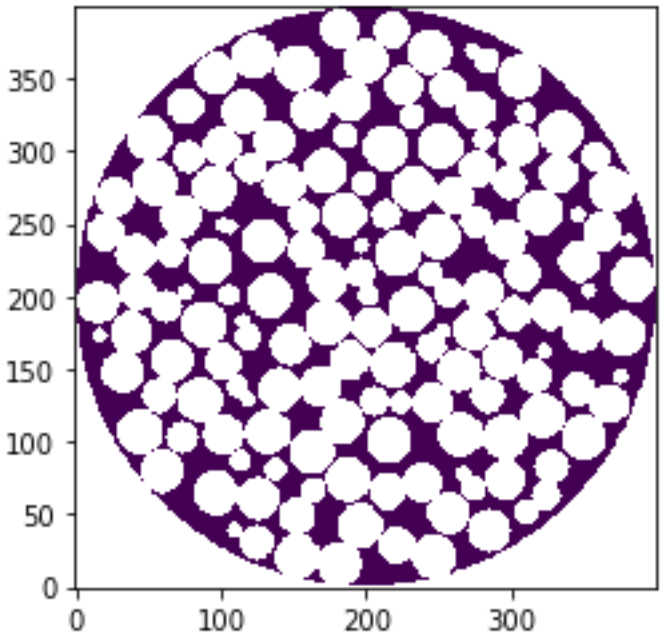
In [5]:

```
spheres = pseudo_gravity_packing(im=cyl, r=sph_vox, max_iter=25000)
ps.imshow(spheres[100, ...]);
```

```
Adding monodisperse spheres of radius 16

 47%|████      | 11679/25000 [11:13<12:47, 17.35it/s]

A total of 11678 spheres were added
```

Out[5]:

```
<AxesSubplot:>
```



In [6]:

```
org_size = 1e-3/28
ratio = 1/1
snow = ps.networks.snow2(phases=spheres,
                         boundary_width=[0, 0, 5],
                         voxel_size=org_size*ratio,
                         parallelization={"divs": [1, 1, 6]})
```

In [16]:

```
settings = {'pore_shape': 'pyramid',
            'throat_shape': 'cuboid',
            'pore_diameter': 'equivalent_diameter',
            'throat_diameter': 'inscribed_diameter'}
pn, geo = op.io.from_porespy(snow.network, settings=settings)
op.io.PNM.save_project(project=pn.project,filename='largenetwork.pnm')
```

Appendix B

Packed Column Dispersion

Here the code presenting generating an elution curve in a packed column and the curve fitting where both the velocity and longitudinal dispersion coefficient as variables is presented

In [1]:

```
import numpy as np
import openpnm as op
import matplotlib.pyplot as plt

np.set_printoptions(precision=5)
np.set_printoptions(threshold=np.inf)
np.random.seed(27)
from scipy.constants import Avogadro as N_A
from scipy.constants import pi
from scipy.optimize import curve_fit
from scipy import special
import Dispersion_Functions as DF
```

In [2]:

```
#dp = particle diameter. if solvent is desired set dp = 'solvent'
#HD and SE are Hindered Diffusion, Size Exculsion, and Taylor Dispersion r
espectivelly
#these inputs are True/False values

dp = 20e-9 # m
num_steps = 1000
particle_density = 1140 # kg/m^3 for EC
vol_frac = 0.0001  # volume particles / volume injected
darcy_velocity = 1.0e-5 # m/s
HD = False
SE = False

if isinstance(dp, str) == True:
    inj_conc = 1.0 # g/m^3
else:
    inj_conc = vol_frac/((pi/6)*dp**3) # particles/m^3

proj = op.io.PNM.load_project(filename='1to1_res_28vox_corrected.pnm')
```

```
pn = proj['net_01']
geo = proj['geo_01']
h = pn.check_network_health()
op.topotools.trim(network=pn, pores=h['trim_pores'])
geo['pore.surface_area'][19652] = np.average(geo['pore.surface_area'])

inlet_pores = pn.pores(['zmin'])
outlet_pores = pn.pores(['zmax'])
nip = [x for x in pn.Ps if x not in inlet_pores]
body_pores = [x for x in nip if x not in outlet_pores]
inlet_throats = pn.find_neighbor_throats(pores=inlet_pores, mode='xor')
outlet_throats = pn.find_neighbor_throats(pores=outlet_pores, mode='xor')
```

In [3]:

```
ratio = 1/1
D = np.max(pn['pore.coords'][:,0:1])-np.min(pn['pore.coords'][:,0:1])
Dc = D*2e-3/(2.5e-2)
L = pn['pore.coords'][outlet_pores][0,2] - pn['pore.coords'][inlet_pores][
0,2]
A = np.pi/4.0*(D)**2.0
porosity = np.sum(geo['pore.region_volume'])/(A*L)


water = op.phases.Water(network=pn)
phys_water = op.physics.Basic(network=pn, phase=water, geometry=geo)

permeability = DF.permeability(pn,geo,water,D,L)

mod = op.models.physics.ad_dif_conductance.ad_dif

phys_water.add_model(propname='throat.ad_dif_conductance', model=mod, s_sc
heme='hybrid')
phys_water.regenerate_models('throat.ad_dif_conductance')

mu = water['pore.viscosity'].max()
del_P = mu*darcy_velocity*L/(permeability)

sf = op.algorithms.StokesFlow(network=pn, phase=water,)
sf.set_value_BC(pores=inlet_pores, values=del_P)
sf.set_value_BC(pores=outlet_pores, values=0.0)
sf.run();
```

Here the changes that account for either the hindered diffusion effect or the velocity profile exclusion effect are made.

In [4]:

```
if dp == 'solvent':
    water['throat.diffusivity'] = 1.85e-9 #m^2/s
    water['pore.diffusivity'] = 1.85e-9 #m^2/s
    D_SE = 1.85e-9 #m^2/s


else:
    flow = sf.rate(throats=pn.Ts,mode='single')
    water['throat.S_E_diffusivity'] = DF.Stokes_Einstein(water['throat.tem
perature'],water['throat.viscosity'],dp)
    water['pore.S_E_diffusivity'] = DF.Stokes_Einstein(water['pore.tempera
ture'],water['pore.viscosity'],dp)
    D_SE = water['throat.S_E_diffusivity'][0]
    phi = dp/geo['throat.diameter']

    if HD == True:
        water['throat.HD_diffusivity'] = DF.Throat_Hindered_Diffusion(geo[
'throat.diameter'],water['throat.S_E_diffusivity'],dp)
        water['pore.HD_diffusivity'] = DF.Pore_Hindered_Diffusion(geo['por
e.diameter'],water['pore.S_E_diffusivity'],dp)
        if SE == True:
            water['throat.HD_SE_diffusivity'] = DF.Throat_Size_Exculsion_D
ispersivity(geo['throat.diameter'],water['throat.HD_diffusivity'],flow,dp)
            water.add_model(propname='throat.diffusivity',model=op.models.
misc.constant, value=water['throat.HD_SE_diffusivity'])
            water.add_model(propname='pore.diffusivity',model=op.models.mi
sc.constant, value=water['pore.HD_diffusivity'])
            phys_water['throat.hydraulic_conductance_old'] = phys_water['t
hroat.hydraulic_conductance']
            phys_water['throat.mask'] = (1+phi-(1/4)*phi**2)
            phys_water.add_model(propname='throat.hydraulic_conductance',
                 model=op.models.misc.product, prop1='throat.hydraulic_con
ductance_old', prop2='throat.mask')
        else:
            water['throat.HD_TD_diffusivity'] =DF.Taylor_Dispersivity(geo[
'throat.diameter'],water['throat.HD_diffusivity'],flow,dp)
            water.add_model(propname='throat.diffusivity', model=op.models
.misc.constant,value=water['throat.HD_TD_diffusivity'])
            water.add_model(propname='pore.diffusivity',model=op.models.mi
sc.constant, value=water['pore.HD_diffusivity'])
```

```
    else:
        if SE == True:
            water['throat.SE_diffusivity'] = DF.Throat_Size_Exculsion_Disp
ersivity(geo['throat.diameter'],water['throat.S_E_diffusivity'],flow,dp)
            water.add_model(propname='throat.diffusivity',model=op.models.
misc.constant, value=water['throat.SE_diffusivity'])
            phys_water['throat.hydraulic_conductance_old'] = phys_water['t
hroat.hydraulic_conductance']
            phys_water['throat.mask'] = (1+phi-(1/4)*phi**2)
            phys_water.add_model(propname='throat.hydraulic_conductance',
                 model=op.models.misc.product, prop1='throat.hydraulic_con
ductance_old', prop2='throat.mask')
            water.add_model(propname='pore.diffusivity',model=op.models.mi
sc.constant, value=water['pore.S_E_diffusivity'])
        else:
            water['throat.TD_diffusivity'] = DF.Taylor_Dispersivity(geo['t
hroat.diameter'],water['throat.S_E_diffusivity'],flow,dp)
            water.add_model(propname='pore.diffusivity',model=op.models.mi
sc.constant, value=water['pore.S_E_diffusivity'])
            water.add_model(propname='throat.diffusivity',model=op.models.
misc.constant, value=water['throat.TD_diffusivity'])
```

```
C:\Users\Stephen PC\Anaconda3\lib\site-packages\openpnm\core\ModelsMixin.p
y:339: FutureWarning: elementwise comparison failed; returning scalar inst
ead, but in the future will perform elementwise comparison
  if propname in kwargs.values():  # Prevent infinite loops of look-ups
```

In [5]:

```
water['pore.diffusivity']=np.average(water['throat.diffusivity'])
phys_water.regenerate_models()
```

In [6]:

```
t_final = 1.5*np.sum(geo['pore.region_volume'])/sf.rate(throats=inlet_thro
ats,mode='group')[0]
if t_final > 1000:
    dt = np.round(t_final/num_steps,0)
elif t_final >10:
    dt = np.round(t_final/num_steps,2)
else:
    dt = np.round(t_final/num_steps,4)

t_final = dt*num_steps
```

```
ad = op.algorithms.TransientAdvectionDiffusion(network=pn, phase=water)
ad.set_value_BC(pores=inlet_pores, values=inj_conc)
ad.set_outflow_BC(pores=outlet_pores)

ad.settings['t_scheme'] = 'implicit'
ad.settings['t_output'] = 0.01
ad.settings['t_tolerance'] = 0.0
ad.settings['store_rate'] = False
ad.settings['t_initial'] = 0.0
ad.settings['t_final'] = dt
ad.settings['t_step'] = dt
ad.settings['t_precision'] = 4

C0 = C = np.zeros(pn.Np)
ad.set_IC(values=C)
total_eluted = [0.0]
total_injected = [0.0]
td_steps = [0.0]
solute_mass = [0.0]
total_adsorbed = [0.0]
inlet_rate = [0.0]
outlet_rate = [0.0]
outlet_rate2 = [0.0]
inj_mass = 0
```

In [7]:

```
for i in range(1, num_steps, 1):

    ad.run()

    td_steps.append(td_steps[-1]+dt)
    inj_mass = ad.rate(pores=inlet_pores,mode='group')*dt
    total_injected.append(total_injected[-1]+inj_mass[0])
    eluted = ad.rate(pores=outlet_pores,mode='group')*dt*-1
    total_eluted.append(total_eluted[-1]+eluted)

    inlet_rate.append(ad.rate(throats=inlet_throats,mode='group')[0])
    outlet_rate2.append(ad.rate(pores=outlet_pores,mode='group')[0]*-1)


    solute_mass.append(np.sum(ad['pore.concentration'][body_pores]*geo['po
re.volume'][body_pores]))
```

```
    C = ad['pore.concentration']
    ad.set_IC(values=C)
    ad.settings['t_initial'] = 0.0

C_ave = outlet_rate2/sf.rate(throats=inlet_throats,mode='group')[0]
Pe_particle = darcy_velocity*Dc/(porosity*D_SE)
Pe = Pe_particle
```

Here the curve fitting is performed

In [8]:

```
def elution_DL_v(step,DL,v):
    x = L
    el1 = 0.5*(special.erfc((x-step*v)/(2*(DL*step)**(1/2))))
    el2 = 0.5*np.exp(v*x/DL)
    el3 = special.erfc((x+step*v)/(2*(DL*step)**(1/2)))
    return inj_conc*(el1+el2*el3)

g = [5e-5,darcy_velocity]
xdata = [float(x) for x in td_steps]
ydata = [float(x) for x in C_ave]
popt, pcov = curve_fit(elution_DL_v,xdata,ydata,p0=g)
disp_coeff = popt[0]
```

In [9]:

