Paper:
Lo, S., Jones, J., Hassan, O. & Hilal, N. (2019). Development of an axisymmetric parallel solution algorithm for membrane separation process. *Desalination, 471*, 114127
http://dx.doi.org/10.1016/j.desal.2019.114127

Development of an axisymmetric parallel solution algorithm for membrane separation process

S.B. Lo[1*], J.W. Jones[1], O. Hassan[1], and N. Hilal[1]

[1]College of Engineering, Swansea University, Swansea SA1 8EN
[*]839372@swansea.ac.uk

August 27, 2019

Abstract

A novel parallel technique that couples the lattice–Boltzmann method and a finite volume scheme for the prediction of concentration polarisation and pore blocking in axisymmetric cross–flow membrane separation process is presented. The model uses the Lattice–Boltzmann method to solve the incompressible Navier–Stokes equations for hydrodynamics and the finite volume method to solve the convection–diffusion equation for solute particles.

Concentration polarisation is modelled for micro–particles by having the diffusion coefficient defined as a function of particle concentration and shear rate. The model considers the effect of an incompressible cake formation. Pore blocking phenomenon is predicted for filtration membrane fouling by using the rate of particles arriving at the membrane surface.

The simulation code is parallelised in two ways. Compute Unified Device Architecture(CUDA) is used for a cluster of graphical processing units(GPUs) and Message Passing Interface(MPI) is utilised for a cluster of central processing units(CPUs), with various parallelisation techniques to optimise memory usage for higher performance. The proposed model is validated by comparing to analytical solutions and experimental result.

Keywords: Filtration, Concentration Polarisation, Cake Formation, Pore Blocking, Parallel Programming

Highlight

- An unifying model for predicting concentration polarisation, cake formation and pore blocking.
- A novel coupling of axisymmetric lattice Boltzmann – finite volume schemes is developed.
- Parallel implementation that caters for multiple hardware platforms.

List of Abbreviations

CUDA	**C**ompute **U**nified **D**evice **A**rchitecture
GPU	**G**raphics **P**rocessing **U**nit
MPI	**M**essage **P**assing **I**nterface
CPU	**C**entral **P**rocessing **U**nit

LBM	**L**attice **B**oltzmann **M**ethod
FV	**F**inite **V**olume
PDF	**P**artial **D**istribution **F**unction
MRT	**M**ultiple **R**elaxation **T**ime
BC	**B**oundary **C**ondition
AoS	**A**rray **o**f **S**tructures
SoA	**S**tructure **o**f **A**rrays
RAM	**R**andom **A**ccess **M**emory
MLUPS	**M**illion **L**attices **U**pdate **P**er **S**econd

List of Symbols

ρ	Density $(kg \cdot m^{-3})$
$\mathbf{u}$	Velocity $(m \cdot s^{-1})$
t	Time(s)
ρ	Density $(kg \cdot m^{-3})$
p	Pressure (Pa)
ν	Kinematic viscosity $(m^2 \cdot s^{-1})$
ϕ	Volume fraction of particles
D	Diffusion coefficient $(m^3 \cdot m^{-2} \cdot s^{-1})$
ϕ_{max}	Maximum volume fraction of particles
γ	Shear rate (s^{-1})
a	Radius of particle (m)
f_i	Partial distribution function
f_i^{eq}	Equilibrium distribution function
$\mathbf{x}$	Displacement (m)
$\mathbf{c}$	Unit lattice velocity
δt	Lattice time-step (s)
Ω	Collision operator
δr	Lattice radial spacing (m)
δz	Lattice axial spacing (m)
$\mathbf{F}$	Body force $(kg \cdot m \cdot s^{-2})$
R	Resistance (m^{-1})
$\mathbf{M}$	Moment matrix
$\mathbf{S}$	Relaxation vector
$\mathbf{m^{eq}}$	Moment equilibrium vector
$\mathbf{F'}$	Body force vector
V	Volume (m^{-3})
Γ	Surface (m^{-2})
Δt	FV time-step (s)
Δr	FV radial spacing (m)
Δz	FV axial spacing (m)
V_c	Volume of particles in a unit volume (m^3)
V_s	Volume of a particle (m^3)
N_c	Number of particles in a unit volume
u_n	Fluid velocity normal to membrane $(m \cdot s^{-1})$
r_p	Rate of particles reaching membrane (s^{-1})
P	Probability of a pore being free
β	Pore blocking parameter $(s \cdot m^{-2})$
N_p^t	Number of free pores at time t
ΔP	Transmembrane pressure (Pa)
A_p	Cross-sectional area of a pore (m^2)
A_m	Surface area of membrane (m^2)
a_p	Radius of membrane pore (m)
μ	Dynamic viscosity $(Pa \cdot s)$
τ	Tortuosity
δ_m	Thickness of membrane (m)
R_m	Membrane resistance (m^{-1})
ϵ	Cake porosity

1 Introduction

Over the last few decades membrane filtration has become increasingly important in industries, due to the higher efficiency compared to other separation methods. Membrane filtration is used in a wide variety of processes, from water treatment[1] and dairy processing[2][3] to oil purification[4][5] and medical devices[6]. These processes require experiments to calibrate the optimal set-up, which include using different types of membrane material, changing the distribution of pore sizes, varying operating pressure, and altering the geometry of the set-up. To speed up the process, simulation models are developed to help determining the optimal operating conditions.

Concentration polarisation is, in general, a reversible fouling, which leads to particles depositing onto the membrane, and in turn, reduces the efficiency of the filtration process. Therefore, predicting the evolution of concentration polarisation is important. Models have been developed for cross-flow membrane filtration from an early stage. These models include Chudacek and Fane[7] that used a resistance model of the filtration theory to predict flux decline. Porter[8] used a model that considers back diffusion phenomenon by utilising a simplified convection–diffusion equation. Zydeny and Colton[9] proposed a model based on shear induced diffusion which is a major back diffusion mechanism in microfiltration[10]. These models are one dimensional and require experimental data to calibrate.

Subsequently, 2D models were developed, such as work by Lee and Clark[11] which solves 2D convection–diffusion equation with an assumed cross-flow velocity field; the model also includes cake formation prediction. Hong et al.[12] proposed a model that predicts the permeate flux decline due to concentration polarisation and cake growth, based on simple particle mass balance equation. However, the model is only applicable for early stages of filtration as cake is forming. Song and Elimelech[13] developed a theory that utilises a filtration number based on particle size and transmembrane pressure to predict whether concentration polarisation evolves to cake formation. The model was further developed[14] to determine the time-dependent flux and time required to reach steady state. Bhattacharjee et al.[15] proposed a model that takes solute particle interactions into consideration, however it is assumed that cake layer is absent. Kim et al.[16] proposed a model that takes the Derjaguin, Verway, Landau and Overbeek interactions between solute particles into consideration to predict formation of cake layer.

Whilst concentration polarisation models are common and well developed, pore blocking is a more complex mechanism that involves particle and pore interactions. Hermia[17] proposed a model for pore blocking in dead-end filtration, which has been converted and applied in cross-flow filtration such as work done by Brião et al.[18] and Konieczny [19]. The model is based on classical dead-end filtration with constant pressure. It derives a differential equation, which depends on the type of pore blocking mechanism involved, to describe the permeate flux decline. Alternative models include Broeckmann et al.[20] which utilise particle size distribution and pore size distribution to calculate the membrane resistance. On a microscopic level, Kim and Liu[21] used a hydrodynamic force bias Monte Carlo method and incorporated it with Brownian and shear induced diffusion to simulate particle behaviour in a cross-flow filtration.

The inorganic nature of ceramic membranes give them higher mechanical strength than organic counterparts[22], thus able to be produced in tubular form. The tubular structure is axisymmetric , which means it has cylindrical symmetry and is not dependent on the angle when using cylindrical coordinates; this allows 2D axisymmetric models to be used instead of traditional 3D models, which has simplified calculations for faster simulations. Early attempts to model ceramic membranes include Doleček[22] who modelled ceramic membrane monoliths and solved with the finite element method. Whilst the monolith honeycomb structure is not axisymmetric, he attempted to utilise the symmetric nature inside the structure to simplify the simulation domain. Konieczny[19] proposed a relaxation model to the permeate flux and membrane resistance, as well as including a pore blocking model(Hermia model). However the models are focusing on simulating the permeate flux only. Pak[23]

proposed a computational fluid dynamic model for cylindrical porous membranes which uses a fine grid for the boundary layer to predict the effects and growth rate of the concentration polarisation. The model does not extend to other fouling mechanisms and assumes low solute concentration which restrict the usage of it. Thus, there exists a need to develop a more in-depth filtration simulation model to solve axisymmetric problems.

Different schemes were used to solve the governing equations in cross-flow filtration. Huang and Morrissey[24] used a finite elements technique to solve convection–diffusion equation, whereas Lee and Clark used a finite difference scheme, both with an assumed velocity profile. Geraldes et al.[25] used a finite volume method to solve for both cross-flow velocity and concentration distribution. Kromkamp et al.[26] developed a model that solves cross-flow hydrodynamics and particle distribution based on Lattice Boltzmann Method(LBM) in a coupled manner. The Kromkamp model is further developed by Kim[27] and Paipuri[28] to include particle interactions and osmotic pressure, and is the basis of our current model.

Attempts to modify a Cartesian grid LBM to axisymmetric LBM were made with various methods. One way is to add an additional term to the 2D LBM, such as the work by Halliday et al.[29]. This type of schemes require a gradient term which disallows efficient parallelisation. For example Niu et al.[30] extended the idea in 2003 to include azimuthal rotation in the scheme; Lee et al.[31] proposed a similar scheme to reduce the compressibility effect in Hallidays model. Others developed axisymmetric schemes that do not involve gradient terms, such as the work done by Xie et al.[32], or the work done by Chen et al.[33] that develops the scheme from vorticity-stream-function. In 2009, Guo et al.[34] proposed an axisymmetric LBM derived from the cylindrical governing equations. The resulting scheme requires more calculations but retains many features close to the standard LBM such as utilising local information only and able to incorporate force terms easily into the scheme. This scheme is subsequently developed into a multiple relaxation time scheme[35] and a thermal scheme[36]. The thermal scheme can also be used for solute particle transport .For these reasons we have chosen this scheme to form the basis of our axisymmetric model.

With modern advances in computing hardware, parallel programming allows simulation to fully utilise the potential of a cluster of CPUs or GPUs for modelling industrial cases with the appropriate grid resolution in an acceptable time scale. Methods to parallelise LBM are well established in literature for CPUs and GPUs. Bella et al.[37] implemented LBM with Open Multi-Processing which scales with high efficiency, Davidson[38] proposed techniques to optimise MPI communications for LBM and was able to attain excellent scalability. Lee et al.[39] analysed the performance of LBM on CPU and GPU, and concluded that LBM is bandwidth bound, meaning that the performance bottleneck is sending data between processor and memory storage. Therefore the selected GPU is roughly 5 times faster than a multi-core CPU due to bandwidth difference, and optimisation techniques are usually focused on minimising data transfer and choosing the best data structure. Various techniques to parallelise LBM on GPU are present in literature such as Rinaldi et al.[40] and Myre et al[41]. There are also attempts to optimise computations[42] according to the design of processors, but such optimisation may not be applicable on other devices that have different architectures.

The focuses of this paper are on microfiltration and ultrafiltration where the main causes of permeate flux decline are concentration polarisation, cake formation and pore blocking. In order to reduce the number of physical experiments needed to calibrate and optimise the process of ceramic filtration, a computational model that is capable of predicting changes in cross-flow filtration is presented. The formulation is extended to enable the modelling of axisymmetric scenarios, making it suitable for simulating experiments using tubular membranes. The simulation software is finally optimised and parallelised for CPU cluster and GPU cluster. The current work assumes that particle interactions, osmotic pressure and cake compaction are negligible. Cake compression is currently ignored as the model focuses on the period when cake is forming rather than a prolonged filtration.

2 Mathematical Model

The filtration model consists of two parts: hydrodynamics of the fluid, represented by the incompressible Navier–Stokes equations; and particle transportation, represented by the convection–diffusion equation. The two systems are solved in two stages in a staggered manner. In stage one the hydrodynamic equations are solved; the convection diffusion equation is solved in stage two using the velocity field obtained in stage one.

The axisymmetric incompressible Navier–Stokes equations can be expressed as

$$\frac{\partial u_r}{\partial t} + u_r \frac{\partial u_r}{\partial r} + u_z \frac{\partial u_r}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial r} + \nu(\nabla^2 u_r - \frac{u_r}{r^2}) \tag{1}$$

$$\frac{\partial u_z}{\partial t} + u_r \frac{\partial u_z}{\partial r} + u_z \frac{\partial u_z}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \nu\nabla^2 u_z \tag{2}$$

where $\mathbf{u}(r, z, t)$ is the velocity of fluid, in terms of radial and axial position, and time; $\nabla^2 = \frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial}{\partial r}) + \frac{\partial^2}{\partial^2 z}$ is the axisymmetric Laplacian, p is the pressure, ν is the kinematic viscosity. The numerical solution of this system of equations can be obtained using the finite difference, finite volume(FV), finite element method or the lattice Boltzmann methods. Here we choose the LBM for its ease of implementation on parallel platforms such as GPUs and CPUs clusters.

The axisymmetric convection–diffusion equation can be written as

$$\frac{\partial \phi}{\partial t} + u_r \frac{\partial \phi}{\partial r} + u_z \frac{\partial \phi}{\partial z} = D\nabla^2 \phi \tag{3}$$

where ϕ is the volume fraction of particle concentration which measures the percentage of the control volume occupied by the particles, D is the diffusion coefficient. It is possible to use the LBM to solve for particle concentration; however, the scheme is computationally expensive for small diffusion coefficients as it requires a small lattice spacing and small time-step to maintain stability. Hence, a finite volume scheme is utilised to solve the convection–diffusion equation.

To couple the two schemes together, kinematic viscosity and diffusion coefficient are assumed to change locally. Kinematic viscosity is based on the local volume fraction of solute ϕ and the viscosity of fluid in the absence of any solute, as proposed by Romero and Davis[43]

$$\nu(\phi) = \nu_0[1 + 1.5\frac{\phi}{(1 - \phi/\phi_{max})}]^2 \tag{4}$$

where ϕ_{max} denotes the maximum packing volume fraction of solute particles. By assuming particles are spherical and they are packed together randomly, the maximum packing fraction is about 0.64[44][45].

Similarly for the convection–diffusion equation, the diffusion coefficient is set to be a function of the local concentration of particles. Cho et al.[46] determined that shear induced diffusion dominates for particle sizes more than $0.5\mu m$, and can be calculated by using shear rate and particle size. The shear induced diffusion coefficient in the filtration device is estimated by using work of Leighton and Acrivos[47]

$$D = 0.33\gamma a^2 \phi^2 (1 + 0.5e^{8.8\phi}) \tag{5}$$

where the diffusion coefficient is related to γ the shear rate of fluid, a the radius of particle and volume fraction of particles ϕ.

2.1 Lattice Boltzmann Scheme for Fluid Flow

The lattice Boltzmann method is derived from the kinetic theory of gases, which treats a body of fluid as a distribution of small particles, which move in random directions with

random velocities. It consists of two steps to solve the Navier–Stokes equations, streaming and collision, which can be expressed by

$$f_i(\mathbf{x}_n + \mathbf{c}_i\delta t, t + \delta t) - f_i(\mathbf{x}_n, t) = \Omega(\mathbf{f}) \tag{6}$$

The discrete Particle Distribution Function(PDF) f_i describes the portion of particles moving in a particular direction. The left hand side represents streaming where particles travel with velocity $\mathbf{c}_i$ from $\mathbf{x}_n$ to their new location $\mathbf{x}_n + \mathbf{c}_i\delta t$ after a time-step δt; the right hand side is the collision operator $\Omega(\mathbf{f})$ which simulates momentum exchange by particle collision.

The simulation scheme used here restricts particles to a possible 9 directions, stationary and a combination of radial direction and axial direction. This is also known as a D2Q9 lattice Boltzmann scheme which stands for 2 dimensions and 9 directions. The velocity vectors $\mathbf{c}_i$ are defined as

$$\mathbf{c}_i(r,z) = \begin{cases} (0,0) & \text{if } i = 0, \\ (cos((i-1)\pi/2), sin((i-1)\pi/2))c & \text{if } i \in [1,4], \\ (cos((2i-9)\pi/4), sin((2i-9)\pi/4))\sqrt{2}c & \text{if } i \in [5,8]. \end{cases} \tag{7}$$

c is the lattice speed which is related to the lattice spacing $\delta_r = \delta_z$ and lattice time δ_t

$$c = \frac{\delta_r}{\delta_t} \tag{8}$$

macroscopic variables density ρ, radial velocity u_r and axial velocity u_z are given by

$$\rho = \frac{1}{r}\sum_{i=0}^{8} f_i, \quad \rho u_r = \frac{r}{r^2+\nu}[\sum_{i=0}^{8} c_{ir} f_i + \frac{\delta_t \rho}{6} + \frac{\delta_t r \rho F_r}{2}], \quad \rho u_z = \frac{1}{r}[\sum_{i=0}^{8} c_{iz} f_i + \frac{\delta_t}{2} r \rho F_z] \tag{9}$$

$$F_z = -\nu R u_z \qquad F_r = -\nu R u_r + \frac{1}{3r}(1 - \frac{6\nu u_r}{r}) \tag{10}$$

where R is the membrane resistance for membrane lattice, or $R = 0$ for a fluid lattice.

The collision operator used in the proposed model is axisymmetric multiple relaxation time developed by Wang et al.[35]. By using Multiple Relaxation Time(MRT) the model is more stable than the traditional single relaxation time model, thus less restrictive when choosing a desirable scale for δ_r and δ_t.

The MRT collision term can be expressed as

$$\Omega(\mathbf{f}) = \mathbf{M}^{-1}\mathbf{S}[\mathbf{m}^{eq} - \mathbf{M}\mathbf{f}] + \mathbf{M}^{-1}(1 - \frac{\mathbf{S}}{2})\mathbf{F}' \tag{11}$$

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix} \tag{12}$$

where $\mathbf{M}$ transform the PDFs into moment space, $\mathbf{M}\mathbf{f} = r(\rho, e, \epsilon, j_z, q_z, j_r, q_r, p_{zz}, p_{rz})$. ρ is mass density, e, ϵ are related to energy and energy squared, $\mathbf{j}$ is momentum, $\mathbf{q}$ is energy flux, and $\mathbf{p}$ is the symmetric traceless viscous stress tensor. Each variable then adjusts according

to the respective equilibrium value $\mathbf{m^{eq}}$ with relaxation time $\mathbf{S}$.

$$\mathbf{m^{eq}} = \begin{pmatrix} \rho^{eq} \\ e^{eq} \\ \epsilon^{eq} \\ j_z^{eq} \\ q_z^{eq} \\ j_r^{eq} \\ q_r^{eq} \\ p_{zz}^{eq} \\ p_{rz}^{eq} \end{pmatrix} = \begin{pmatrix} \rho \\ \rho[-2+3(u_z^2+u_r^2)] \\ \rho[1-3(u_z^2+u_r^2)] \\ \rho u_z \\ -\rho u_z \\ \rho u_r \\ -\rho u_r \\ \rho(u_z^2-u_r^2) \\ \rho u_z u_r \end{pmatrix} \tag{13}$$

The values of $\mathbf{S} = (s_0, ..., s_8)$ used in this work are $s_0 = s_3 = s_5 = 0$ by conservation of mass and momentum. $s_1 = 1.5, s_2 = 1.4, s_4 = s_6 = 1.5$ and $s_7 = s_8 = \frac{1}{3\nu+0.5}$, are chosen for current work.

Finally $\mathbf{F'}$ represents the forces acting on the fluid and is given by,

$$\mathbf{F'} = r \begin{pmatrix} 0 \\ -3\rho(u_z F_z + u_r F_r)(-2+3u_z^2+3u_r^2) \\ 3\rho(u_z F_z + u_r F_r)(-2+3u_z^2+3u_r^2) \\ \rho F_z \\ \rho(3F_z u_r^2 + 6u_z u_r F_r - F_z) \\ \rho F_r \\ \rho(3F_r u_z^2 + 6u_z u_r F_z - F_r) \\ -\rho[3(u_z F_z + u_r F_r)(u_z^2-u_r^2) - 2(u_z F_z - u_r F_r)] \\ -\rho[3u_z u_r(u_z F_z + u_r F_r) - (u_z F_r + u_r F_z)] \end{pmatrix} \tag{14}$$

2.1.1 Boundary Conditions for Lattice Boltzmann

Four types of boundary conditions(BCs) are used in the current simulation code. Mirror boundary condition is used for line of symmetry, solid boundary condition is used for solid walls, velocity and pressure boundary conditions for inlet and outlet boundaries are adapted from implementation proposed by Zou and He[48], with suitable changes made for axisymmetric LBM.

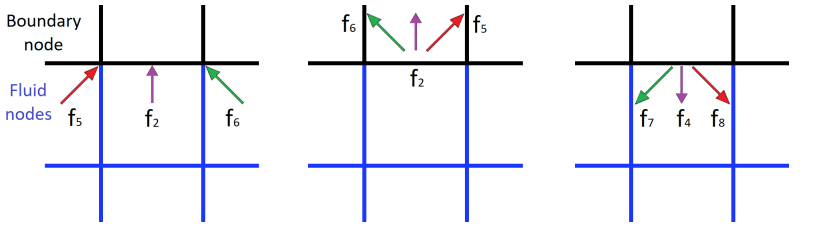
Symmetry and Wall Boundary Conditions

Mirror boundary condition, also known as full-slip wall, reflects the PDFs as they approaches the boundary in such a way that the lattice node behind the boundary mimics the one in front. Figure 1 illustrates the process of streaming and how the boundary condition redirects the PDFs to the correct node. This implementation sets the boundary at half way between nodes, i.e. the boundary is at $r = 0$ and the first node is located at $r = 0.5$. The boundary condition with the mirror placed on the top side is illustrated by figure 1 and can be summarised as

$$\begin{pmatrix} f_8^{i,j-1} \\ f_4^{i,j-1} \\ f_7^{i,j-1} \end{pmatrix} = \begin{pmatrix} f_5^{i,j} \\ f_2^{i,j} \\ f_6^{i,j} \end{pmatrix} \tag{15}$$

where node (i, j) is the mirror node.