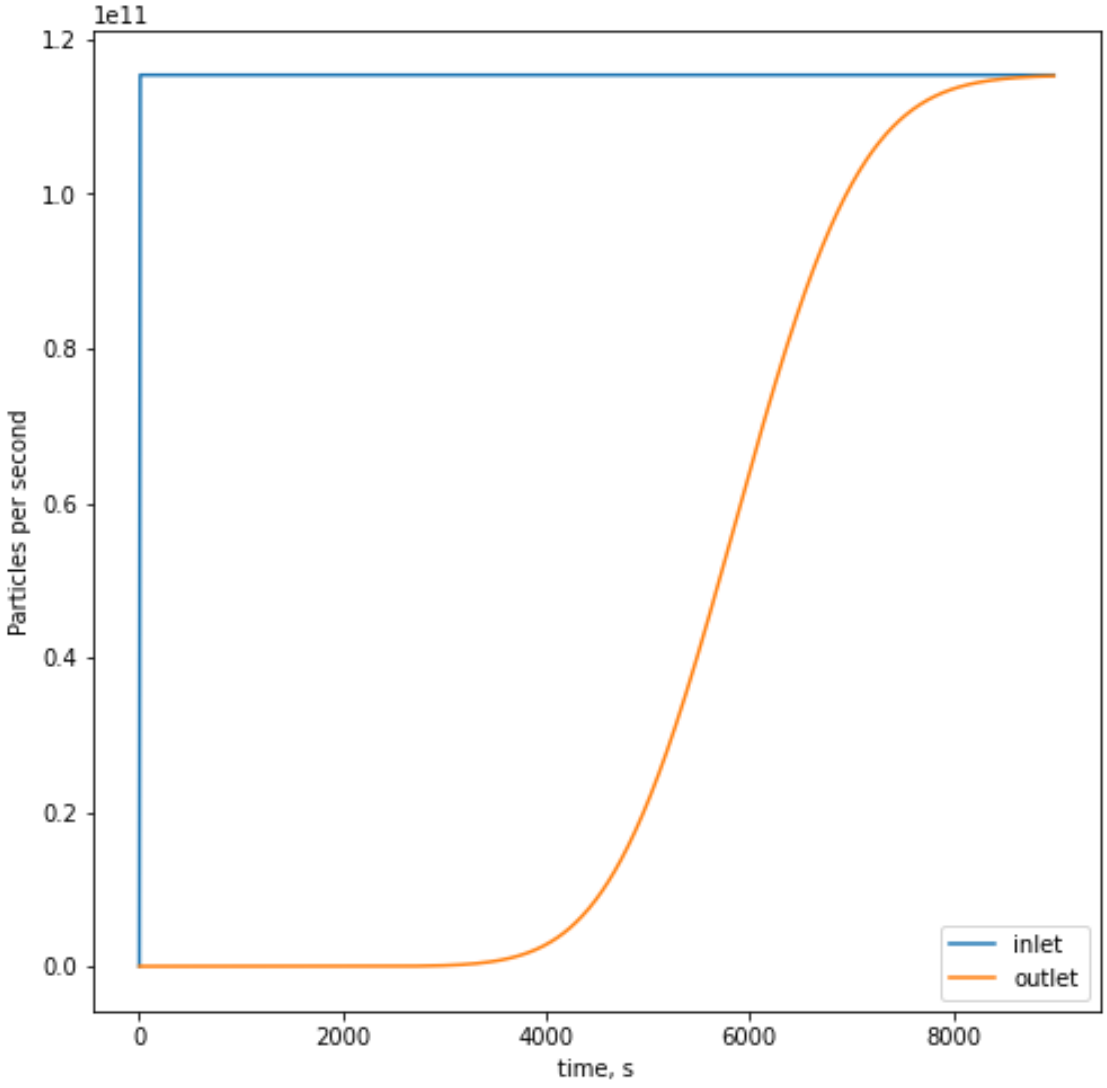
```
#DuPlessis tortusity
tortuosity = porosity/(1-(1-porosity)**(2/3))
D_eff_ave = (np.sum(water['pore.diffusivity']*geo['pore.volume'])+np.sum(w
ater['throat.diffusivity']*geo['throat.volume']))/(np.sum(geo['pore.volume
'])+np.sum(geo['throat.volume']))

sf = op.algorithms.StokesFlow(network=pn, phase=water)
sf.set_value_BC(pores=inlet_pores, values=del_P)
sf.set_value_BC(pores=outlet_pores, values=0.0)
sf.run();
darcy_velocity_2 = np.sum(sf.rate(throats=pn.Ts,mode='single')[inlet_throa
ts])/A

print('Particle Size = '+str(dp)+ ' m')
print('Darcy Velocity = '+str(darcy_velocity)+' m/s')
print('Dispersion Coefficient 2 Variables = '+str(disp_coeff)+' m^2/s')
print('fitted interstinial velocity = '+str(popt[1])+' m/s')
print('estimated interstinial velocity = '+str(darcy_velocity/porosity)+'
m/s')
```

```
print('Particle Peclet Number = '+str(Pe))
print('Average Effective Diffusion = '+str(D_eff_ave)+' m^2/s')
print('Tortuosity = '+str(tortuosity))
print('Particle Darcy Velocity = '+str(darcy_velocity_2)+' m/s')
el = [0 for i in range(len(ydata))]
for i in range(len(ydata)):
    el[i] = elution_DL_v(xdata[i],popt[0],popt[1])
```

```
Particle Size = 2e-08 m
Darcy Velocity = 1e-05 m/s
Dispersion Coefficient 2 Variables = 5.22299185645705e-08 m^2/s
fitted interstinial velocity = 2.541211791951999e-05 m/s
estimated interstinial velocity = 2.5268567845686274e-05 m/s
Particle Peclet Number = 2052.319635521078
Average Effective Diffusion = 2.443734214700465e-11 m^2/s
Tortuosity = 1.387302120810477
Particle Darcy Velocity = 1.0000000000017919e-05 m/s
```

In [10]:

```
plt.figure(figsize=(8, 8));
plt.plot(td_steps,inlet_rate,td_steps,outlet_rate2);
plt.gca().legend(('inlet','outlet'));
plt.xlabel('time, s');
plt.ylabel('Particles per second');
```


1e11
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Particles per second
0
2000
4000
6000
8000
time, s
inlet
outlet

Appendix C

2-Dimensional Invasion Percolation and Adsorption

Here a sample 2-d 50x50 model is presented. This model is useful for visualizing phenomena and testing the physics of adsorption, but it does not represent a realisitc porous media.

In [1]:

```
import openpnm as op
import numpy as np
import matplotlib.pyplot as plt
from ipywidgets import interact, IntSlider
np.random.seed(32)
np.set_printoptions(precision=5)
np.set_printoptions(threshold=np.inf)
from scipy.constants import pi
import Stephen_Functions as SF
import Dispersion_Functions as DF
```

In [2]:

```
# Variables

dp = 20.0e-9 # Colloid diameter, m
dt = 200 # Time Step, s
phib_kbT = 8 # phi_b div by k_b*t, dimensionless
num_steps = 100
mass_inj = 20.0 # µg
particle_density = 1140 # kg/m^3 for EC
vol_frac = 0.001  # volume particles / volume injected
del_P = 1.0 # Pa

# The mass injected is defined in terms of mass. This value is used to calculate
the target_inj, which is defined
# in units of particles. The TransientAdvectionDiffusion algorthim is ran with th
e inj_conc as the concentration of the
# inlet pores. Once the total injected particles is greater than the target_inj t
he concentration of the inlet pores
# is then set to zero.

target_inj = mass_inj*1e-9/((4*pi*((dp/2)**3)/3)*particle_density) #particles
inj_conc = vol_frac/((pi/6)*dp**3) # particles/m^3
HD = True
SE = True
```

In [3]:

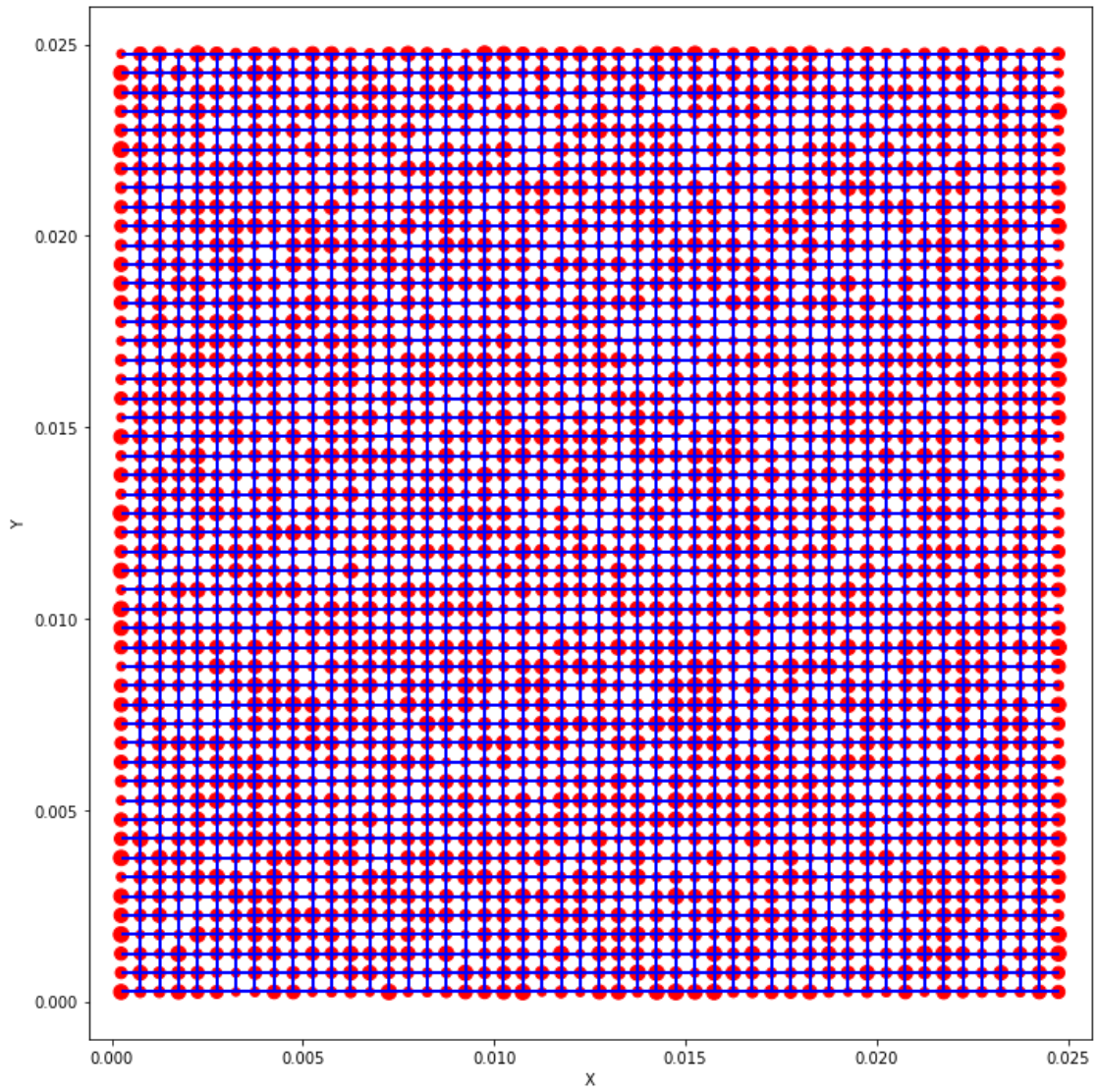
```
pn = op.network.Cubic(shape=[50,50],spacing=5e-4)
geo = op.geometry.StickAndBall(network=pn, pores=pn['pore.all'], throats=pn['thro
at.all'])
ip_inlet_pores = [1673,500,1438,1921,2264,894,232,1368,193,1923]
geo.add_model(propname='pore.surface_area', model=SF.pore_surface_area)

#remove throats connecting inlet and outlet pores
T_f = pn.find_neighbor_throats(pores=pn.pores(['left']), mode='xnor')
op.topotools.trim(throats=T_f, network=pn)
T_b = pn.find_neighbor_throats(pores=pn.pores(['right']), mode='xnor')
op.topotools.trim(throats=T_b, network=pn)

inlet_pores = pn.pores(['left'])
outlet_pores = pn.pores(['right'])
body_pores = np.delete(pn.Ps,[inlet_pores,outlet_pores])
inlet_throats = pn.find_neighbor_throats(pores=inlet_pores, mode='xor')
outlet_throats = pn.find_neighbor_throats(pores=outlet_pores, mode='xor')

water = op.phases.Water(network=pn)
phys_water = op.physics.Standard(network=pn, phase=water, geometry=geo)

# The thermophysical properties of the oil phase are not defined in OpenPNM and m
ust be specified. Most of these properties
# are not necessary to the invasion percolation algorthim

oil = op.phases.GenericPhase(network=pn)
oil['pore.molecular_mass'] = oil['throat.molecular_mass'] = 120.0  # g/mol
oil['pore.viscosity'] = oil['throat.viscosity'] = 1.6  # Pa*s
oil['pore.temperature'] =  298 # K
oil['pore.surface_tension'] = lamda_0 = 0.0720
oil['pore.contact_angle'] = contact_angle = 89
oil['pore.diffusivity'] = 0
oil['pore.thermal_conductivity'] =0
oil['pore.electrical_conductivity'] =0
oil['pore.molecular_weight'] = 100

phys_oil = op.physics.Standard(network=pn, phase=oil, geometry=geo)
ip = op.algorithms.InvasionPercolation(network=pn, phase=oil)
ip.set_inlets(pores=ip_inlet_pores)
ip.run()

# It is necessary to fill trapped water pores because the way that the diffusivit
y physics is defined colloids may
# diffusive into one oil pore and if that oil pore is connected to a trapped wate
r pore the colloid may then diffusive
# into the trapped water
```

```
oil.update(ip.results(Snwp=0.1))
mask = ~oil['pore.occupancy']
water_pores = op.topotools.find_clusters(network=pn,mask=mask)
water_pores = water_pores[0]>0
oil['pore.occupancy'] += water_pores

water['throat.no_flow'] = pn.tomask(throats=pn.find_neighbor_throats(pores=pn.Ps[
oil['pore.occupancy']]))

oil['throat.occupancy'] = oil['pore.occupancy'][pn['throat.conns']][:,0]*oil['por
e.occupancy'][pn['throat.conns']][:,1]

oil['pore.surface_occupancy'] = pn.tomask(pn.find_connected_pores(water['throat.n
o_flow'] ^ oil['throat.occupancy'],
                                                                  flatten=True))
* oil['pore.occupancy']
# water['throat.no_flow'] refers to throats containing either oil or water that i
s connected to a oil pore. Since there is
# no advection through an oil pore there must be no advection through a throat co
nnected to an oil pore as well.

water['throat.no_flow'] = pn.tomask(throats=pn.find_neighbor_throats(pores=pn.Ps[
oil['pore.occupancy']]))

# The InvasionPerculation algorthim sometimes allows water throats that has oil p
ores connected on both ends.
# this line finds these throats and changes them to oil throats.

oil['throat.occupancy'] = oil['pore.occupancy'][pn['throat.conns']][:,0]*oil['por
e.occupancy'][pn['throat.conns']][:,1]

# The oil pores which are connected to water throats is useful to know, but is no
t used in the advection diffusion algorthim.
# These pores are the only ones which the colloids can attach to, thus solving th
em now can be used to determine
# the maximum possible adsorption achieved.

oil['pore.surface_occupancy'] = pn.tomask(pn.find_connected_pores(water['throat.n
o_flow'] ^ oil['throat.occupancy'],
                                                                  flatten=True))
* oil['pore.occupancy']

# Here the hydraulic conductance of all oil pores and water['throat.no_flow'] thr
oats is multiplied by a value that
# makes the value effectivly zero. Next, the diffusive conductances of all oil po
res and throats are also multipied by
# a value that makes them effectivly zero. In previous models these values were s
imply set to 1e-50 instead of
```

```
# creating a model that multiples these throats by a value that makes them effect
ivly zero. This new method is done
# because there were issues that arose when the physics models were regenerated w
hen preforming the advective diffusive
# algorthim.

phys_water['throat.mask'] = oil['throat.occupancy']*1e-50 + np.invert(oil['throat
.occupancy'])*1.0
phys_water['throat.mask2'] = water['throat.no_flow']*1e-50 + np.invert(water['thr
oat.no_flow'])*1.0
phys_water['throat.diffusive_conductance_original'] = phys_water['throat.diffusiv
e_conductance']
phys_water['throat.hydraulic_conductance_original'] = phys_water['throat.hydrauli
c_conductance']
phys_water.add_model(propname='throat.diffusive_conductance',
                     model=op.models.misc.product, prop1='throat.diffusive_conduc
tance_original', prop2='throat.mask')
phys_water.add_model(propname='throat.hydraulic_conductance',
                     model=op.models.misc.product, prop1='throat.hydraulic_conduc
tance_original', prop2='throat.mask2')

------------------------------------------------------------
CRITICAL   : front and back labels have been switched to obey the right-hand rule
SOURCE     : openpnm.network.Cubic.__init__
TIME STAMP : 2022-07-18 02:45:14,209
------------------------------------------------------------
```