Solid boundary condition is very similar to mirror boundary condition, except it imposes no-slip condition on the boundary and PDFs are reversed and bounced back to the source node. Figure 2 illustrates the process of this boundary condition. Similar to the mirror boundary condition, the solid wall boundary by this method is located half way between lattice nodes. The boundary condition with the wall placed on the top side is illustrated by



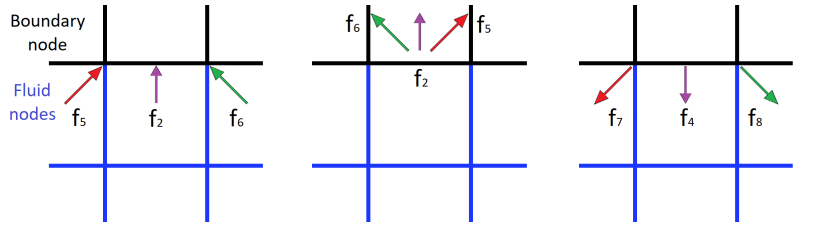
(a) Approaching wall (b) Arrived at wall (c) Reflected by wall

Figure 1: Illustration of mirror boundary condition: partial distribution functions f_2, f_5, f_6, (a) approaching wall, (b) arrived at wall and (c) reflected by mirror boundary condition.

figure 2 and can be summarised as

$$\begin{pmatrix} f_7^{i-1,j-1} \\ f_4^{i,j-1} \\ f_8^{i+1,j-1} \end{pmatrix} = \begin{pmatrix} f_5^{i,j} \\ f_2^{i,j} \\ f_6^{i,j} \end{pmatrix} \tag{16}$$

where node (i, j) is the wall node.



(a) Approaching wall (b) Arrived at wall (c) Reflected by wall

Figure 2: Illustration of wall boundary condition: partial distribution functions f_2, f_5, f_6, (a) approaching wall, (b) arrived at wall and (c) reflected by wall boundary condition.

Pressure and Velocity Boundary Condition

To impose a pressure or velocity boundary condition, Zou and He[48] proposed a scheme to calculate the missing PDFs. The principle is to use the equations of macroscopic variables and one assumption to find the missing macroscopic variable and three PDFs. For example an inlet from the bottom of the grid will be missing f_2, f_5 and f_6 due to no nodes streaming from below. Suppose velocity is known, then by using equation 9,

$$f_2 + f_5 + f_6 = r\rho u_z + f_4 + f_7 + f_8 \tag{17}$$

By substituting this into the density equation, the value of density can be calculated

$$\rho = \frac{f_0 + f_1 + f_3 + 2(f_4 + f_7 + f_8)}{r(1 - u_z)} \tag{18}$$

In order to solve the system of equations, one more equation is required, which Zou and He proposed an assumption to complete the system.

$$f_2 - f_2^{eq} = f_4 - f_4^{eq} \tag{19}$$

by expanding the equilibrium terms, f_2 can be obtained by

$$f_2 = f_4 + \frac{2r\rho}{3} u_z \tag{20}$$

Using the remaining radial momentum equation

$$f_5 = \frac{r\rho u_z}{6} + \frac{(r^2+\nu)\rho u_r}{2r} - \frac{\rho}{12} + f_7 + \frac{f_3 - f_1}{2} \tag{21}$$

$$f_6 = \frac{r\rho u_z}{6} - \frac{(r^2+\nu)\rho u_r}{2r} + \frac{\rho}{12} + f_8 - \frac{f_3 - f_1}{2} \tag{22}$$

If pressure is known, instead of the incoming axial velocity, then by $p = \frac{1}{3}\rho$, and the density of the boundary is defined. Starting with the density equation,

$$f_2 + f_5 + f_6 = r\rho - f_0 - f_1 - f_3 - f_4 - f_7 - f_8 \tag{23}$$

and substituting into the axial momentum equation

$$u_z = \frac{r\rho - f_0 - f_1 - f_3 - 2(f_4 + f_7 + f_8)}{r\rho} \tag{24}$$

Again using the same assumption for f_2

$$f_2 = f_4 + \frac{2r\rho}{3} u_z \tag{25}$$

which leads to the same equation for the two remaining PDFs when using the radial momentum equation

$$f_5 = \frac{r\rho u_z}{6} + \frac{(r^2+\nu)\rho u_r}{2r} - \frac{\rho}{12} + f_7 + \frac{f_3 - f_1}{2} \tag{26}$$

$$f_6 = \frac{r\rho u_z}{6} - \frac{(r^2+\nu)\rho u_r}{2r} + \frac{\rho}{12} + f_8 - \frac{f_3 - f_1}{2} \tag{27}$$

2.2 Finite Volume Scheme for Particle Transportation

An obvious limitation in using LBM to solve the convection–diffusion equation is the case of low diffusivity. One of the simulations conducted in this paper has diffusion coefficient of the order 10^{-12}, which gives a relaxation time of $1/(3D + 0.5)$ close to 0.5, which makes the model unstable. To overcome this limitation with LBM it would require a very fine grid which, in turn, requires tiny time-steps, thus drastically increasing the computational cost. To address these limitations, the finite volume method is used to solve the convection–diffusion equation, which is capable of accommodating the small value of diffusion coefficient without major drawback.

The convection–diffusion equation is discretised by integrating each term of the convection–diffusion equation over the control volume V of each node, which can be rearranged as follow

$$\int_V \frac{\partial \phi}{\partial t} dV + \int_V \nabla \cdot (\mathbf{u}\phi) dV = \int_V \nabla \cdot (D \nabla \phi) dV \tag{28}$$

By using the divergence theorem, the two integrals are changed to surface integrals over the surface Γ of the control volume V

$$\int_V \frac{\partial \phi}{\partial t} dV + \int_\Gamma \mathbf{n} \cdot (\mathbf{u}\phi) d\Gamma = \int_\Gamma \mathbf{n} \cdot (D \nabla \phi) d\Gamma \tag{29}$$

where $\mathbf{n}$ is the normal to Γ.

In order to discretise in time, the equation is further integrated in time, from time t to time $t + \Delta t$.

$$\int_t^{t+\Delta t} \int_V \frac{\partial \phi}{\partial t} dV dt + \int_t^{t+\Delta t} \int_\Gamma \mathbf{n} \cdot (\mathbf{u}\phi) d\Gamma dt = \int_t^{t+\Delta t} \int_\Gamma \mathbf{n} \cdot (D \nabla \phi) d\Gamma dt \tag{30}$$

The first term is discretised to $2\pi r\Delta_r\Delta_z(\phi_{t+\Delta t}-\phi_t)$, where Δ_r and Δ_z are radial spacing and axial spacing. The other two terms are discretised in time in the form of

$$\int_t^{t+\Delta t}\phi dt = [\theta\phi_{t+\Delta t}+(1-\theta)\phi_t]\Delta t \tag{31}$$

Here $\theta = 0$ results in an explicit scheme, while $\theta = 1/2$ gives the Crank Nicholson scheme[49], and $\theta = 1$ gives a fully implicit scheme. Fully implicit scheme is unconditionally stable[50] whereas the other schemes have a restriction on the size of time-step for the scheme to be stable. In the current work, the explicit scheme is used. This choice is due to the ease of parallelisation, as explicit schemes avoid the need for the solution of systems of equations which reduces the efficiency of the parallel implementation.

The surface integrals are evaluated using values at the centre of each edge. The equation for each node can be described as

$$\begin{aligned} &V(\phi^P_{t+\Delta t}-\phi^P_t)/\Delta t = \\ &\Gamma_n u^n_r\phi^n - \Gamma_s u^s_r\phi^s + \Gamma_w u^w_z\phi^w - \Gamma_e u^e_z\phi^e \\ &+\Gamma_n D\frac{\partial\phi^n}{\partial r} - \Gamma_s D\frac{\partial\phi^s}{\partial r} + \Gamma_e D\frac{\partial\phi^e}{\partial z} - \Gamma_w D\frac{\partial\phi^w}{\partial z} \end{aligned} \tag{32}$$

where $V = 2\pi r\Delta_z$, $\Gamma_n = 2\pi(r-\Delta_r/2)\Delta_z$, $\Gamma_s = 2\pi(r+\Delta_r/2)\Delta_z$, $\Gamma_e = \Gamma_w = 2\pi r\Delta_r$. The notations n, e, s, w are illustrated by figure 3, which denotes the points on the middle of each side of the boundary. Velocity at the boundary of each node is obtained by linear interpolation of values obtained from LBM, and the differentials of ϕ are evaluated with the values of the nearby nodes.

In particular, we are using the upwind scheme in this model, which means values of ϕ are chosen according to the velocity of the fluid, and can be summarised as

$$\begin{aligned} \phi^w &= \begin{cases} \phi^W & \text{if } u^w_r > 0, \\ \phi^P & \text{if } u^w_r < 0 \end{cases} \quad \phi^e = \begin{cases} \phi^P & \text{if } u^e_r > 0, \\ \phi^E & \text{if } u^e_r < 0 \end{cases} \\ \phi^n &= \begin{cases} \phi^N & \text{if } u^n_z > 0, \\ \phi^P & \text{if } u^n_z < 0 \end{cases} \quad \phi^s = \begin{cases} \phi^P & \text{if } u^s_z > 0, \\ \phi^S & \text{if } u^s_z < 0 \end{cases} \end{aligned} \tag{33}$$

The choice of an upwind scheme instead of a central difference scheme is due to the expected Peclet number of the flow. The Peclet number is defined as

$$Pe = \frac{\text{convective transport rate}}{\text{diffusive transport rate}} \tag{34}$$

When $|Pe| > 2$ the central difference scheme may lead to physically impossible situations such as negative concentration which lead to instability, while upwind scheme ensures stability at the cost of reduction in the spacial accuracy. The example presented later has $Pe = O(10^6)$, which makes upwind scheme the clear choice throughout the domain. It was previously shown[27] that using an upwind scheme for the solution of the convection-diffusion equation will result in a first order accurate scheme, which is still appropriate for the current application.

The current work focuses on microfiltration and ultrafiltration, where the bulk of the flow does not change much during filtration. As such, concentration polarisation and other fouling mechanisms happen in the vicinity of the membrane and we focus the simulation on that region. Solute concentration is resolved only over a small portion of the domain near the membrane as shown in figure 4. The figure shows a schematic of the grids used. The finite volume grid with spacing of Δ_r and Δ_z, shown in red, is used in the region above the membrane nodes, shown in green, and overlays the fluid nodes which has a spacing of δ_r and δ_z, shown in blue. The remainder of the domain is assumed to have the bulk concentration. The selected height for the solution of convection-diffusion equation is refined in the radial

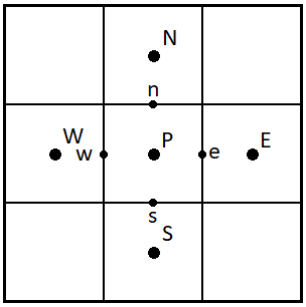
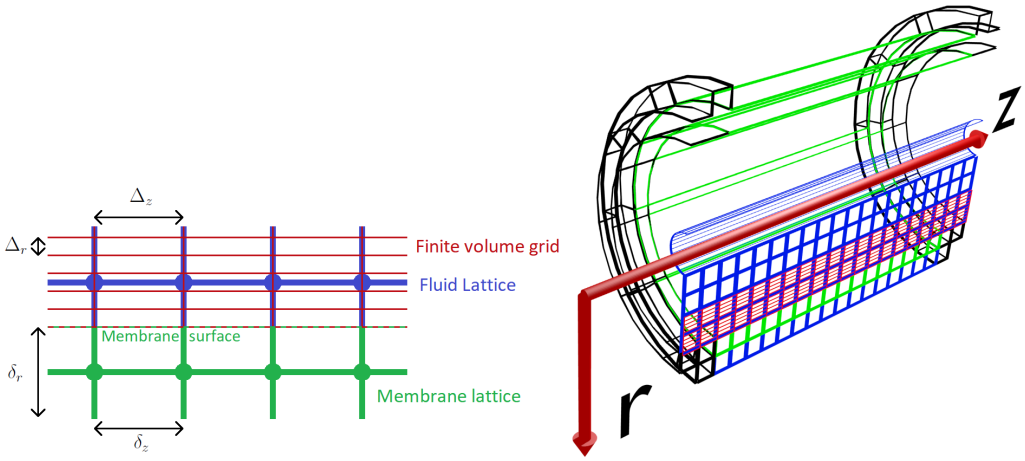


Figure 3: Illustration of notations used in finite volume equation. P is the centre of current node, W,N,E,S denotes centre of neighbouring nodes and w,n,e,s denotes centre of the boundary between current and neighbouring nodes.