In [4]:

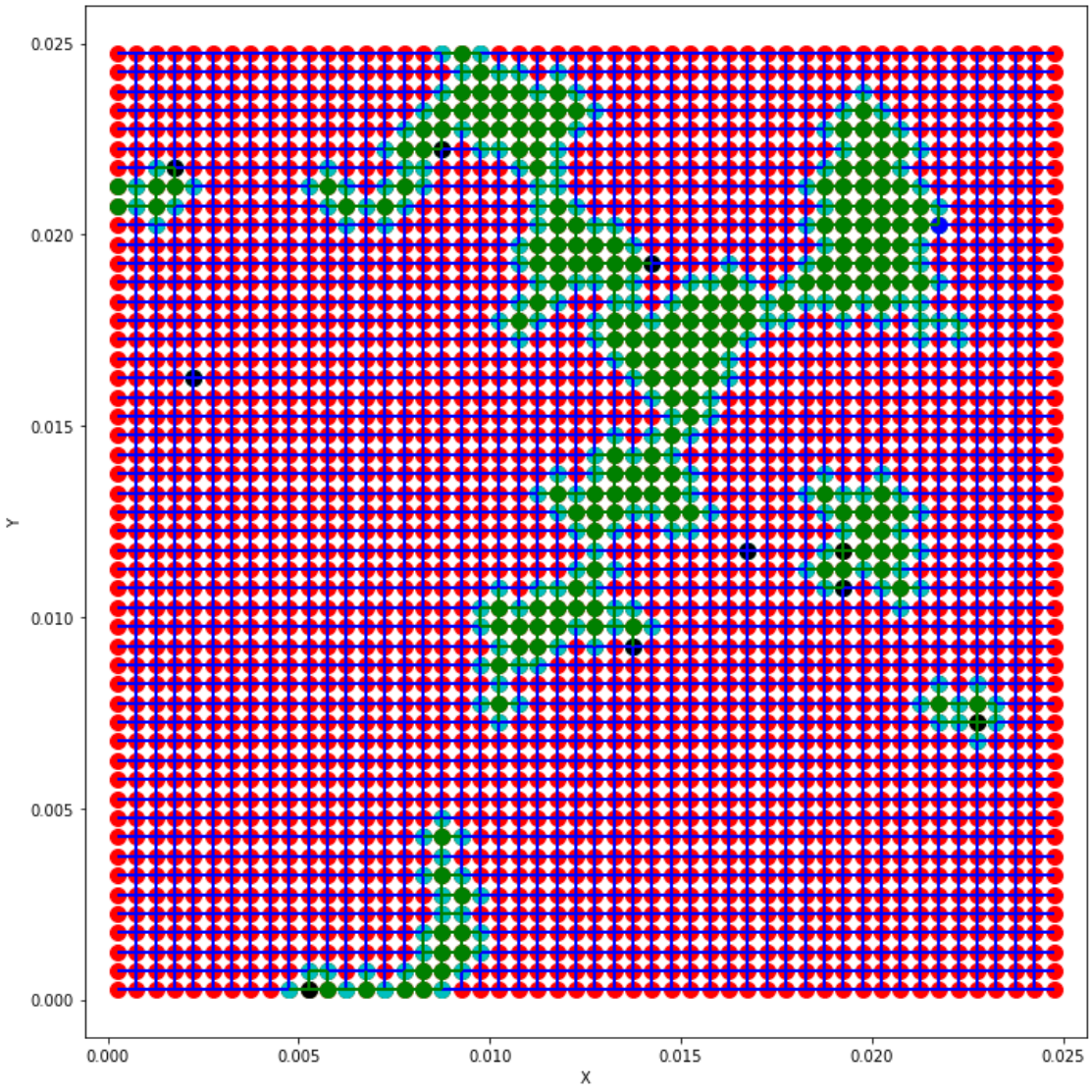
```
fig, ax = plt.subplots(figsize=[10, 10])
fig = op.topotools.plot_coordinates(network=pn,ax=ax,size_by=geo['pore.diameter']
, markersize=100)
fig = op.topotools.plot_connections(network=pn,ax=ax, linewidth=2)
```



In [5]:

```
fig, ax = plt.subplots(figsize=[10, 10])
fig = op.topotools.plot_connections(network=pn,ax=ax, linewidth=2)
fig = op.topotools.plot_connections(network=pn,ax=ax, throats=oil['throat.occupan
cy'], color='green', linewidth=2)

fig = op.topotools.plot_coordinates(network=pn,ax=ax, markersize=100)
fig = op.topotools.plot_coordinates(network=pn,ax=ax, pores=oil['pore.occupancy']
, color='green', markersize=100)
fig = op.topotools.plot_coordinates(network=pn,ax=ax, pores=oil['pore.surface_occ
upancy'], color='c', markersize=100)
fig = op.topotools.plot_coordinates(network=pn,ax=ax, pores=ip_inlet_pores, color
```

```
='k', markersize=100)
fig = op.topotools.plot_coordinates(network=pn,ax=ax, pores=2190, color='b', mark
ersize=100)
```



In [6]:

```
# Here the StokesFlow algortham is run. In the 2d model the pressure gradient is
the manulipated variable. However, in
# the packed cylinder model the flow rate will be the manulipated variable instea
d. This is because defining the permiablity
# of a 2d network is unclear as the cross-sectional area cannot be defined.

sf = op.algorithms.StokesFlow(network=pn, phase=water,)
```

```
sf.set_value_BC(pores=inlet_pores, values=del_P)
sf.set_value_BC(pores=outlet_pores, values=0.0)
sf.run();
flow = sf.rate(throats=pn.Ts,mode='single')
phi = dp/geo['throat.diameter']
```

In [7]:

```
water['throat.S_E_diffusivity'] = DF.Stokes_Einstein(water['throat.temperature'],
water['throat.viscosity'],dp)
water['pore.S_E_diffusivity'] = DF.Stokes_Einstein(water['pore.temperature'],wate
r['pore.viscosity'],dp)
D_SE = SF.Stokes_Einstein(water['pore.temperature'][0],water['pore.viscosity'][0]
,dp)

if HD == True:
    water['throat.HD_diffusivity'] = DF.Throat_Hindered_Diffusion(geo['throat.dia
meter'],water['throat.S_E_diffusivity'],dp)
    water['pore.HD_diffusivity'] = DF.Pore_Hindered_Diffusion(geo['pore.diameter'
],water['pore.S_E_diffusivity'],dp)
    if SE == True:
        water['throat.HD_SE_diffusivity'] = DF.Throat_Size_Exculsion_Dispersivity
(geo['throat.diameter'],water['throat.HD_diffusivity'],flow,dp)
        water.add_model(propname='throat.diffusivity',model=op.models.misc.consta
nt, value=water['throat.HD_SE_diffusivity'])
        water.add_model(propname='pore.diffusivity',model=op.models.misc.constant
, value=water['pore.HD_diffusivity'])
        phys_water['throat.hydraulic_conductance_old'] = phys_water['throat.hydra
ulic_conductance']
        phys_water['throat.mask3'] = (1+phi-(1/4)*phi**2)
        phys_water.add_model(propname='throat.hydraulic_conductance',
             model=op.models.misc.product, prop1='throat.hydraulic_conductance_ol
d',prop2='throat.mask2', prop3='throat.mask3')
    else:
        water['throat.HD_TD_diffusivity'] =DF.Taylor_Dispersivity(geo['throat.dia
meter'],water['throat.HD_diffusivity'],flow,dp)
        water.add_model(propname='throat.diffusivity', model=op.models.misc.const
ant,value=water['throat.HD_TD_diffusivity'])
        water.add_model(propname='pore.diffusivity',model=op.models.misc.constant
, value=water['pore.HD_diffusivity'])

else:
    if SE == True:
        water['throat.SE_diffusivity'] = DF.Throat_Size_Exculsion_Dispersivity(ge
o['throat.diameter'],water['throat.S_E_diffusivity'],flow,dp)
        water.add_model(propname='throat.diffusivity',model=op.models.misc.consta
nt, value=water['throat.SE_diffusivity'])
        phys_water['throat.hydraulic_conductance_old'] = phys_water['throat.hydra
ulic_conductance']
```

```
        phys_water['throat.mask3'] = (1+phi-(1/4)*phi**2)
        phys_water.add_model(propname='throat.hydraulic_conductance',
             model=op.models.misc.product, prop1='throat.hydraulic_conductance_ol
d',prop2='throat.mask2', prop3='throat.mask3')
        water.add_model(propname='pore.diffusivity',model=op.models.misc.constant
, value=water['pore.S_E_diffusivity'])
    else:
        water['throat.TD_diffusivity'] = DF.Taylor_Dispersivity(geo['throat.diame
ter'],water['throat.S_E_diffusivity'],flow,dp)
        water.add_model(propname='pore.diffusivity',model=op.models.misc.constant
, value=water['pore.S_E_diffusivity'])
        water.add_model(propname='throat.diffusivity',model=op.models.misc.consta
nt, value=water['throat.TD_diffusivity'])
```

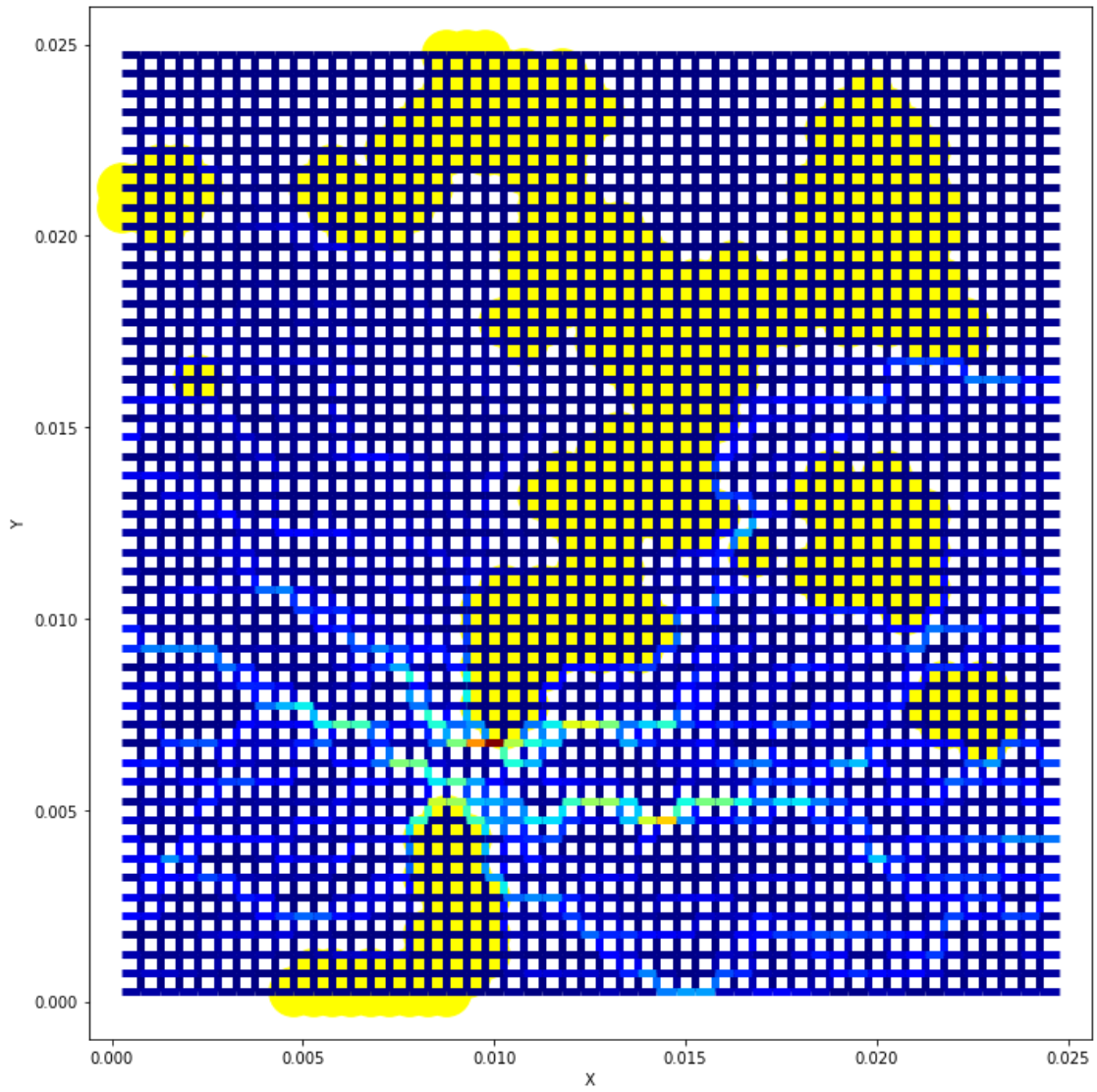
```
C:\Users\Stephen PC\Anaconda3\lib\site-packages\openpnm\core\ModelsMixin.py:339:
FutureWarning: elementwise comparison failed; returning scalar instead, but in th
e future will perform elementwise comparison
  if propname in kwargs.values():  # Prevent infinite loops of look-ups
```

In [8]:

```
dif_con_mod = op.models.physics.diffusive_conductance.ordinary_diffusion
phys_water.add_model(propname='throat.diffusive_conductance_modified', model=dif_
con_mod)
phys_water.add_model(propname='throat.diffusive_conductance', model=op.models.mis
c.product,
                     prop1='throat.diffusive_conductance_modified', prop2='throat
.mask')
mod = op.models.physics.ad_dif_conductance.ad_dif
phys_water.add_model(propname='throat.ad_dif_conductance', model=mod, s_scheme='p
owerlaw')
phys_water.regenerate_models('throat.ad_dif_conductance')
water.regenerate_models()
phys_water.regenerate_models()
```

In [9]:

```
fig, ax = plt.subplots(figsize=[10, 10])
fig = op.topotools.plot_connections(network=pn,ax=ax,color_by=flow,linewidth=5);
fig = op.topotools.plot_coordinates(network=pn,ax=ax, pores=oil['pore.occupancy']
, color='yellow', markersize=1000);
```



In [10]:

```
# The advection diffusion algorthim is created here

ad = op.algorithms.TransientAdvectionDiffusion(network=pn, phase=water)

# models for the total amount of particles adsorbed, the surface area which the p
articles can be adosrbed to,
# and the pore volume are created here.

phys_water['pore.adsorbed'] = np.zeros(pn.Np)
phys_water['pore.surface_area'] = geo['pore.surface_area']
phys_water['pore.volume'] = geo['pore.volume']
```

```
# The value for the blocking function is determined here, This will be updated at
each timestep.

phys_water.add_model(propname='pore.B',
               model=SF.Blocking,area='pore.surface_area',
               adsorbed='pore.adsorbed',theta_max=0.91,r=dp/2,m=3,alg_name=ad)

# The reaction coefficient is defined here

k = -2.32*((D_SE/dp)*(phib_kbT/pi)**0.5)*np.exp(-phib_kbT) #m/s, 2.32 is carried
over from the blocking function
water['pore.concentration'] = 0.0

# The model 'pore.reaction_coeff' is created which is the product of the previous
ly defined reaction coefficient,
# the surface area of the pore, and the blocking function. The final value for th
e reaction that takes place is defined
# as a product of the pore concentration and the 'pore.reaction_coeff'.

phys_water.add_model(propname='pore.reaction_coeff',model=SF.reaction_coefficient
,
                     surface_area='pore.surface_area',reaction_constant=k,B='pore.
B')
phys_water.add_model(propname='pore.rxn',model=op.models.physics.source_terms.pow
er_law,
                    A1='pore.reaction_coeff', X='pore.concentration')
phys_water.regenerate_models()
```

In [11]:

```
# Here the settings for the TransientAdvectionDiffusion algorthim is defined. The
length of one step is the same as the
# final step, because the reaction is put in a for loop where the final step is i
ncreased by dt for every step in the
# for loop. This is done intil num_steps is reached.

ad.set_value_BC(pores=inlet_pores, values=inj_conc)
ad.set_outflow_BC(pores=outlet_pores)
phys_water.models['pore.B']['alg_name'] = ad.name
ad.settings['t_scheme'] = 'implicit'
ad.settings['t_output'] = 0.01
ad.settings['t_tolerance'] = 0.0
ad.settings['store_rate'] = False
ad.settings['t_final'] = dt
ad.settings['t_step'] = dt
ad.settings['t_precision'] = 4
ps = oil['pore.occupancy']*np.isnan(ad['pore.bc_value'])*np.isnan(ad['pore.bc_out
flow'])
```

```
# The locations where the reaction takes place is defined here as the oil pores.

ad.set_source(propname='pore.rxn', pores=ps)

# These values represent various lists for values at each timestep. The value at
each time step is appended to the list
# while the algorthim is being ran.

C0 = C = np.zeros(pn.Np)
ad.set_IC(values=C)
total_eluted = [0.0]
total_injected = [0.0]
td_steps = [0.0]
solute_mass = [0.0]
total_adsorbed = [0.0]
inlet_rate = [0.0]
outlet_rate = [0.0]

# These values are used in the adsorption reaction equations. They are updated at
each time step.

ad['pore.adsorbed@0'] = np.zeros(pn.Np)
ad['pore.theta@0'] = np.zeros(pn.Np)
#Pe = water['throat.peclet.ad']
inj_mass = 0
```

In [12]:

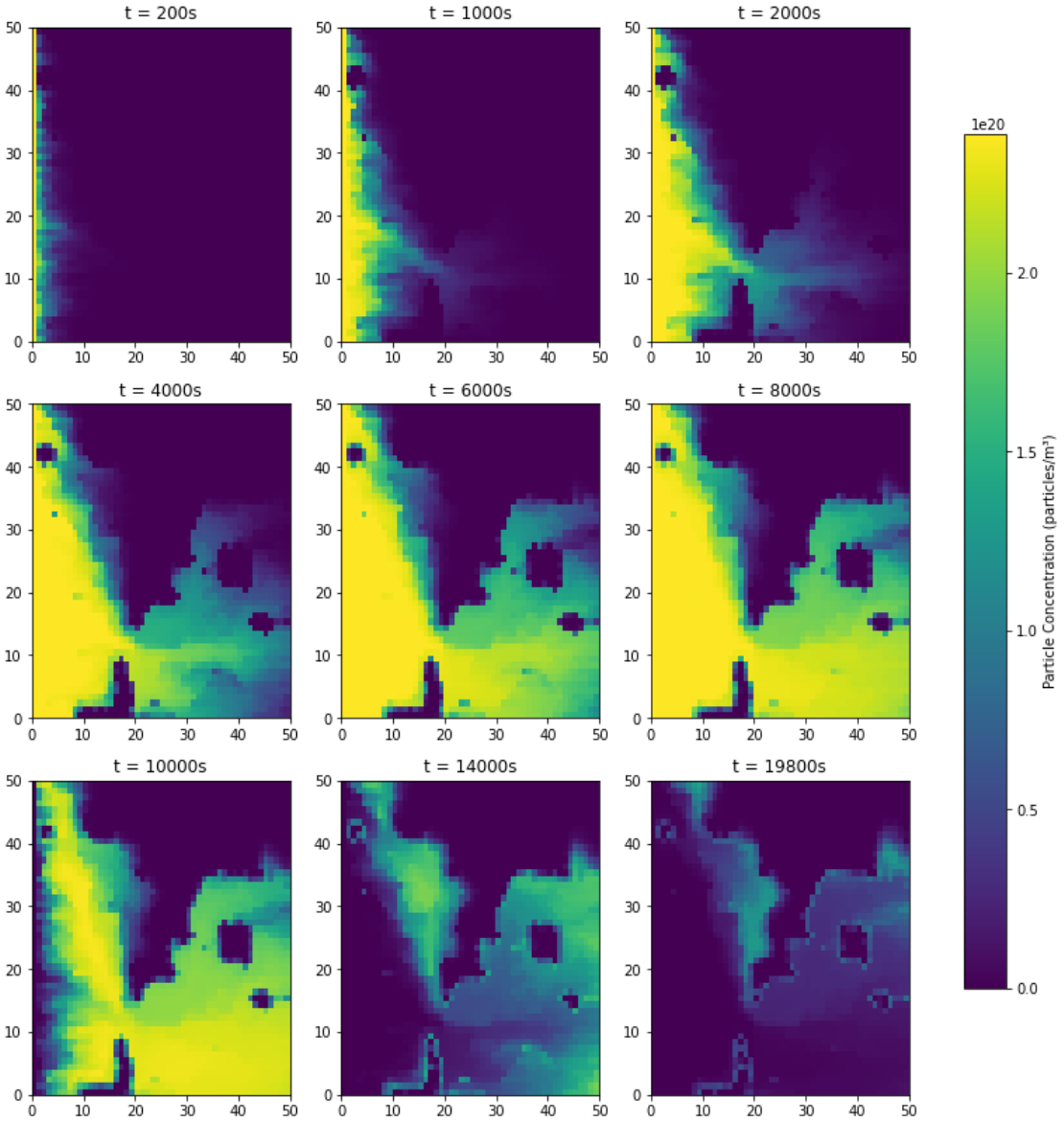
```
# Finally, the TransientAdvectionDiffusion algorthim is run. This is done by runn
ing one step and then updating values
# that are used to calculate the reaction coefficient.

for i in range(1, num_steps, 1):
    # If the total amount of injected particles is greater than the target inject
ed is checked here. If it is greater
    # the inlet concentration is set to zero.

    if target_inj < total_injected[-1]:
        ad.set_value_BC(pores=inlet_pores, values=0.0)

    ad.run()

    #if len(ad.settings['t_solns']) > 2:
    #   ad.settings['t_solns'].remove(ad.settings['t_solns'][-2])

    # Various values are now calculated here and appended to their respective lis
ts.
```

```
    td_steps.append(td_steps[-1]+dt)
    inj_mass = ad.rate(throats=inlet_throats,mode='group')*dt
    total_injected.append(total_injected[-1]+inj_mass[0])
    eluted = ad.rate(throats=outlet_throats,mode='group')[0]*dt
    total_eluted.append(total_eluted[-1]+eluted)
    inlet_rate.append(ad.rate(throats=inlet_throats,mode='group'))
    outlet_rate.append(ad.rate(throats=outlet_throats,mode='group'))
    solute_mass.append(np.sum(ad['pore.concentration'][body_pores]*geo['pore.volu
me'][body_pores]))

    # The particles adsorbed after the current time step is calculated, and then
the total amount of adsorbed particles
    # is calculated by adding this value to the total amount of adsorbed particle
s from the previous time step.

    dNdt = -1*phys_water['pore.rxn.rate']*ps
    phys_water['pore.adsorbed'] = ad['pore.adsorbed@'+str(dt*i)] = ad['pore.adsor
bed@'+str(dt*(i-1))] + dNdt*dt
    #phys_water['pore.adsorbed'] = ad['pore.adsorbed@'+str(dt*i)] = ad['pore.adso
rbed@'+str(ad.settings['t_solns'][-2])] + dNdt*dt
    total_adsorbed.append(np.sum(ad['pore.adsorbed@'+str(ad.settings['t_solns'][-
1])]))
    ad['pore.theta@'+str(ad.settings['t_solns'][-1])] = np.pi*(dp/2)**2*ad['pore.
adsorbed@'+str(ad.settings['t_solns'][-1])]/geo['pore.surface_area']


    phys_water.regenerate_models()

    # The inital and final timestep is updated for the next step, as well as the
inital concentration for the next step is
    # set as the final concentration for the current step.

    C = ad['pore.concentration']
    ad.set_IC(values=C)
    ad.settings['t_initial'] = i*dt
    ad.settings['t_final'] = (i+1.0)*dt
```

In [13]:

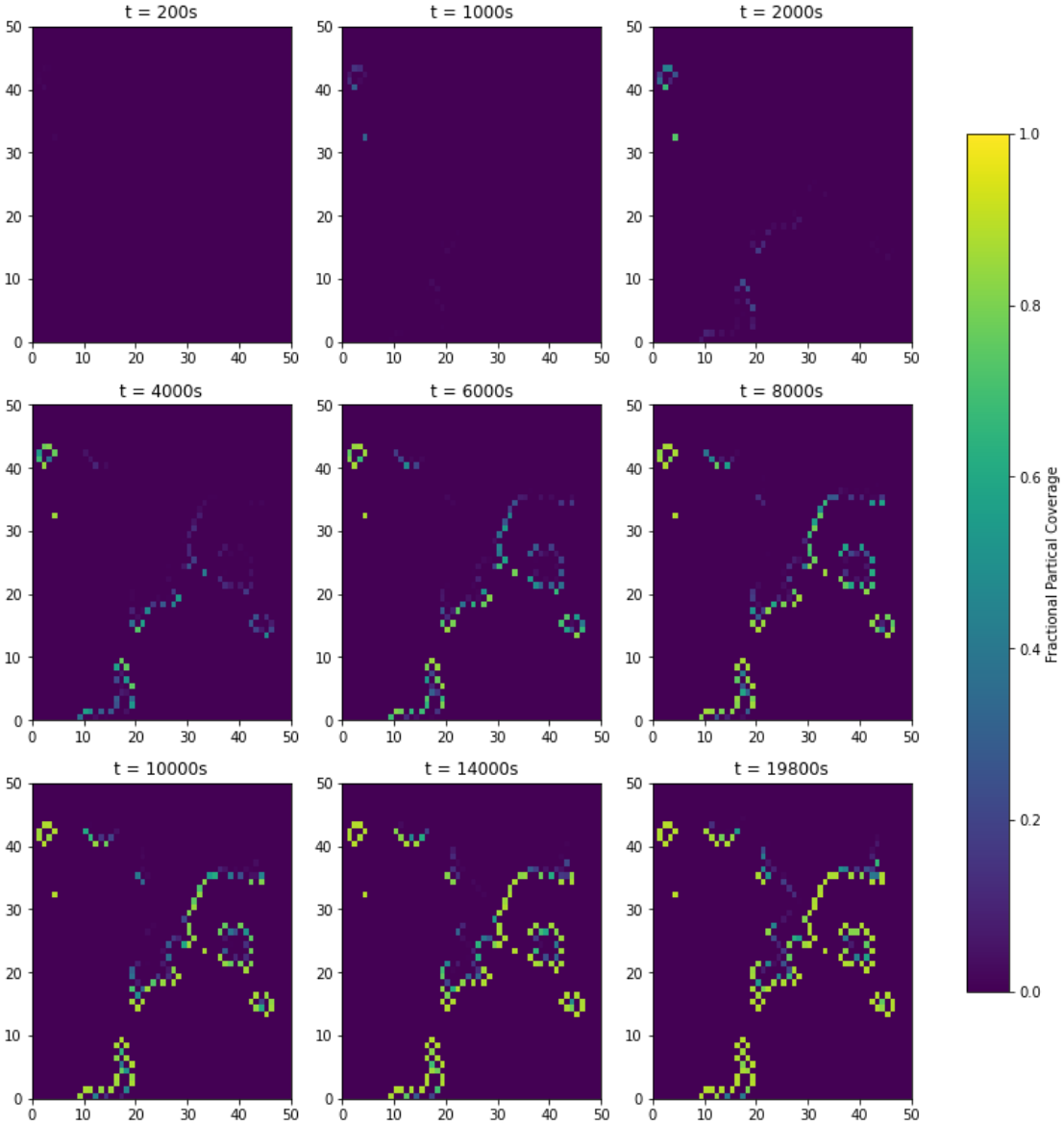
```
con_min = 0 # mol/m^3
con_max = inj_conc # mol/m^3
fig, axs = plt.subplots(3, 3,figsize=(14, 14));
plot_step = [[0,4,9],[19,29,39],[49,69,98]]
for col in range(3):
    for row in range(3):
        ax = axs[row, col]
        c = ad['pore.concentration@'+str(ad.settings['t_solns'][plot_step[row][co
l]])]
```

```
        cc = c.reshape(pn.shape[0],pn.shape[1])
        pcm = ax.pcolormesh(cc.T,vmin=con_min,vmax=con_max);
        ax.set_title('t = '+str(ad.settings['t_solns'][plot_step[row][col]])+'s')
;
fig.colorbar(pcm,ax=axs,label='Particle Concentration (particles/m\u00b3)',shrink
=0.8);
#fig.colorbar(pcm, ax=axs, shrink=0.8,location='right');
plt.show();
```