(a) (b)

Figure 4: a) Schematic of the finite volume grid(red) and the lattice Boltzmann grid(blue and green). b) 3D illustration of simulation grids overlaid on a cross-section of tubular filtration set-up.

direction. Refinement of the mesh in the vicinity of the membrane results in many solute control volumes within a single fluid node as shown in figure 4.

A time-step of the FV scheme may differ from a time-step of the LB scheme, depending on the problem and stability of the solution, thus saving computational resource from unnecessary calculations. In order to maintain stability, the coefficient of each ϕ term from equation 32 must be positive. By putting in the upwind scheme, assuming positive velocity in both direction and ignoring the diffusion terms(which are small in our axisymmetric example), we can rearrange the equation to

$$V\phi^P_{t+\Delta t}/\Delta t = \Gamma_n u^n_r \phi^N + \Gamma_w u^w_z \phi^W + [V/\Delta t - \Gamma_e u^e_z - \Gamma_s u^s_r]\phi^P \tag{35}$$

where the left hand side and the first two terms on the right hand side are positive, and the coefficient of ϕ^P gives the stability condition of

$$V/\Delta t > \Gamma_e u^e_z + \Gamma_s u^s_r \tag{36}$$

assuming $r >> \Delta_r$, the value of the allowable time step that ensures stability is given by

$$\Delta t < (\frac{\Delta_r}{\Delta_z} u^e_z + u^s_r)^{-1} \tag{37}$$

2.2.1 Boundary Conditions for Finite Volume Method

In comparison with the BCs for LBM, the BCs for FV are considerably simpler as the values required at the boundary are the macroscopic variables rather than partial distribution functions.

Take a mirror wall at $r = 0$ for example, there is no convection term nor diffusion term due to symmetry. If we take the north side as $r = 0$, then the terms $2\pi(r - \Delta_r/2)\Delta_z u_r^n \phi^n$ and $D2\pi(r - \Delta_r/2)\Delta_z \frac{\delta\phi^n}{\delta r}$ will vanish. Similarly for solid walls, the appropriate convection term and diffusion term are modified accordingly.

As for inlet and outlet, the value of ϕ and its gradient are set appropriately. For the inlet we set $\phi = \phi_0$ as we assume the flow to have constant solute concentration; for the outlet, the gradient of ϕ normal to the boundary is set to 0.

2.3 Pore Blocking and Cake Formation Model

Pore blocking model is implemented by applying a probability term, P which calculates the probability of pores being free after each time-step, inspired by Starov et al.[51] and developed by Kim[27].

To calculate the rate of particles reaching the membrane surface, we calculate the volume of particles in a unit volume, V_c. Conveniently this is ϕ. Then by considering the volume of a single particle V_s, we can calculate the number of particles in a unit volume, N_c

$$N_c = \frac{V_c}{V_s} \tag{38}$$

from this we can find the number of particles per lattice spacing $\sqrt[3]{N_c}/\delta_r$. Using the local velocity normal to the membrane surface, u_n, the rate of particle reaching the membrane surface, r_p, is

$$r_p = u_n \sqrt[3]{N_c}/\delta_r^2 \tag{39}$$

This is used to obtain the probability of a pore being free, P,

$$P = 1 - (\beta r_p N_p^t A_p) \tag{40}$$

where β is determined from experimental data, A_p is the cross-sectional area of a pore and N_p^t is the number of free pores at time t. Thus with P and N_p^t,

$$N_p^{t+1} = P N_p^t \tag{41}$$

Using Hagen-Poiseuille equation which describes the flux through a membrane as

$$J = \frac{\epsilon_m a_p^2 \Delta P}{8\mu\tau\delta_m} \tag{42}$$

where τ is the tortuosity of a pore which is taken as 1 in current work, a_p is the radius of a pore and δ_m is the membrane thickness. ϵ_m can be formulated as the ratio between pore area and membrane surface area,

$$\epsilon_m = \frac{\pi N_p a_p^2}{A_m} \tag{43}$$

where A_m is the surface area of the membrane. Thus equation 42 becomes

$$J = \frac{\pi N_p a_p^4 \Delta P}{8\mu\tau A_m \delta_m} \tag{44}$$

By equating the above equation with the common membrane equation for microfiltration,

$$J = \frac{\Delta P}{\mu R_m} \tag{45}$$

we arrive at the resultant membrane resistance R_m

$$R_m = \frac{8\tau\delta_m A_m}{\pi N_p^t a_p^4} \tag{46}$$

The pore blocking module is implemented as a boundary condition and updates the membrane resistance every time-step. Each membrane node along the membrane surface has its unique N_p^t and is used to compute the resistance at that lattice node. In the case that the membrane is thicker than one node then the nodes beneath have their resistance adjusted according to the top node.

When the concentration of solute particles reaches ϕ_{max}, the node is treated as an incompressible cake formation, and the pore blocking mechanism stops. The cake formation is treated as incompressible due to simplicity, which can be modified accordingly in the future for scenarios where cake compaction is a factor. Resistance of the cake formed can be calculated by the Carmen–Kozeny relationship, defined as

$$R = \frac{45(1-\epsilon)^2}{\epsilon^3 a^2} \tag{47}$$

where ϵ is the cake porosity which would be $1-0.64$ here and a is the radius of the particles. In this paper the examples do not exhibit this mechanism as the particle concentration does not reaches 0.64, however the validation can be referred to work by Kim[27].

3 Parallelisation

The schemes are parallelised in two ways, by using CUDA and MPI, for GPU cluster and CPU cluster respectively. This section aims to give an overview of the parallelisation and optimisation techniques used.

LBM are generally memory bound[39], however this changes depending on the collision operator, additional calculations (such as dynamically updated viscosity and diffusion coefficient) and the hardware used. Double precision in particular is an issue for GPUs which are generally designed for single precision calculations. The ratio between CUDA cores to double precision cores is 2:1 for scientific GPUs and up to 24:1 for other GPUs, which means that while the dataset doubles in size, the speed of GPU computing is generally much less than half. Thus the increase in compute time is considerably higher when using double precision and can change the bottleneck of the simulation. For CPUs this is generally not an issue as the single precision to double precision performance is usually 2:1.

3.1 Memory Management

In a simple LB scheme the collision step and streaming step are implemented as individual functions as shown in figure 5. The collision function reads data from the random access memory(RAM), performs collision calculations, and then saves outputs in the RAM. The streaming function reads data from the RAM, and then saves data to their new location in the RAM.

It is obvious then that streaming is no more than a read and write and can be combined with collision to minimise data transfer. However streaming cannot be combined in a simple manner as it would cause new data to overwrite old data that is still required for nearby nodes. Thus one way to avoid this is to have two sets of memories, memory space A for odd time-steps and memory space B for even time-steps; the usage of two memory spaces is also referred as "AB pattern". When the processor finishes collision for a node it stores data in the other memory space, which is holding data two time-steps before, therefore does not cause any memory clashes. Whilst this is a possible solution, requiring twice the memory is not feasible for large grids.

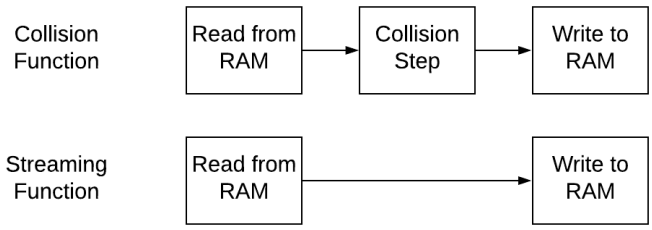


Figure 5: Illustration of collision function and streaming function.

To combat this issue Bailey et al.[52] proposed a method to combine the two processes such that streaming is done by accessing the same set of memory in one time-step. The usage of a single memory space gives the name "AA pattern" as it reads from memory space A and writes to memory space A. AA pattern has two different schemes for odd time-steps and even time-steps, as illustrated in figure 6. In odd time-steps, data for the node with position $\mathbf{x}$ is read normally from $f_i[\mathbf{x}]$ and after the collision step it is stored in the same set of memory but swapped pairwise with the PDF with opposite velocity $f_{\bar{i}}[\mathbf{x}]$, i.e f_1 is stored at where f_3 was and vice versa, f_2 is swapped with f_4, etc. In even time-steps, the processor pulls data from nearby cells; for the ith element the data is read from $f_{\bar{i}}[\mathbf{x} - \mathbf{c}_i]$, i.e f_1 from the left cell, f_3 from the right cell etc. Afterwards data is swapped pairwise like before and stored in $f_i[\mathbf{x} + \mathbf{c}_i]$, f_3 is now stored in the left cell, f_1 in the right cell, thus pushing the PDFs out to achieve streaming.

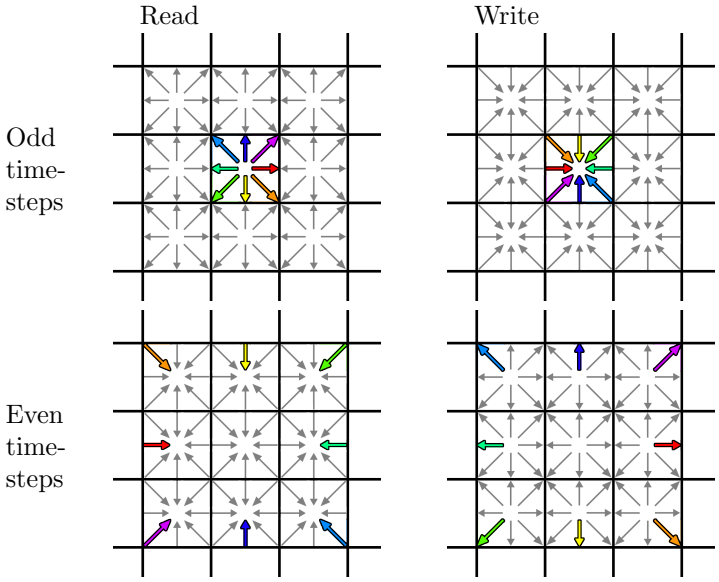


Figure 6: Steaming using AA pattern. Top left: odd time-steps load partial distribution functions(PDFs) from current position. Top right: odd time-steps stores PDFs in current position and swapped pairwise. Bottom left: even time-steps load swapped PDFs from neighbouring positions. Bottom right: even time-steps store PDFs to neighbouring positions.

Intuitively there are two types of memory structures for LBM, array of structures(AoS) and structure of arrays(SoA). AoS means that all PDFs of the first node are stored together then all PDFs of the second node are stored after, etc. SoA means that f_0 of the entire

Table 1: Time taken(ms) for each operation with MPI implementation and CUDA implementation, using 1024×10240 grid. Speed up factor when compared to single core performance is also shown.

	collision	streaming	combined step
1 core	303.9	71.9	311.9
2 cores	152.4 (1.99x)	37.4 (1.92x)	157.7 (1.98x)
4 cores	84.1 (3.61x)	23.4 (3.07x)	86.2 (3.62x)
8 cores	45.2 (6.7x)	21.4 (3.36x)	46.6 (6.69x)
14 cores	28.5 (10.6x)	21.0 (3.42x)	29.0 (10.8x)
CUDA 2880 cores	20.3	6.1	20.6

lattice are stored in an array, and f_1 of all nodes are stored afterwards or in a separate array, etc.

For GPUs the choice is obvious as SoA will automatically give coalesced memory access, whereas AoS will have a warp loading data from various segments of the memory. For CPUs, Wittmann et al.[53][54] presented detailed tests on memory structures with different streaming methods. It is shown that the "AA pattern" has the best performance out of the streaming implementations and the vectorised AA pattern using SoA has better performance than using AoS. As a result SoA is chosen for both versions in the current paper. The AA pattern streaming method which was introduced by Bailey et al.[52] is adopted and explained below. Alternatively there are other structures such as those proposed by Calore et al.[55] which aims to exploit data locality for more complex layouts.

In order to measure the efficiency of the AA pattern compared to the traditional scheme, the time taken for each operation and the speed up of using multiple cores or GPUs was performed. Each operation was tested with a 1024×10240 grid on an Intel Xeon E5645(using up to 14 cores); a similar test was performed on a NVIDIA Titan Z(using 1 of 2 GPUs within). Table 1 contains details of the results and speed up when using multiple cores of the CPU and the speed up when using the GPU. As the number of CPU cores used increases, the speed up of streaming levels off very quickly, indicating that the memory bandwidth is fully utilised and having multiple processors requesting data do not have any positive impact. Collision shows that by increasing the number of cores, the time taken to complete this operation scales linearly. The use of the AA pattern that combines collision and streaming operation has very similar times and speed up when compared to collision step, and it is up to 75% more efficient than having two separate operations.

The AA pattern resulted in a 40% speedup on GPUs when compared to the traditional scheme. However, the ratio of computation time to memory transfer time shows that the GPU spends most of the time on computation and the combined step has smaller benefits.

3.2 Grid Sub–division

As the algorithm is designed to perform simulations on a cluster of CPUs/GPUs, the workload must be divided between processors. With a 2D rectangular domain it is then logical to divide it in one of two possible ways: along the longest axis, or along both axes. The former minimises the number of neighbours to a maximum of 2, which minimises the communication overhead but has a bigger set of data to communicate, due to longer boundaries; the latter divides the domain such that the boundary of each subdomain is minimised, which minimises the size of messages but has up to 8 neighbours to communicate with. This is due to diagonal PDFs which require communication with diagonal neighbouring block.

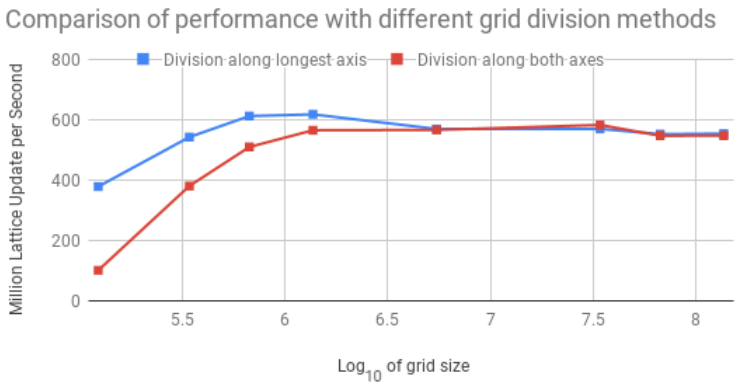


Figure 7: Comparison of performance with different grid division methods using various grid sizes, measured in million lattice update per second. Grid size varies from 76×180 to 7600×18000

The method of dividing the simulation domain between processors was investigated by using 10 6-core Intel Xeon E5645 CPUs with increasing domain size. The problem solved in this simulation was proposed by Kromkamp[26]. Figure 7 shows the million lattices updates per second(MLUPS) of the simulations, plotted against the logarithm of grid size. Grid size is defined as the number of lattice nodes in the grid, which is 76×180 for the smallest one and 7600×18000 for the largest one. For smaller domains the communication overhead is significant to the overall runtime and therefore dividing along the longest axis is more beneficial. On larger domains collision and streaming take considerably more time than communication so the benefits of having shorter messages is negligible. Thus it is better to divide the domain along the longest axis, for better performance in small grids and ease of programming. In particular for GPUs, packing and sending communication messages is a relatively slow operation, which further favours division along the longest axis.

4 Simulation Algorithm

Coupling the lattice Boltzmann method and the finite volume method has been carried out in the past[28][56][57], the procedure involves passing interpolated or averaged data between the two systems. In particular, the coupled scheme used here is based on Paipuri[28] and is extended to axisymmetric form. The LB scheme requires the volume fraction of solute particles from the FV scheme to calculate the local kinematic viscosity. As each LB cell contains multiple FV nodes, the average volume fraction is used in the computation of the viscosity.

Algorithm 1 describes the implementation of the coupling technique. Firstly, the LB scheme reads the particle concentration from the FV scheme and updates the local viscosity. The fluid equations are then solved for a number of time-steps. The resultant velocity is passed to the FV scheme, which updates the diffusion coefficient and solves for the volume fraction of solute particles.

Algorithm 2 describes the lattice Boltzmann scheme in detail. The scheme begins by checking the state of the lattice node and then setting the correct resistance and viscosity. Afterwards, calculations of the MRT collision operator are carried out and outputs are saved according to AA pattern(explained below) to complete the streaming step. When all nodes are processed, communications and boundary conditions are applied and then advance to the next LB time-step or pass the data to the particle solver.

Algorithm 3 describes the finite volume scheme in detail. The solver computes the velocity from data obtained by the fluid solver, which is used in the upwind scheme to

for *iter* $\leftarrow 1$ **to** *finite volume steps* **do**
  for *iter* $\leftarrow 1$ **to** *lattice Boltzmann steps per finite volume step* **do**
    Update viscosity
Fluid solver
    Solve Navier–Stokes equations for fluid flow
    Update membrane resistance with pore blocking model
    Calculate flux through membrane
  end
  Calculate velocity field from lattice Boltzmann scheme
Particle solver
  Compute macroscopic variables and diffusion coefficient
  Solve convection–diffusion equation to obtain particle volume fraction
  Calculate particle volume fraction for lattice Boltzmann scheme
end

Algorithm 1: Coupling of fluid and particle solvers using lattice Boltzmann–finite volume scheme

determine values of ϕ. The diffusion coefficient is calculated and terms are modified if boundary conditions are applicable. After the computation of the new ϕ, the state of each node is checked for cake formation.

5 Validation and Results

In this section we first validate the accuracy of the chosen numerical scheme, then apply the technique to the simulation of a practice ceramic filtration problem. We finally use the scheme in predictive mode to investigate possible variations the geometry in order to improve efficiency of the filtration process.

5.1 Poiseuille Flow in Annular Section

A simple validation test for the fluid part of the filtration model is Poiseuille flow. Consider a flow in an annulus, driven by a constant pressure difference Δp between two ends and fluid flows between an inner and outer cylinder with radii R_1 and R_2 respectively. The velocity profile can be solved analytically and is given by

$$u(r) = -\frac{\partial p}{\partial z}\frac{1}{4\mu}(R_1^2 - r^2) - \frac{\partial p}{\partial z}\frac{1}{4\mu}(R_2^2 - R_1^2)\frac{ln(r/R_1)}{ln(R_2/R_1)} \tag{48}$$

Annulus was modelled by a grid of $nr \times nz$ points, where $nz >> nr$, $R_1 = nr$, $R_2 = 2nr$, $\nu = 0.016667$. The pressure gradient varied according to the number of lattice nodes used such that the Reynold's number $Re = uL/\nu = 15$ remains constant. The maximum relative error in velocity is defined as[48]

$$err = max\frac{\sqrt{(u_z^{ex} - u_z)^2 + (u_r^{ex} - u_r)^2}}{u_0} \tag{49}$$

where $\mathbf{u}^{ex}$ is the analytical velocity and u_0 is the peak velocity. Figure 8a shows the simulation result (red cross) and the analytical solution (blue line), and figure 8b shows a 2D plot of axial velocity of a segment of the annulus. Table 2 shows the maximum relative errors when using different nr and nz values. The ratio between simulations show that the LB scheme is 2nd order accurate, which is the expected order of accuracy.

5.2 Tube with Sinusoidal Dirichlet Condition

In this example, the accuracy of the solute solver is tested. Fluid flow inside a pipe of radius R_0 and is assumed to be a plug flow with constant axial velocity U and zero radial velocity.

```
begin
    for node ← 1 to lattice Boltzmann size do
        Read φ and flag
        if φ > 0.64 then
            Flag node as cake
        end
        if node is membrane or cake then
            Compute resistance R
        else
            Set R = 0
            Compute kinematic viscosity ν
        end
        Read f_0, ..., f_8 according to AA pattern scheme
        Compute m_1, m_2, m_4, m_6, m_7, m_8
        Compute macroscopic variables
        Compute force terms and equilibrium terms
        Compute new m_1, m_2, m_4, m_6, m_7, m_8
        Compute new f_0, ..., f_8
        Save f_0, ..., f_8 according to AA pattern scheme
    end
    Communicate with other processors
    Perform boundary conditions
    Update membrane resistance with pore blocking model
    Calculate flux through membrane
end
```

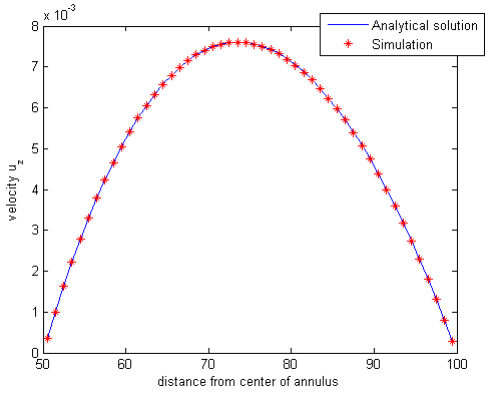
Algorithm 2: Detailed fluid solver using lattice Boltzmann scheme.

```
begin
    for node ← 1 to finite volume size do
        Compute u from lattice Boltzmann scheme
        Read φ according to upwind scheme
        Compute shear rate dependent diffusion coefficient D
        Compute finite volume terms
        if node has boundary condition then
            Modify finite volume term
        end
        Compute new φ
        if φ > 0.64 then
            Flag node as cake
        end
    end
    Communicate with other processors
    Compute φ for lattice Boltzmann scheme
end
```

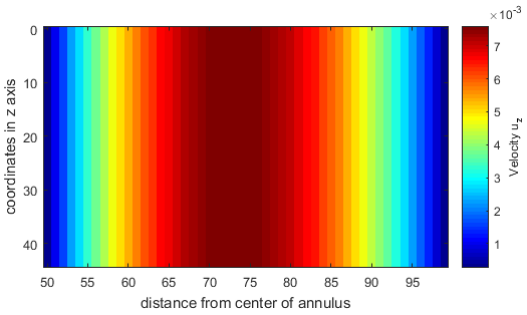
Algorithm 3: Detailed particle solver using finite volume scheme.

Table 2: Maximum relative errors for Poiseuille flow in annular section, simulated with a lattice of $nr \times nz$ nodes. The error ratio of the smallest grid to the four grids are presented to indicate order of accuracy.

nr	50	100	200
nz	1200	2400	4800
err	4.0215(-4)	9.9981(-5)	2.4926(-5)
ratio	1	4.0222	16.1334



(a) Plot of simulation result and analytical solution.



(b) Plot of velocity in a segment of the annulus.

Figure 8: Plot of Poiseuille flow in annular section, simulated using 50×1200 grid.

Table 3: L-2 norm errors of tube with sinusoidal Dirichlet condition on different grid resolutions $nr \times nz$. The error ratio of the smallest grid to the four grids are presented to indicate order of accuracy.

nr	100	200	400
nz	200	400	800
err	7.6651(-3)	3.8317(-3)	1.9273(-3)
ratio	1	2.0005	3.9770

Periodic boundary conditions are applied on the two ends, a mirror boundary condition for the axis of symmetry and a sinusoidal Dirichlet condition is imposed on the pipe wall

$$\phi(z) = cos(2\pi z/L), \text{at} \quad r = R_0 \tag{50}$$

The analytical solution for ϕ is[58]

$$\phi(r, z) = Re[e^{ikz} \frac{I_0(\lambda r)}{I_0(\lambda R_0)}] \tag{51}$$

where I_0 is the modified Bessels function of the first kind of order 0, $\lambda = k\sqrt{1 + \frac{iU}{Dk}}$ and $k = 2\pi/L$.