In [14]:

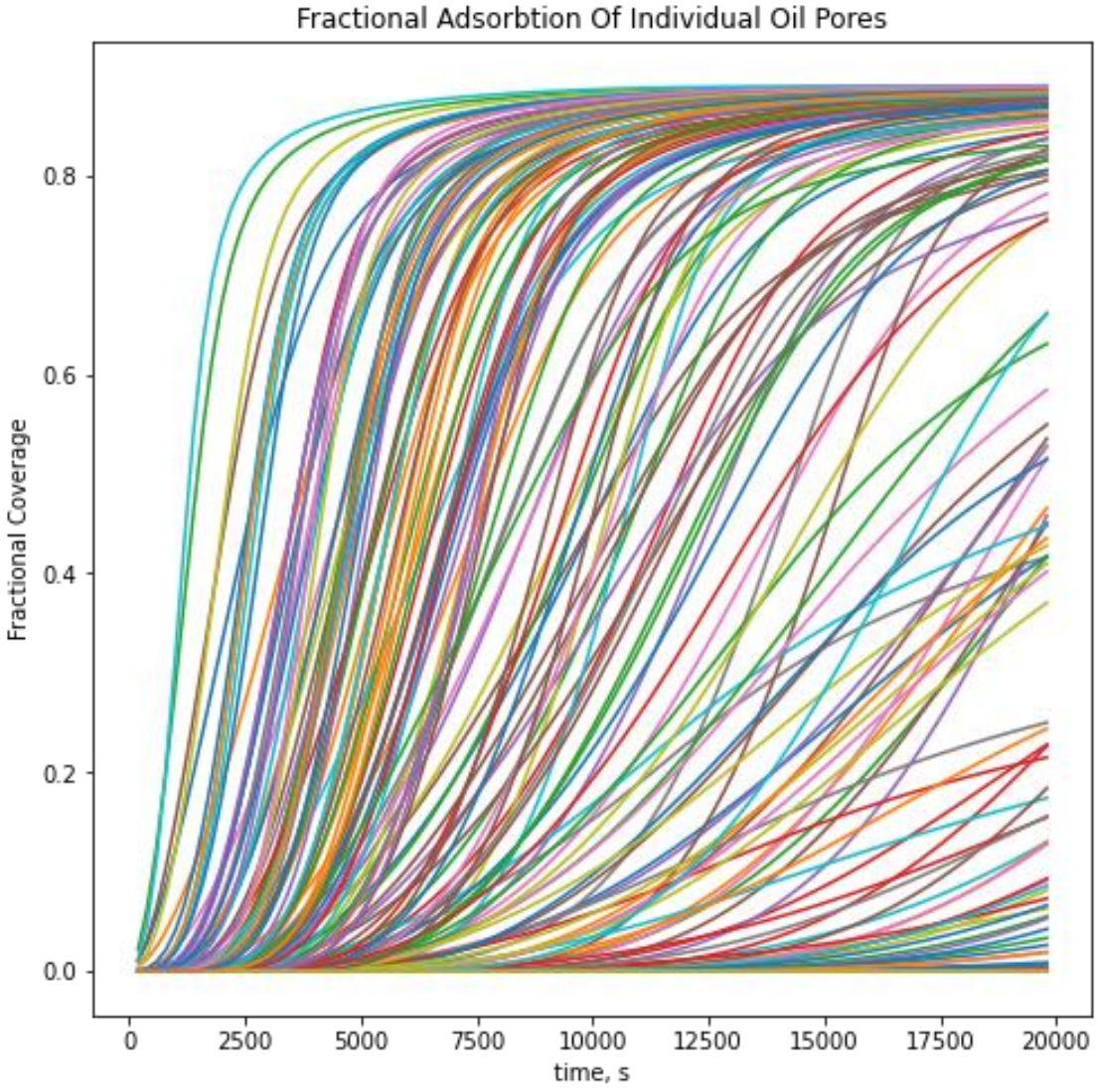
```
fig, axs = plt.subplots(3, 3,figsize=(14, 14));
plot_step = [[0,4,9],[19,29,39],[49,69,98]]
for col in range(3):
    for row in range(3):
        ax = axs[row, col]
        c = ad['pore.theta@'+str(ad.settings['t_solns'][plot_step[row][col]])]
        cc = c.reshape(pn.shape[0],pn.shape[1])
        pcm = ax.pcolormesh(cc.T,vmin=0,vmax=1);
        ax.set_title('t = '+str(ad.settings['t_solns'][plot_step[row][col]])+'s')
;
fig.colorbar(pcm,ax=axs,label='Fractional Partical Coverage',shrink=0.8);
plt.show();
```



In [15]:

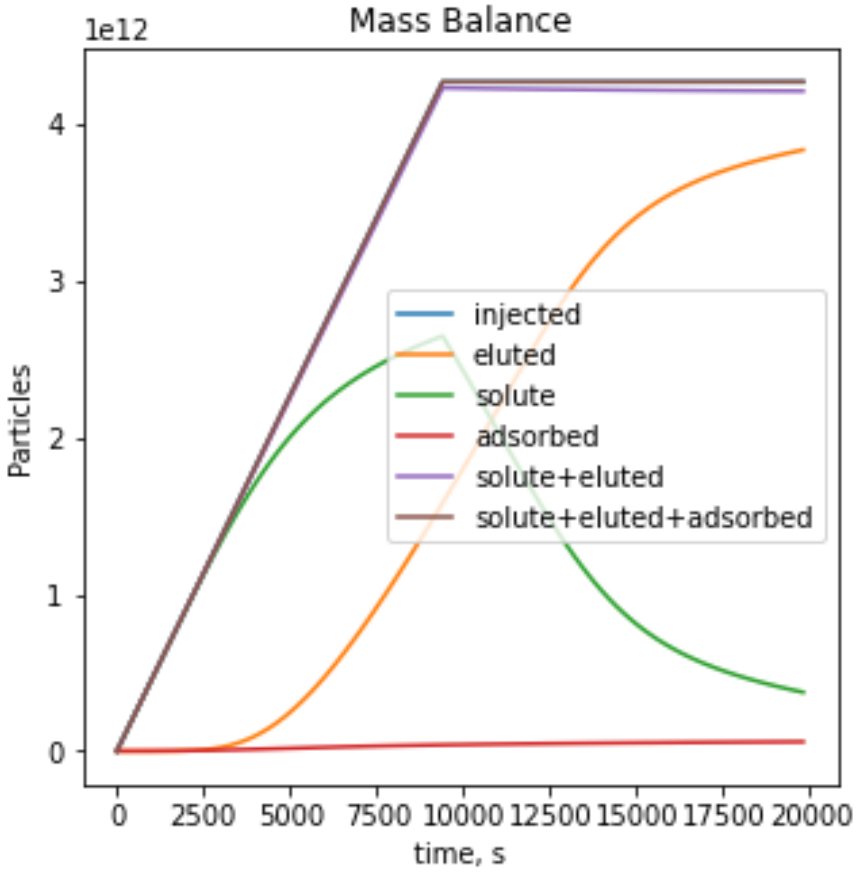
```
def plot_trace(alg, pores, propname='pore.theta'):
    vals = []
    times = []
    q = propname
    for t in alg.settings['t_solns']:
        try:
            vals.append(alg[f'{q}@' + t][pores])
            times.append(float(t))
        except:
```

```
            pass
    return times, vals

data = plot_trace(ad,ps)
plt.figure(figsize=(8,8))
plt.plot(*data);
plt.title('Fractional Adsorbtion Of Individual Oil Pores');
plt.ylabel('Fractional Coverage');
plt.xlabel('time, s');
```



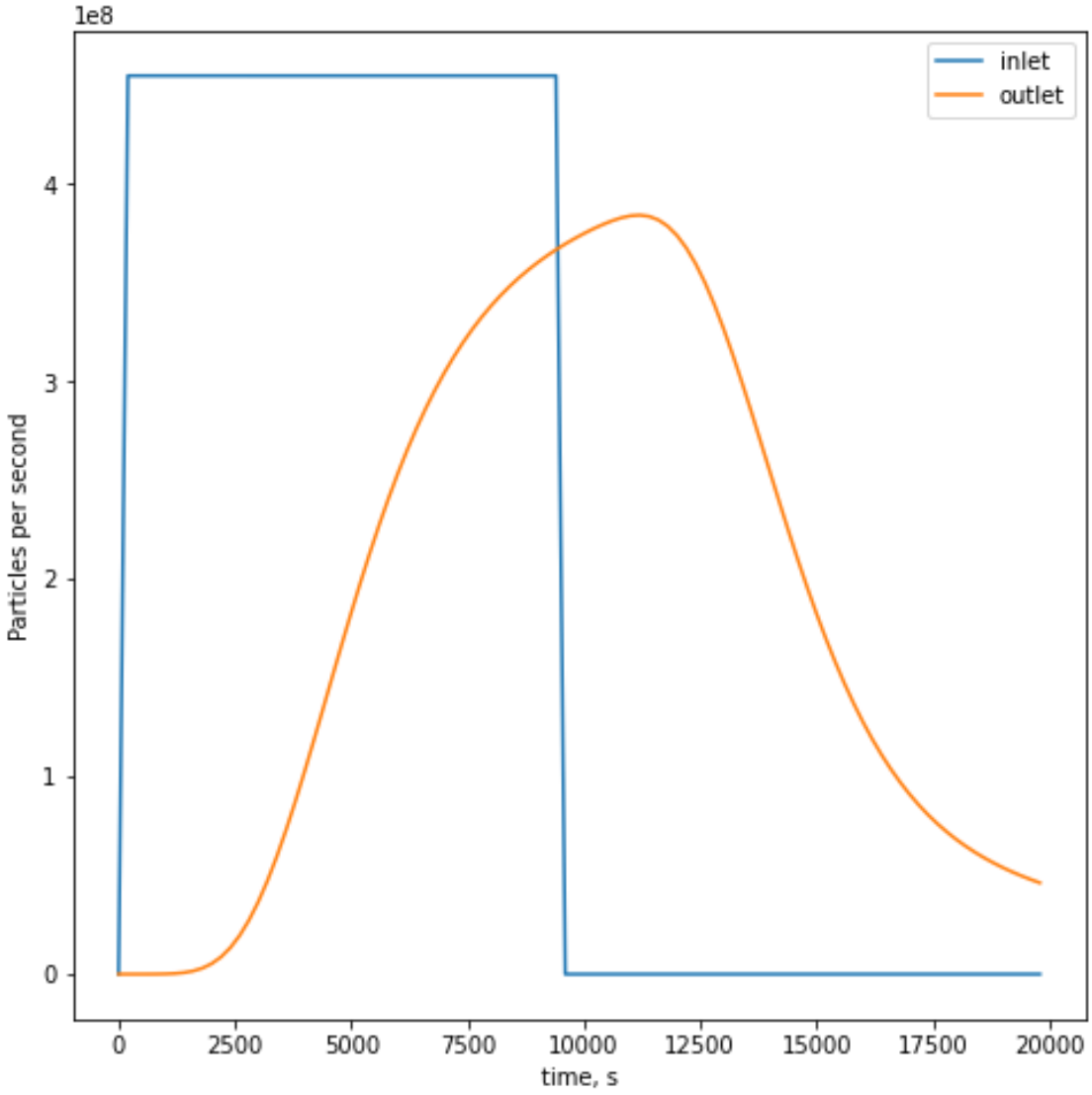
In [16]:

```
s_e = np.add(total_eluted,solute_mass)
s_e_a = np.add(s_e,total_adsorbed)

plt.figure(figsize=(5, 5));
plt.title('Mass Balance')
plt.plot(td_steps,total_injected,td_steps,total_eluted,td_steps,solute_mass
         ,td_steps,total_adsorbed,td_steps,s_e,td_steps,s_e_a)
plt.gca().legend(('injected','eluted','solute','adsorbed','solute+eluted','solute
+eluted+adsorbed'));
plt.xlabel('time, s');
plt.ylabel('Particles');
```