For this example $D = 1/120$ and $U = 0.01$, a grid of 50×100 finite volume nodes was used and the solution is advanced until steady state was reached. Subsequent simulations were scaled accordingly to investigate the order of accuracy, measured by the L-2 norm error which is defined as

$$E = \left[\sum(\phi - \phi_{ex})^2 / \sum \phi_{ex}^2\right]^{1/2} \tag{52}$$

Table 3 shows the error of simulations and confirm that that the FV upwind scheme is first order accurate. Figure 9 illustrates the distribution of the solute concentration within the domain.

5.3 Ceramic Membrane Filtration

The experimental work carried out by Ogunbiyi[59] is used to validate the current scheme. Tubular ceramic membrane was used in this experiment and is made of alumina(70%),

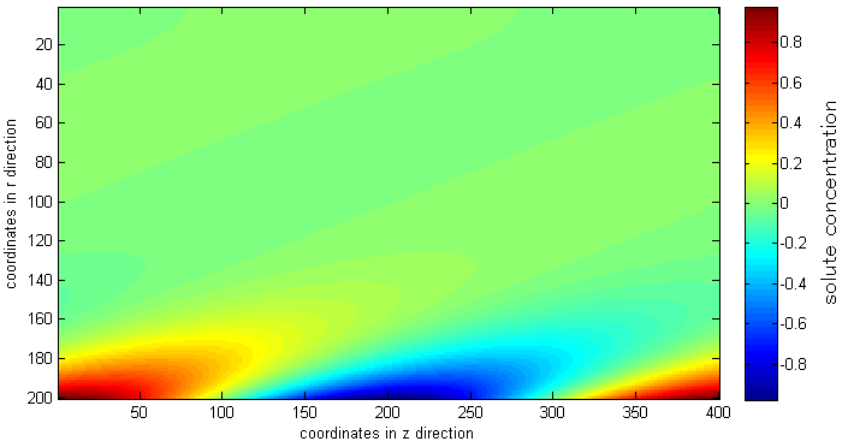


Figure 9: Simulation result of solute concentration of tube with sinusoidal Dirichlet condition, using 200×400 grid.

zirconia(25%) and yttria(5%), with nominal pore size of $0.5\mu m$ and porosity of 50%. The membrane was suspended horizontally in a membrane module during the experiment. The feed used in this experiment was bentonite suspension, with size of particles found to be in the region of $2-10\mu m$. The tube has an inner radius of $5mm$, thickness of $1mm$ and $210mm$ in length. Suspension was prepared at $0.05g/L$ and was fed into the tube at $1.2L/min$ with $2.5bar$ pressure. This particular experiment was chosen due to the high pressure used and made pore blocking the dominant fouling mechanics. The pure water flux was recorded as $421.2L/m^2h$ at $1bar$ pressure, and by using the basic filtration equation, the membrane resistance was found to be $R_m = 8.55 \times 10^{11}$; the value of β for the pore blocking model was found to be 2.04×10^{-3}.

Bentonite clays are used in many industries such as waste absorbent, drilling muds, pelletising iron ore, foundry sand and adhesives.[60] They are also used to form a cake on walls of rock formation to prevent the collapse of pit walls during drilling by stopping water flow.[61] In particular, permeability of the cake is important as too high will lead to excess water passing through and too low will lead to increased thickness and makes drilling difficult to manoeuvre. Therefore knowledge of the filtration properties of a bentonite clay is vital to determine if its suitability.

The Reynold's number of this experiment is of the order $Re = O(2000)$. It is expected that changes of the flow field will be minimal over the length of the tube. Hence, a represented length of the tube will be used in the current simulation. Also, since our interest is in the concentration polarisation that occur close to the boundary of the ceramic membrane, it is possible to assume that the concentration of pollutant will remain constant away from the boundary.

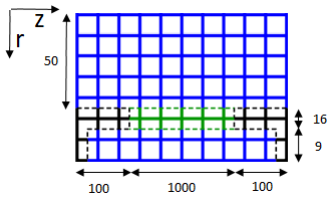


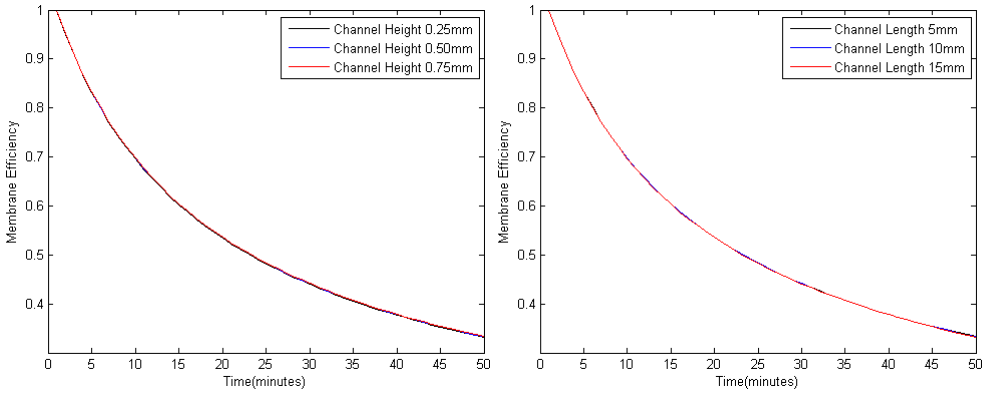
Figure 10: Schematic of the domain set–up for $10mm$ membrane's length and $0.5mm$ height above the membrane. Blue represents fluid nodes, black represents wall nodes and green represents membrane nodes.

However, in order to justify our assumptions, two set of meshes were generated. The first set of meshes are for three domains that include the membrane, a constant membrane's length of $10mm$ and heights above the membrane of $0.25mm, 0.5mm$ and $0.75mm$. The second set of meshes are for three domains that include the membrane, a constant height above the membrane of $0.5mm$ and membrane' lengths of $5mm, 10mm$ and $15mm$ respectively. Mesh spacing of 100 points per mm was used for all meshes. Figure 10 shows a schematic of the grid used for the simulation of $10mm$ membrane's length and $0.5mm$ height above the membrane. The upper part of the domain represents fluid nodes above the membrane and has 50 nodes representing $0.5mm$ in height; this is modified to 25 and 75 nodes for height above the membrane of $0.25mm$ and $0.75mm$ respectively. The lower section represents the membrane and permeate chamber, which consists of 25 nodes in height for all simulations. A $1.0mm$ solid wall was placed at the start and end of the membrane to smoothen the effect of boundary conditions. In these simulations, each LB time–step is $\delta_t = 600,000^{-1}s$ and each FV time-step is $\Delta_t = 300,000^{-1}s$, together with the length scale this gives an intial kinematic viscosity $\nu = 0.0167$ in LB scheme.

The permeate flux measured in the experiment is scaled in relation to the flux after one minute, and is used to indicate the membrane efficiency; the permeate flux for simulations are scaled similarly. Membrane efficiency can be calculated as

$$\text{membrane efficiency at time t} = \frac{\text{permeate flux at time t}}{\text{permeate flux at 1 minute mark}} \tag{53}$$

Figure 11a shows the membrane efficiency for different domain's height and figure 11b shows the membrane efficiency for different domain's length. The efficiency measured on the smallest domain of each group are marginally different to the efficiency measured on the other two grids. This justify the use a domain which is $10mm$ long and has a height of $0.5mm$ above the membrane.



(a) Variable height (b) Variable length

Figure 11: Plots of the simulation results for various domain sizes. Membrane efficiency is permeate flux normalised to permeate flux at 1 minute mark.

Figure 12 shows a comparison between the simulation result and the experimental result, with the filtration flux normalised to the initial flux(measured at first minute mark). Simulation is in good agreement with the experiment with the exception of the drop in the efficiency at the 5.0 minute mark. The addition of the solid wall at the start and end of the membrane would have resulted in a local disturbance in the flow pattern which may have reduced the initial pore blocking and resulted in a higher efficiency.

To demonstrate the predictive capability of the developed scheme, a study of the effect of changing the geometry of the ceramic tube is carried out. Two new scenarios were designed;

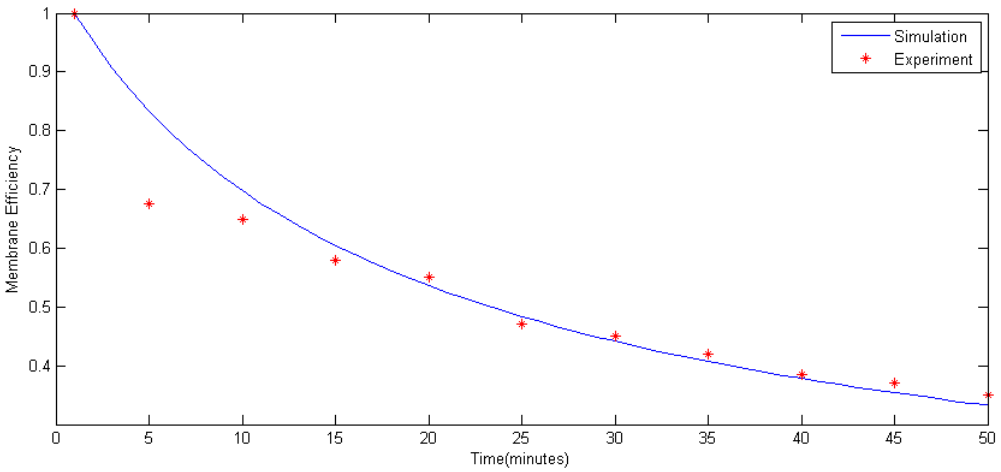
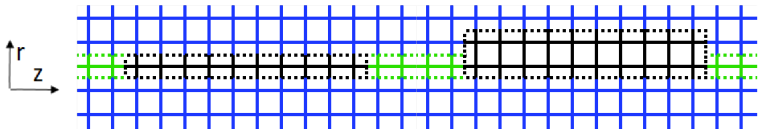


Figure 12: Plot of ceramic membrane filtration simulation result and experimental result. Membrane efficiency is permeate flux normalised to permeate flux at 1.0 minute mark.

the first case involved the addition of uniform non-permeable inter–spaces, while the second case replaces the inter–spaces with small steps. The addition of inter–spaces and steps are illustration in figure 13. The additional four inter–spaces of $0.1mm$ spread evenly on the analysed membrane were treated as walls. The steps that replaced the inter–spaces have a height of $0.01mm$.

Figure 14 shows the membrane efficiency of the original simulation(blue), with inter–spaces(black) and with steps(red). The membrane efficiency of the two new simulations were scaled using the flux after one minute of the original membrane. Efficiency of the membrane with inter–spaces declines at the same rate as the original membrane, ending with efficiency of 0.32 compared to 0.33 for the original membrane. This indicates that the benefit that results from disturbing the flow by separating the membrane into smaller sections does not overcome the reduction of the effective area of the membrane. However, the addition of steps resulted in a lower flux decline than the original simulation, with 0.38 membrane efficiency at the end of simulation. Figure 15 shows a plot of volume fraction in the vicinity of a step; particles flow from left to right and are concentrated on the surface of membrane. The step disperse particles by promoting turbulence, which alleviates the effect of concentration polarisation on the right hand side of the step. This shows that the step can be a useful device for longer filtration processes to maintain higher steady permeate flux.



(a) Membrane separated by inter–space. (b) Membrane separated by a step.