In [17]:

```
plt.figure(figsize=(8, 8));
plt.plot(td_steps,inlet_rate,td_steps,outlet_rate);
plt.gca().legend(('inlet','outlet'));
plt.xlabel('time, s');
plt.ylabel('Particles per second');
```



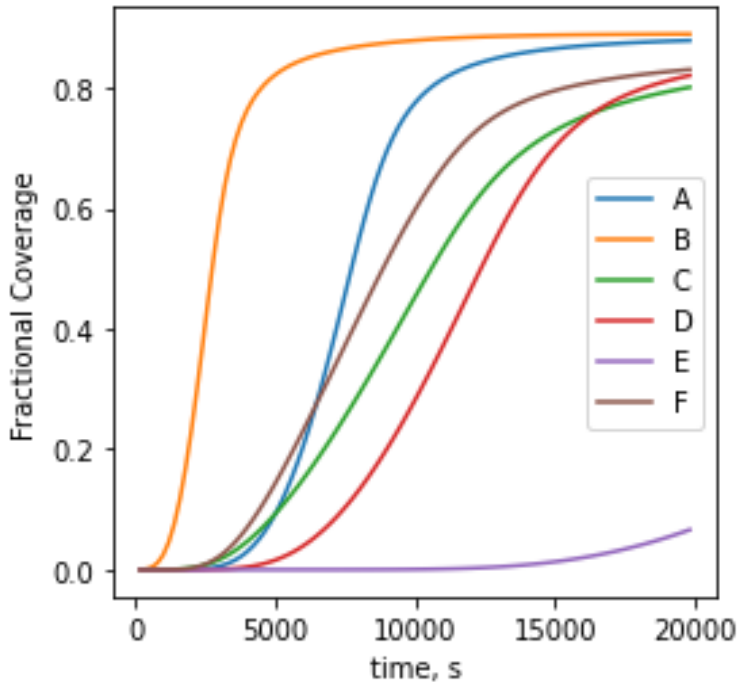
In [18]:

```
highlighted_pores=[740,859,901,1324,2190,2214]
m_highlighted_pores=pn.tomask(pores=highlighted_pores)
```

In [19]:

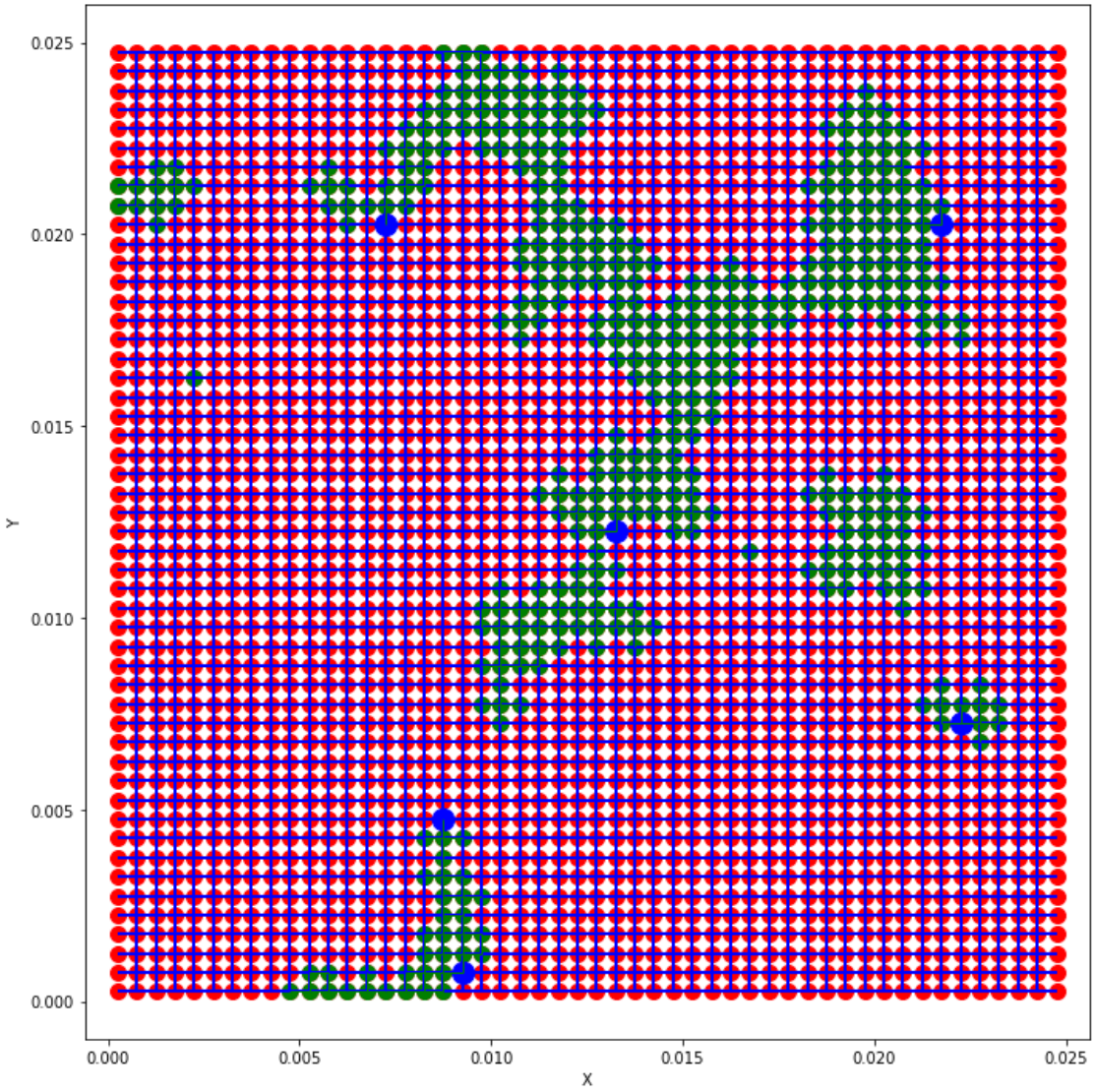
```
def plot_trace(alg, pores, propname='pore.theta'):
    vals = []
    times = []
    q = propname
    for t in alg.settings['t_solns']:
        try:
            vals.append(alg[f'{q}@' + t][pores])
```

```
            times.append(float(t))
        except:
            pass
    return times, vals

data = plot_trace(ad,m_highlighted_pores)
plt.figure(figsize=(4,4))
plt.plot(*data);
#plt.title('Fractional Adsorbtion Of Individual Oil Pores');
plt.ylabel('Fractional Coverage');
plt.xlabel('time, s');
plt.legend(['A','B','C','D','E','F']);
```



In [20]:

```
fig, ax = plt.subplots(figsize=[10, 10])
fig = op.topotools.plot_connections(network=pn,ax=ax, linewidth=2)
fig = op.topotools.plot_connections(network=pn,ax=ax, throats=oil['throat.occupan
cy'], color='green', linewidth=1)

fig = op.topotools.plot_coordinates(network=pn,ax=ax, markersize=100)
fig = op.topotools.plot_coordinates(network=pn,ax=ax, pores=oil['pore.occupancy']
, color='green', markersize=100)
fig = op.topotools.plot_coordinates(network=pn,ax=ax, pores=highlighted_pores, co
lor='b', markersize=200)
```


0.025
0.020
0.015
0.010
0.005
0.000
Y
0.000
0.005
0.010
0.015
0.020
0.025
X

Appendix D

Packed Column Invasion Percolation and Adsorption

Irreversible Adsorption in Packed Column

Simulation of invasion-percolation of a NWP (oil in this case) and adsorption of nanoparticles onto the water-NWP interface. Hindered diffusion and Size exclusion effects are removed from this example.

In [1]:

```
import openpnm as op
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(32)
np.set_printoptions(precision=5)
np.set_printoptions(threshold=np.inf)
from scipy.constants import pi
import Stephen_Functions as SF
import Dispersion_Functions as DF
```

Variables

This simulation will be for a 3 PV injection. If a continuous injection is desired remove the Pore_Volumes_Injected variable and adjust the TAD algorithm accordingly. The inputted concentration is defined in terms of volume fraction in this case, but no matter how it is defined it must be converted to (particles/m^3).

In [2]:

```
# Variables
excel_file_name = 'slow 3PV 20 nm 0_0001 vol frac v4 0_1snwp.xlsx'
dp = 20.0e-9 # Colloid diameter, m
phib_kbT = 8 # phi_b div by k_b*t, dimensionless
dt = 200 # Time Step, s
num_steps = 1500
Pore_Volumes_Injected = 3

vol_frac = 0.00001  # volume particles / volume injected
inj_conc = vol_frac/((pi/6)*dp**3) # particles/m^3

particle_density = 1140 # kg/m^3 for EC
darcy_velocity = 1.0e-6 # m/s
snwp = 0.1
```

Network and Geometry

A saved pore network is loaded from a previous SNOW2 extraction, and the topology(pn) and geometry (geo) is defined.

In [3]:

```
proj = op.io.PNM.load_project(filename='1to1_res_28vox_corrected.pnm')

pn = proj['net_01']
geo = proj['geo_01']

#these lines remove throats with negative length that crash simulations
h = pn.check_network_health()
op.topotools.trim(network=pn, pores=h['trim_pores'])

#define inlet and outlet pores and throats
inlet_pores = pn.pores(['zmin'])
outlet_pores = pn.pores(['zmax'])
inlet_throats = pn.find_neighbor_throats(pores=inlet_pores, mode='xor')
outlet_throats = pn.find_neighbor_throats(pores=outlet_pores, mode='xor')

# body_pores are all pores besides inlet and outlet pores
body_pores = [x for x in pn.Ps if x not in np.concatenate((inlet_pores,outlet_por
es))]

# Here the diameter and length is automatically calculated, Dc is the bead size
D = np.max(pn['pore.coords'][:,0:1])-np.min(pn['pore.coords'][:,0:1])
Dc = D*2e-3/(2.5e-2)
L = pn['pore.coords'][outlet_pores][0,2] - pn['pore.coords'][inlet_pores][0,2]
A = np.pi/4.0*(D)**2.0
porosity = np.sum(geo['pore.region_volume'])/(A*L)

#phase and physics for water
water = op.phases.Water(network=pn)
phys_water = op.physics.Basic(network=pn, phase=water, geometry=geo)
```

Invasion Percolation

Oil is selected as the NWP here, but to use oil it is neccesssary to generate a custom phase. To have the same randomly generated inlet pores selected, the randomly generated list is saved. If new randomally generated pores is desired than `random.choices(pn.Ps, k=100)` can be used

In [4]:

```
oil = op.phases.GenericPhase(network=pn)
oil['pore.molecular_mass'] = oil['throat.molecular_mass'] = 120.0  # g/mol
oil['pore.viscosity'] = oil['throat.viscosity'] = 1.6  # Pa*s
oil['pore.temperature'] =  298 # K
oil['pore.surface_tension'] = lamda_0 = 0.0720
oil['pore.contact_angle'] = contact_angle = 89
oil['pore.diffusivity'] = 0
oil['pore.thermal_conductivity'] =0
oil['pore.electrical_conductivity'] =0
oil['pore.molecular_weight'] = 100

random_numbers = [31475, 24909, 3302, 6048, 24759, 11029, 5446, 38462, 11392, 807
2, 31374, 13181, 26760,
                  7637, 33085, 32621, 38301, 16641, 30049, 27943, 6615, 5395, 682
3, 32643, 38704, 15168,
                  2403, 21726, 25389, 18762, 7886, 6648, 16476, 14064, 7854, 2902
5, 533, 20038, 25353,
                  35339, 30421, 20836, 21037, 1078, 20226, 23151, 35199, 15857, 3
2470, 25446, 28551, 2829,
                  8934, 19007, 7092, 16336, 35474, 14094, 9781, 12581, 30716, 384
00, 27160, 23725, 16750,
                  34777, 19206, 5136, 348, 2758, 17357, 7149, 27419, 5341, 5204,
613, 28505, 22170, 31845,
                  4394, 30909, 37639, 18820, 28567, 17672, 9968, 23465, 7919, 105
04, 16563, 9998, 16659,
                  23033, 4103, 836, 26054, 35659, 5235, 32712, 7642]

# the physics for the oil phase has to be defined prior to the IP algorithm.
phys_oil = op.physics.Basic(network=pn, phase=oil, geometry=geo)

# Here the IP algorithm is performed
ip = op.algorithms.InvasionPercolation(network=pn, phase=oil)
ip.set_inlets(pores=random_numbers)
ip.run()

# this line designates which pores are occupied by the oil phase
oil.update(ip.results(Snwp=snwp))

# Here trapped water pores are determined and converted to oil pores
mask = ~oil['pore.occupancy']
water_pores = op.topotools.find_clusters(network=pn,mask=mask)
water_pores = water_pores[0]>5
oil['pore.occupancy'] += water_pores
```

Physics of NWP

Here the physics of the NWP are changed to the desired properties. For a explanation into the code see the 2-D adsorption model.