Figure 13: Illustration of membrane with the addition of inter–space and step. Blue represents fluid nodes, green represents membrane nodes and black represents inter–space and step.

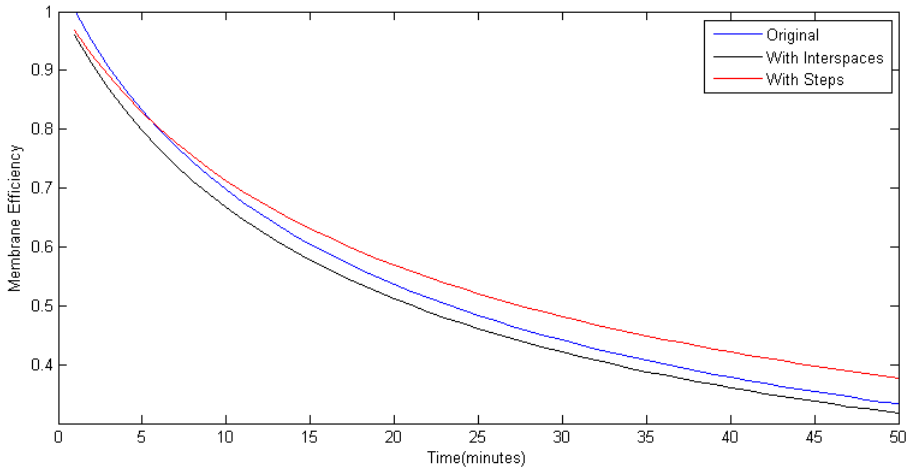


Figure 14: Plot of simulation results with the addition of inter–space between membranes and with the addition of step between membranes. Membrane efficiency is permeate flux normalised to permeate flux of the original filtration at 1 minute mark.

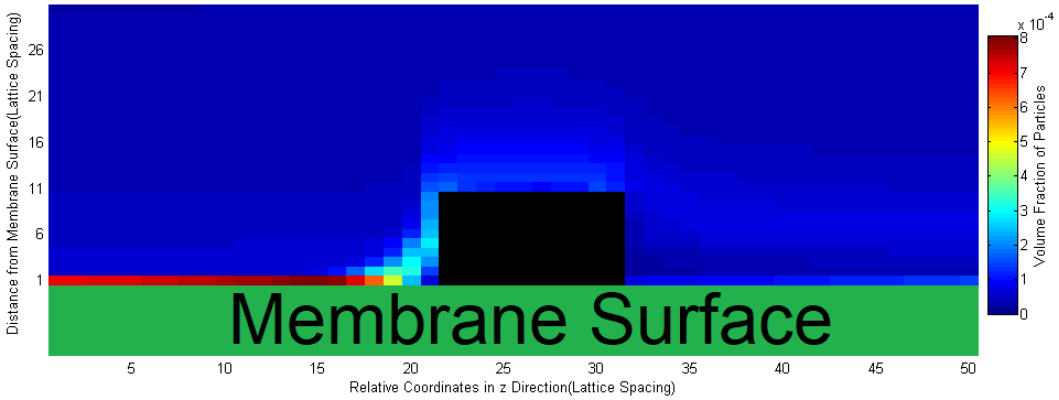
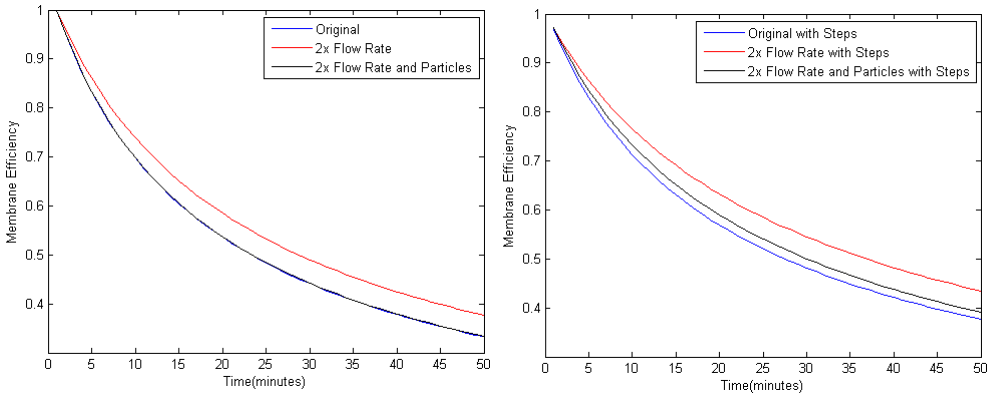


Figure 15: Plot of volume fraction of particles in the vicinity of a step. Black space represents the step, with particles flowing from left to right.

The simulation set-up is further modified by changing the flow rate ($2.4L/min$) and the particle concentration ($0.1g/L$ bentonite solution) with and without the additional steps. Figure 16a shows the simulation results, with the blue line representing the original simulation, the red line representing simulation with double flow rate, and the black line representing simulation with double flow rate and double particle concentration. Without the addition of steps, using double flow rate is expected to have higher membrane efficiency due to the increase of the flow velocity and, hence, reduction in the concentration polarisation. Doubling the flow rate and the particle concentration resulted in a similar efficiency to the original set-up. This shows that the increase in the shear rate due to the increase of the flow velocity did not have a significant effect on the overall efficiency. The efficiency that resulted from the inclusion of steps along the ceramic are shown in figure 16b. Simulation using double flow rate increased the efficiency by 14%, whereas doubling the flow rate and the particle concentration see membrane efficiency increased by 3%. Doubling the flow rate observes more substantial gain in efficiency than the original set-up, as the higher flow rate generates stronger turbulence to alleviate membrane fouling.



(a) Simulations without steps

(b) Simulations with steps

Figure 16: Plot of membrane efficiency of ceramic membrane with the addition of steps. Using the original set-up, with double flow rate, and with double flow rate and particle concentration.

In similar simulations[27][28], the time taken to execute a simulation using the original serial code is between real time for small simulations to a couple days for larger ones. With the help of parallelisation and optimisations, these can be done in minutes. These simulations have Reynold's number of order 1, the length and velocity scale allow generous time scale(up to 10^4 steps per second) to keep kinematic viscosity with a stable region. However the high Reynold's number of this experiment requires small values of δ_r and δ_t in order to keep the scaled kinematic viscosity within a stable region. Whilst the simulation domain can be kept reasonably small by simulating a section of the tube, the time scale cannot be simplified in a similar fashion and gives 6×10^5 steps per second; the harsh scaling makes the simulation cost hundred-fold higher than the previous simulations, which are impractical for a serial simulation code. The presented simulations requires 12 CPU hours using 12 14-core Intel Xeon E5-2680 v4 CPUs.

6 Conclusion

In this paper, a two-dimensional cross-flow membrane filtration model is modified to an axisymmetric model and parallelised for CPU or GPU cluster. The proposed model solves the incompressible Navier–Stokes equations for fluid flow and convection–diffusion equation for particle transport. These equations are solved in a staggered manner using an axisymmetric version of the D2Q9 lattice Boltzmann method for fluid and explicit finite volume scheme for particles. The lattice Boltzmann method was chosen due to its ability to cope with complex boundary, porous medium and ease of parallelisation on clusters. As for particle transport, the finite volume scheme was used in place of the lattice Boltzmann method due to ability to cope with extreme values and scaling appropriately for the boundary layer; since only first order accuracy is required this made an explicit finite volume scheme suitable. This coupled scheme allows the prediction of concentration polarisation and cake formation. With the addition of a pore blocking scheme, the model is also capable of predicting flux decline due to particles blocking the membrane pores. The individual modules of the model have been validated against analytical solutions for different scenarios and together the model has been validated against experimental data. The ease of using the developed model in a predictive mode was demonstrated and it showed that altering the geometry and the operating conditions can increase the efficiency of the filtration process.

The model can be further enhanced by adding particle interactions or organic interactions, or inclusion of osmotic pressure. Cake compression is also a factor that should be

considered for long term filtrations as the cake formation here is assumed to be incompressible.

References

[1] P. S. Goh, A. F. Ismail, A review on inorganic membranes for desalination and wastewater treatment, *Desalination* 434, 60 - 80. https://doi.org/10.1016/j.desal.2017.07.023

[2] M. Rosenberg, Current and future applications for membrane processes in the dairy industry, *Trends in Food Science & Technology*, 6(1), 12 - 19. https://doi.org/10.1016/S0924-2244(00)88912-8

[3] G. Daufin, J. -P. Escudier, H. Carrère, S. Bérot, L. Fillaudeau, M. Decloux, Recent and Emerging Applications of Membrane Processes in the Food and Dairy Industry, *Food and Bioproducts Processing*, 79(2), 89 - 102. https://doi.org/10.1205/096030801750286131

[4] S. Munirasu, M. A. Haija, F. Banat, Use of membrane technology for oil field and refinery produced water treatmentA review, *Process Safety and Environmental Protection*, 100, 183 - 202. https://doi.org/10.1016/j.psep.2016.01.010

[5] Y. Wang, X. Wang, Y. Liu, S. Ou, Y. Tan, S.Tang, Refining of biodiesel by ceramic membrane separation, *Fuel Processing Technology*, 90(3), 422 - 427. https://doi.org/10.1016/j.fuproc.2008.11.004

[6] D. F. Stamatialis, B. J. Papenburg, M. Gironés, S. Saiful, S. N. M. Bettahalli, S. Schmitmeier, M. Wesslinga, Medical applications of membranes: Drug delivery, artificial organs and tissue engineering, *Journal of Membrane Science*, 308 (12), 1 - 34. https://doi.org/10.1016/j.memsci.2007.09.059

[7] M. W. Chudacek, A. G. Fane, The dynamics of polarisation in unstirred and stirred ultrafiltration, *Journal of Membrane Science* 21 (2), 1984, 145 - 160. https://doi.org/10.1016/S0376-7388(00)81551-3

[8] M. C. Porter, Concentration polarization with membrane ultrafiltration, *Ind. Eng. Chem. Prod. Res. Dev.* 11 (3), 1972, 234 - 248. DOI: 10.1021/i360043a002

[9] A. L. Zydney, C. K. Colton, A concentration polarization model for the filtrate flux in cross-flow microfiltration of particulate suspensions, *Chemical Engineering Communications* 47 (1-3), 1986, 1 - 21. https://doi.org/10.1080/00986448608911751

[10] G. Belfort, R. H. Davis, A. L. Zydney, The behavior of suspensions and macromolecular solutions in cross-flow microfiltration, *Journal of Membrane Science* 96 (12), 1994, 1 - 58. https://doi.org/10.1016/0376-7388(94)00119-7

[11] Y. Lee, M. M. Clark, Modeling of flux decline during cross-flow ultrafiltration of colloidal suspensions, *Journal of Membrane Science* 149 (2), 1998, 181 - 202. https://doi.org/10.1016/S0376-7388(98)00177-X

[12] S. Hong, R. Faibish, M. Elimelech, Kinetics of Permeate Flux Decline in Cross-flow Membrane Filtration of Colloidal Suspensions, *Journal of Colloid and Interface Science* 196 (2), 1997, 267 - 277. https://doi.org/10.1006/jcis.1997.5209

[13] L. Song, M. Elimelech, Theory of Concentration Polarization in Cross-flow Filtration, *Journal of the Chemical Society, Faraday Transactions*, 91(19), 1995, 3389 - 3398. DOI:10.1039/FT9959103389

[14] L. Song, Flux decline in cross-flow microfiltration and ultrafiltration: mechanisms and modeling of membrane fouling, *Journal of Membrane Science* 139 (2), 1998, 183 - 200. https://doi.org/10.1016/S0376-7388(97)00263-9

[15] S. Bhattacharjee, A. S. Kim, M. Elimelech, Concentration polarization of interacting solute particles in cross-flow membrane filtration, *Journal of Colloid and Interface Science* 212 (1), 1999, 81 - 99. https://doi.org/10.1006/jcis.1998.6045

[16] S. Kim, M. Marion, B.-H. Jeong, E. M. Hoek, Cross-flow membrane filtration of interacting nanoparticle suspensions, *Journal of Membrane Science* 284 (1-2), 2006, 361 - 372. https://doi.org/10.1016/j.memsci.2006.08.008

[17] J. Hermia, Constant pressure blocking filtration lawsApplication to power-law non-Newtonian fluids, *Transaction of The Institution of Chemical Engineers* 60, 1982, 183187.