In [5]:

```
water['throat.no_flow'] = pn.tomask(throats=pn.find_neighbor_throats(pores=pn.Ps[
oil['pore.occupancy']]))
oil['throat.occupancy'] = oil['pore.occupancy'][pn['throat.conns']][:,0]*oil['por
e.occupancy'][pn['throat.conns']][:,1]
oil['pore.surface_occupancy'] = pn.tomask(pn.find_connected_pores(water['throat.n
o_flow'] ^ oil['throat.occupancy'],
                                                                  flatten=True))
* oil['pore.occupancy']
water['throat.no_flow'] = pn.tomask(throats=pn.find_neighbor_throats(pores=pn.Ps[
oil['pore.occupancy']]))
oil['throat.occupancy'] = oil['pore.occupancy'][pn['throat.conns']][:,0]*oil['por
e.occupancy'][pn['throat.conns']][:,1]
oil['pore.surface_occupancy'] = pn.tomask(pn.find_connected_pores(water['throat.n
o_flow'] ^ oil['throat.occupancy'],
                                                                  flatten=True))
* oil['pore.occupancy']

phys_water['throat.mask'] = oil['throat.occupancy']*1e-50 + np.invert(oil['throat
.occupancy'])*1.0
phys_water['throat.mask2'] = water['throat.no_flow']*1e-50 + np.invert(water['thr
oat.no_flow'])*1.0
phys_water['throat.diffusive_conductance_original'] = phys_water['throat.diffusiv
e_conductance']
phys_water['throat.hydraulic_conductance_original'] = phys_water['throat.hydrauli
c_conductance']

phys_water.add_model(propname='throat.diffusive_conductance',
                     model=op.models.misc.product, prop1='throat.diffusive_conduc
tance_original', prop2='throat.mask')
phys_water.add_model(propname='throat.hydraulic_conductance',
                     model=op.models.misc.product, prop1='throat.hydraulic_conduc
tance_original', prop2='throat.mask2')
mod = op.models.physics.ad_dif_conductance.ad_dif
phys_water.add_model(propname='throat.ad_dif_conductance', model=mod, s_scheme='h
ybrid')
phys_water.regenerate_models('throat.ad_dif_conductance')
```

Stokes flow

Permeability is found using Darcy's law, and the necessary pressure gradient to achieve the desired flow rate is used

In [6]:

```
permeability = SF.permeability(pn,geo,water,D,L)
mu = water['pore.viscosity'].max()
del_P = mu*darcy_velocity*L/(permeability)

sf = op.algorithms.StokesFlow(network=pn, phase=water)
sf.set_value_BC(pores=inlet_pores, values=del_P)
sf.set_value_BC(pores=outlet_pores, values=0.0)
sf.run();
flow = sf.rate(throats=pn.Ts,mode='single')
```

Diffusivity and Advection-Diffusion

In [7]:

```
D_SE = water['throat.diffusivity'] = water['pore.diffusivity'] = DF.Stokes_Einste
in(
    water['pore.temperature'][0],water['pore.viscosity'][0],dp)
water.regenerate_models()
phys_water.regenerate_models('throat.ad_dif_conductance')
```

Adsorption Physics

The rate in which particles in solute are adsorbed onto the interface is given by:

$$r_{i}^{ads}=\frac{d n_{ads}}{dt}=A_{s}k_{a}\bar B(\theta)C_{0}$$

Where $\bar B(\theta)$ is given by

$$\bar B(\theta)=2.32\bigl(1-\frac{\theta}{\theta_{max}}\bigr)^{m} $$

For each pore a new $A_{s}k_{a}\bar B(\theta)$ is solved for every time step. This is refered to as `pore.reaction_coeff`.

In [8]:

```
ad = op.algorithms.TransientAdvectionDiffusion(network=pn, phase=water)

phys_water['pore.adsorbed'] = np.zeros(pn.Np)
phys_water['pore.surface_area'] = geo['pore.surface_area']
phys_water['pore.volume'] = geo['pore.volume']

phys_water.add_model(propname='pore.B',
                model=SF.Blocking,area='pore.surface_area',
                adsorbed='pore.adsorbed',theta_max=0.91,r=dp/2,m=3,alg_name=ad)

k = -2.32*((D_SE/dp)*(phib_kbT/pi)**0.5)*np.exp(-phib_kbT) #m/s, 2.32 is carried
over from the blocking function
water['pore.concentration'] = 0.0
```

```
phys_water.add_model(propname='pore.reaction_coeff',model=SF.reaction_coefficient
,
                     surface_area='pore.surface_area',reaction_constant=k,B='pore.
B')
phys_water.add_model(propname='pore.rxn',model=op.models.physics.source_terms.pow
er_law,
                    A1='pore.reaction_coeff', X='pore.concentration')
phys_water.regenerate_models()
```

Transient advection diffusion settings

Source terms can be added to OpenPNM, which can be thought as a chemical reaction. The rate of adsorption for each pore, r_{i}^{ads}, is taken to be a negative "source term".

It is necessary to define length of each time step and the final time in a transient algotithm. However, it is desired to update the physics for each time step to account for the blocking. This would normally be difficult since it would require a rewriting of the `transientreactivetransport` code in openpnm. A workaround for this problem was found by only running one time-step at a time, and then putting the `TransientAdvectiveDiffusion` algorithm in a for loop. The inital conditions for particle concentration is updated to match the final concentrations for the previous time-step.

In [9]:

```
ad.set_value_BC(pores=inlet_pores, values=inj_conc)
ad.set_outflow_BC(pores=outlet_pores)
phys_water.models['pore.B']['alg_name'] = ad.name
ad.settings['t_scheme'] = 'implicit'
ad.settings['t_output'] = 0.01
ad.settings['t_tolerance'] = 0.0
ad.settings['store_rate'] = False
ad.settings['t_final'] = dt
ad.settings['t_step'] = dt
ad.settings['t_precision'] = 4

# All pores occupied by an oil phase is set to be a "sink".

ps = oil['pore.occupancy']*np.isnan(ad['pore.bc_value'])*np.isnan(ad['pore.bc_out
flow'])
ad.set_source(propname='pore.rxn', pores=ps)

# Initial condition of having 0 concentration of particles everywhere

C0 = C = np.zeros(pn.Np)
ad.set_IC(values=C)
```

```
# Defining lists that record data for each step
total_eluted = [0.0]
total_injected = [0.0]
td_steps = [0.0]
solute_mass = [0.0]
total_adsorbed = [0.0]
inlet_rate = [0.0]
outlet_rate = [0.0]

ad['pore.adsorbed@0'] = np.zeros(pn.Np)
ad['pore.theta@0'] = np.zeros(pn.Np)
total_flow = sf.rate(throats=inlet_throats,mode='group')[0]
```

In [11]:

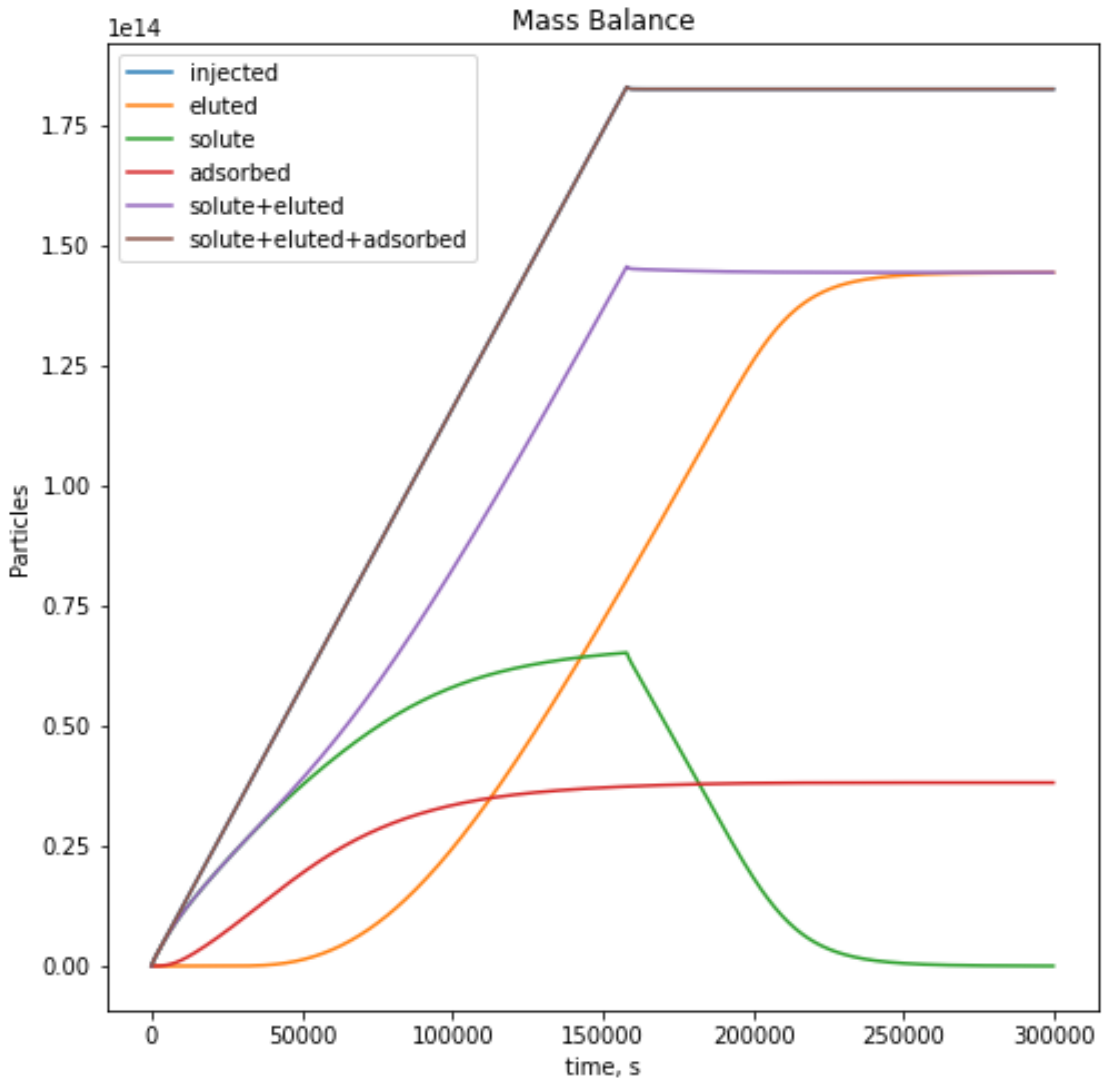
```
oil_frac = sum(geo['pore.volume']*oil['pore.occupancy'])/sum(geo['pore.volume'])
effective_porosity = (sum(geo['pore.volume'])-sum(geo['pore.volume']*oil['pore.oc
cupancy']))/(A*L)
void_volume = effective_porosity*L*A

for i in range(1, num_steps, 1):

    # This checks if the desired number of pore volumes has been injected. If it
has the injection concentration is set to 0
    # These lines can be rewritten so it is shut off when a desired mass of parti
cles is injected, or removed completly
    # if a step change injection is desired instead.

    if void_volume*Pore_Volumes_Injected < total_flow*td_steps[-1]:
        ad.set_value_BC(pores=inlet_pores, values=0.0)

    ad.run()

    # these lines just recored desired mass balance data

    td_steps.append(td_steps[-1]+dt)
    inj_mass = ad.rate(pores=inlet_pores,mode='group')*dt
    total_injected.append(total_injected[-1]+inj_mass[0])
    eluted = ad.rate(pores=outlet_pores,mode='group')[0]*dt*-1.0
    total_eluted.append(total_eluted[-1]+eluted)
    inlet_rate.append(ad.rate(throats=inlet_throats,mode='group')[0])
    outlet_rate.append(ad.rate(pores=outlet_pores,mode='group')[0]*-1)
    solute_mass.append(np.sum(ad['pore.concentration'][body_pores]*geo['pore.volu
me'][body_pores]))

    # Here the total number of particles is calculated, and the new blocking valu
es for each pore is determined.
```

```
    dNdt = -1*phys_water['pore.rxn.rate']*ps
    phys_water['pore.adsorbed'] = ad['pore.adsorbed@'+str(dt*i)] = ad['pore.adsor
bed@'+str(dt*(i-1))] + dNdt*dt
    total_adsorbed.append(np.sum(ad['pore.adsorbed@'+str(ad.settings['t_solns'][-
1])]))
    ad['pore.theta@'+str(ad.settings['t_solns'][-1])] = np.pi*(dp/2)**2*ad['pore.
adsorbed@'+str(ad.settings['t_solns'][-1])]/geo['pore.surface_area']

    print(i)

    # THe physics has to be updated for values that change as the simulation proc
eedes

    phys_water.regenerate_models('pore.B')
    phys_water.regenerate_models('pore.reaction_coeff')
    phys_water.regenerate_models('pore.rxn')

    # The inital concentrations for every pore is updated to match the final conc
entrations for the previous step

    C = ad['pore.concentration']
    ad.set_IC(values=C)
    ad.settings['t_initial'] = i*dt
    ad.settings['t_final'] = (i+1.0)*dt
```

This chart presents a mass balance for all particles in the network. A verification that the mass is conserved is to check if the curve for injected particles matches the curve for solute+eluted+adsorbed

In [12]:

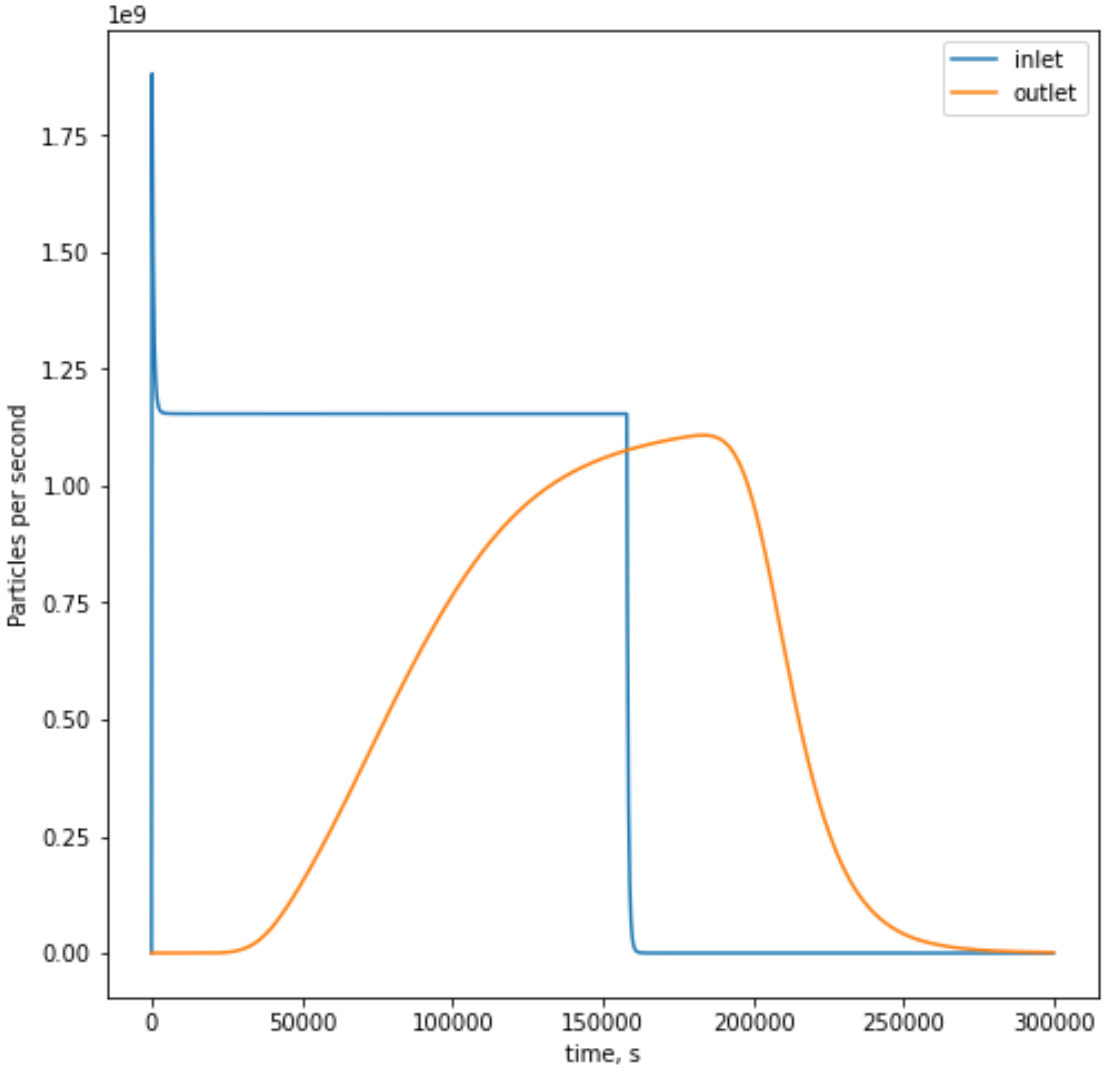
```
s_e = np.add(total_eluted,solute_mass)
s_e_a = np.add(s_e,total_adsorbed)

plt.figure(figsize=(8, 8));
plt.title('Mass Balance')
plt.plot(td_steps,total_injected,td_steps,total_eluted,td_steps,solute_mass
         ,td_steps,total_adsorbed,td_steps,s_e,td_steps,s_e_a)
plt.gca().legend(('injected','eluted','solute','adsorbed','solute+eluted','solute
+eluted+adsorbed'));
plt.xlabel('time, s');
plt.ylabel('Particles');
```



Injection and elution curves. When there is a low Peclet number a large spike appears at the start of the inlet injection curve.

In [13]:

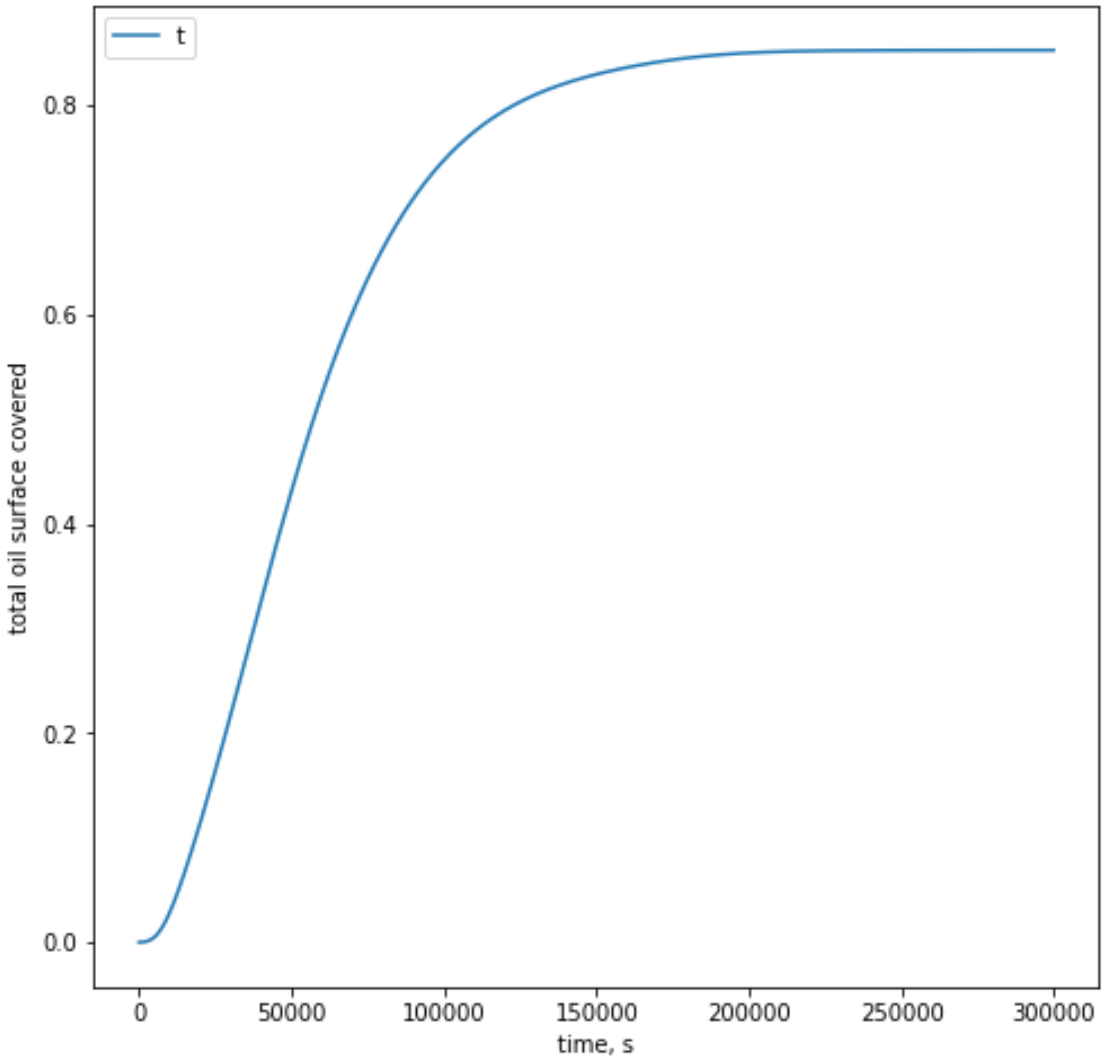
```
plt.figure(figsize=(8, 8));
plt.plot(td_steps,inlet_rate,td_steps,outlet_rate);
plt.gca().legend(('inlet','outlet'));
plt.xlabel('time, s');
plt.ylabel('Particles per second');
```



In [14]:

```
oil_surface_area = sum(oil['pore.surface_occupancy']*geo['pore.surface_area'])
total_frac_coverage = np.multiply(total_adsorbed,3.14/4*dp**2/(oil_surface_area))

plt.figure(figsize=(8, 8));
plt.plot(td_steps,total_frac_coverage);
plt.gca().legend(('total coverage'));
plt.xlabel('time, s');
plt.ylabel('total oil surface covered');
print(total_frac_coverage[-1])
```

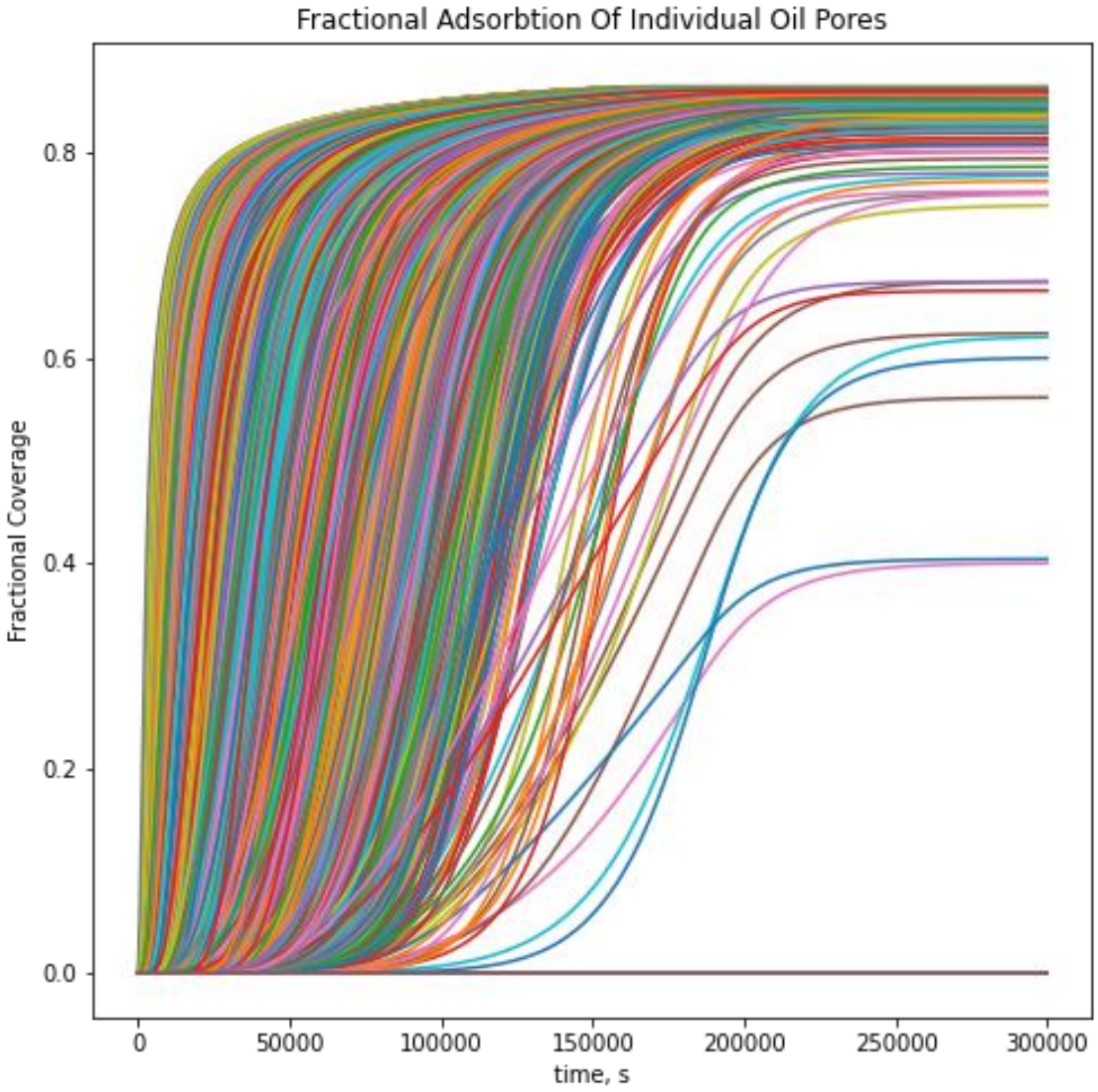
```
0.8528534551256293
```



In [15]:

```
def plot_trace(alg, pores, propname='pore.theta'):
    vals = []
    times = []
    q = propname
    for t in alg.settings['t_solns']:
        try:
            vals.append(alg[f'{q}@' + t][pores])
            times.append(float(t))
        except:
            pass
    return times, vals
```

```
data = plot_trace(ad,ps)
plt.figure(figsize=(8,8))
plt.plot(*data);
plt.title('Fractional Adsorbtion Of Individual Oil Pores');
plt.ylabel('Fractional Coverage');
plt.xlabel('time, s');
```



Calculating other things and saving data to an excel file

In [19]:

```
tau = 1.39798998 # tortuosity
D_eff_ave = (np.sum(water['pore.diffusivity']*geo['pore.volume']*(1-oil['pore.occ
```

```
upancy']))+
             np.sum(water['throat.diffusivity']*geo['throat.volume']*
                (1-oil['throat.occupancy'])))/(np.sum(geo['pore.volume']*
                (1-oil['pore.occupancy']))+np.sum(geo['throat.volume']*(1-oil['th
roat.occupancy'])))
Peclet_number = darcy_velocity*Dc/(D_eff_ave*effective_porosity)
PV = [total_flow*i/void_volume for i in td_steps]
CC0 = outlet_rate/total_flow/inj_conc

Dl = (Peclet_number/6)*np.log((3*tau*Peclet_number/2)-0.25)*D_eff_ave
Vb = A*L
Vw = Vb*effective_porosity
a0 = oil_surface_area/Vw
dkn1 = k*-1*L*L*a0/D
CC0 = outlet_rate/total_flow/inj_conc
```

In [17]:

```
import pandas as pd

a = {'time step, s': td_steps,'Pore Volumes': PV, 'Total Injected Particles': tot
al_injected,
     'Total Eluted Particles':total_eluted,'Total Solute Particles':solute_mass,
'Total Adsorbed Particles':total_adsorbed,
     'Maximum Coverage Frac': total_frac_coverage,'C/C0':CC0}

df = pd.DataFrame(data=a)

b = {'Particle Size m':[dp],'Phi_b/k_b*T':[phib_kbT],'Particle Volume Frac':[vol_
frac],
    'Darcy Velocity m/s':[darcy_velocity], 'Oil void space frac':[oil_frac],
    'Effective Diffusivity':[D_eff_ave],'Peclet Number':[Peclet_number],'Oil Surf
ace Area':[oil_surface_area],
    'Adsorption rate constant':[k*-1], 'Damkholer dispersive':[dkn1]}
df2 = pd.DataFrame(data=b)

c = {'Time Step s':[dt],'Particle Density g/l':[particle_density],'Column Length
m':[L],
     'Column Cross-Sectional Area m^2':[A],'absolute porosity':[porosity],'Effect
ive Porosity':[effective_porosity]}
df3 = pd.DataFrame(data=c)
with pd.ExcelWriter(excel_file_name) as writer:
    df.to_excel(writer,sheet_name='outputs')
    df2.to_excel(writer,sheet_name='important_constants')
    df3.to_excel(writer,sheet_name='other_constants')
```