[18] V. Brião, C. Tavares, Pore Blocking Mechanism for the Recovery of Milk Solids from Dairy Wastewater by Ultrafiltration, *Brazilian Journal of Chemical Engineering* 29 (2), 2012, 393 - 407. http://dx.doi.org/10.1590/S0104-66322012000200019

[19] K. Konieczny , Modelling of membrane filtration of natural water for potable purposes , *Desalination* 143, 2002, 123 - 139. https://doi.org/10.1016/S0011-9164(02)00234-5

[20] A. Broeckmann, J. Busch, T. Wintgens, W. Marquardt, Modeling of pore blocking and cake layer formation in membrane filtration for wastewater treatment, *Desalination* 189, 2006, 97 - 109. https://doi.org/10.1016/j.desal.2005.06.018

[21] A. Kim and Y. Liu, Critical flux of hard sphere suspensions in cross-flow filtration: Hydrodynamic force bias Monte Carlo simulations, *Journal of Membrane Science* 323 (1), 2008, 67 - 76. https://doi.org/10.1016/j.memsci.2008.06.029

[22] P.Doleček, Mathematical modelling of permeate flow in multichannel ceramic membrane, *Journal of Membrane Science* 100 (2), 1995, 111 - 119. https://doi.org/10.1016/0376-7388(94)00258-Z

[23] A. Pak, T. Mohammadi, S.M. Hosseinalipour, V. Allahdini, CFD modeling of porous membranes, *Desalination* 222 (1-3), 2008, 482 - 488. https://doi.org/10.1016/j.desal.2007.01.152

[24] L. Huang, M. T. Morrissey, Finite element analysis as a tool for cross-flow membrane filter simulation, *Journal of Membrane Science* 155 (1), 1999, 19 - 30. https://doi.org/10.1016/S0376-7388(98)00300-7

[25] V. Geraldes, V. Semião, M. N. Pinho, Numerical modelling of mass transfer in slits with semi-permeable membrane walls, *Engineering Computations* 17 (3), 2000, 192 - 218. https://doi.org/10.1108/02644400010324857

[26] J. Kromkamp, A. Bastiaanse, J. Swarts, G. Brans, R. van der Sman, R. Boom, A suspension flow model for hydrodynamics and concentration polarisation in cross-flow microfiltration, *Journal of Membrane Science* 253 (12), 2005, 67 - 79. https://doi.org/10.1016/j.memsci.2004.12.028

[27] S. Kim, Coupled Lattice Boltzmann and Finite Volume Method for Modelling Cross-flow Filtration, PhD thesis, *Swansea University*, 2015.

[28] M. Paipuri, S. H. Kim, O. Hassan, N. Hilal, K. Morgan, Numerical modelling of concentration polarisation and cake formation in membrane filtration processes, *Desalination*, 365, 151 - 159. https://doi.org/10.1016/j.desal.2015.02.022

[29] I. Halliday, L. A. Hammond, C. M. Care, K. Good, A. Stevens, Lattice Boltzmann equation hydrodynamics, *Phys. Rev. E* 64:011208, 2001. DOI: 10.1103/PhysRevE.64.011208

[30] X. Niu, C. Shu, Y. Chew, An Axisymmetric Lattice Boltzmann Model for Simulation of Taylor-Couette Flows Between Two Concentric Cylinders, *International Journal of Modern Physics C* 14 (6), 2003, 785 - 796. https://doi.org/10.1142/S0129183103004929

[31] T. Lee, H. Huang, C. Shu, An axisymmetric incompressible lattice BGK model for simulation of the pulsatile flow in a circular pipe, *International Journal for Numerical Methods in Fluids* 49(1), 2005, 99 - 116. https://doi.org/10.1002/fld.997

[32] W. Xie, An axisymmetric multiple-relaxation-time lattice Boltzmann scheme, *Journal of Computational Physics* 281, 2015, 55 - 66. https://doi.org/10.1016/j.jcp.2014.10.019

[33] S. Chen, J. Tölke, S. Geller, M. Krafczyk, Lattice Boltzmann model for incompressible axisymmetric flows, PHYSICAL REVIEW E 78:046703, 2008. DOI: 10.1103/PhysRevE.78.046703

[34] Z. Guo, H. Han, B. Shi, C. Zheng, Theory of the lattice Boltzmann equation: Lattice Boltzmann model for axisymmetric flows, *Phys. Rev. E* 79:046708, 2009. DOI: 10.1103/PhysRevE.79.046708

[35] L. Wang, Z. Guo, C. Zheng, Multi-relaxation-time lattice Boltzmann model for axisymmetric flows, *Computers & Fluids* 39(9), 2010, 1542 1548. https://doi.org/10.1016/j.compfluid.2010.05.007

[36] L. Zheng, Z. Guo, B. Shi, C. Zheng, Lattice Boltzmann equation for axisymmetric thermal flows, *Computers & Fluids* 39(6), 2010, 945 - 952. https://doi.org/10.1016/j.compfluid.2010.01.006

[37] G. Bella, N. Rossi, S. Filippone, S. Ubertini, Using OpenMP on a Hydrodynamic Lattice-Boltzmann Code, 2002, accessed 7/8/2018. https://www.researchgate.net/publication/228823416_Using_openMP_on_a_hydrodynamic_lattice-Boltzmann_code

[38] E. Davidson, Message-passing for Lattice Boltzmann, Master's dissertation, *University of Edinburgh*, 2008.

[39] V. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, P. Dubey, Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU, *ISCA '10 Proceedings of the 37th annual international symposium on Computer architecture*, 451 - 460. https://doi.org/10.1145/1815961.1816021

[40] P.R. Rinaldi, E.A. Dari, M.J. Vénere, A. Clausse, A Lattice-Boltzmann solver for 3D fluid simulation on GPU, *Simulation Modelling Practice and Theory* 25, 2012, 163 - 171. https://doi.org/10.1016/j.simpat.2012.03.004

[41] J. Myre, S. D. C. Walsh, D. Lilja, M. O. Saar, Performance analysis of single-phase, multiphase, and multicomponent lattice-Boltzmann fluid flow simulations on GPU clusters, *Concurrency and Computation Practice and Experience* 23 (4), 2011, 332 - 350 https://doi.org/10.1002/cpe.1645

[42] G.Wellein, T.Zeiser, G.Hager, S.Donath, On the single processor performance of simple lattice Boltzmann kernels, *Computers & Fluids* 35 (89), 2006, 910 - 919. https://doi.org/10.1016/j.compfluid.2005.02.008

[43] C. A. Romero, R. H. Davis, Global model of cross-flow microfiltration based on hydrodynamic particle diffusion, *Journal of Membrane Science* 39 (2), 1988, 157 - 185. https://doi.org/10.1016/S0376-7388(00)80987-4

[44] J. G. Berryman, Random close packing of hard spheres and disks, *Physical Review A*, 27:1053, 1983. https://doi.org/10.1103/PhysRevA.27.1053

[45] W. S. Jodrey, E. M. Tory, Computer simulation of close random packing of equal spheres, *Phys. Rev. A* 32:2347, 1985. https://doi.org/10.1103/PhysRevA.32.2347

[46] J. Cho, I. S. Kim, J. Moon, B. Kwon, Determining Brownian and shear-induced diffusivity of nano- and micro-particles for sustainable membrane filtration, *Desalination* 188 (13), 2006, 213 - 216. https://doi.org/10.1016/j.desal.2005.04.119

[47] D.T. Leighton, A. Acrivos, Viscous resuspension, *Chem. Eng. Sci.* 41(6), 1986, 1377 - 1384. https://doi.org/10.1016/0009-2509(86)85225-3

[48] Q. Zou and X. He, On pressure and velocity boundary conditions for the lattice Boltzmann BGK model, Physics of Fluids 9(6), 1997, 1591 - 1598. https://doi.org/10.1063/1.869307

[49] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type, *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1), 1947, 50 - 67. https://doi.org/10.1017/S0305004100023197

[50] H. Versteeg, W. Malalasekera, An Introduction to Computational Fluid Dynamics: The Finite Volume Method, *Longman Scientific & Technical*, 1995. ISBN 0-582-21884-5

[51] V. Starov, D Lloyd, A. Filippov, S. Glaser, Sieve mechanism of microfiltration separation, *Separation and Purification Technology* 26(1), 2002, 51 - 59. https://doi.org/10.1016/S1383-5866(01)00116-2

[52] P. Bailey, J. Myre, S.D.C. Walsh, D.J. Lilja, M.O. Saar, Accelerating lattice Boltzmann fluid flow simulations using graphics processors, *International Conference on Parallel Processing*, 2009. DOI: 10.1109/ICPP.2009.38

[53] M. Wittmann, T. Zeiser, G. Hager, G. Wellein, Comparison of different Propagation Steps for the Lattice Boltzmann Method, *Computers & Mathematics with Applications* 65(6), 2013, 924 - 935.https://doi.org/10.1016/j.camwa.2012.05.002

[54] M. Wittmann, V. Haag, T. Zeiser, H. Kstler, G. Wellein, Lattice Boltzmann Benchmark Kernels as a Testbed for Performance Analysis, *Computers & Fluids* 172, 2018, 582 - 592. https://doi.org/10.1016/j.compfluid.2018.03.030

[55] E. Calore, A. Gabbana, S. Schifano, R. Tripiccione, Optimization of lattice Boltzmann simulations on heterogeneous computers, *The International Journal of High Performance Computing Applications*, 2017. https://doi.org/10.1177/1094342017703771

[56] S. C. Mishra, H. K.Roy, Solving transient conduction and radiation heat transfer problems using the lattice Boltzmann method and the finite volume method, *Journal of Computational Physics*, 223(1), 89 - 107. https://doi.org/10.1016/j.jcp.2006.08.021

[57] Y. Sun, X. Zhang, Analysis of transient conduction and radiation problems using lattice Boltzmann and finite volume methods, *International Journal of Heat and Mass Transfer*, 97, 611 - 617. https://doi.org/10.1016/j.ijheatmasstransfer.2016.01.074

[58] L. Li, R. Mei, J. Klausner, Multiple-relaxation-time lattice Boltzmann model for the axisymmetric convection diffusion equation, *International Journal of Heat and Mass Transfer* 67, 2013, 338 - 351. https://doi.org/10.1016/j.ijheatmasstransfer.2013.08.039

[59] O. Ogunbiyi, Effects of Turbulence Promoter Geometry on Flux Sustainability and Ceramic Membrane Efficiency, PhD thesis, *University of Nottingham*, 2007.

[60] L. D. Maxim, R. Niebo, E. E. McConnell, Bentonite toxicology and epidemiology a review, *Inhalation Toxicology*, 28;13, 591 - 617. https://doi.org/10.1080/08958378.2016.1240727

[61] M.Benna, N. Kbir-Ariguib, C. Clinard, F. Bergaya, Static filtration of purified sodium bentonite clay suspensions. Effect of clay content, *Applied Clay Science*, 19(16)1, 103 - 120. https://doi.org/10.1016/S0169-1317(01)00050-3