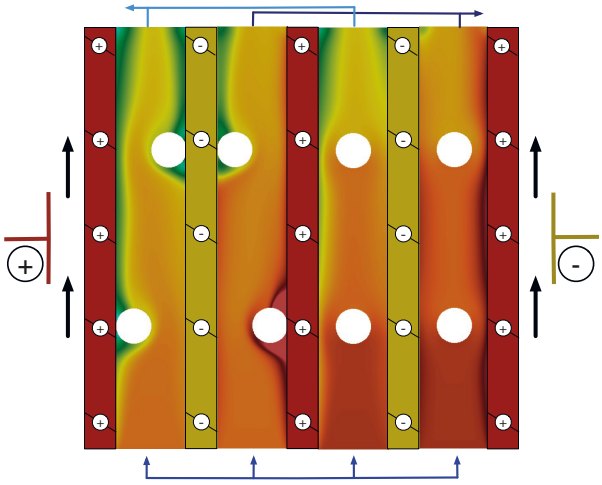
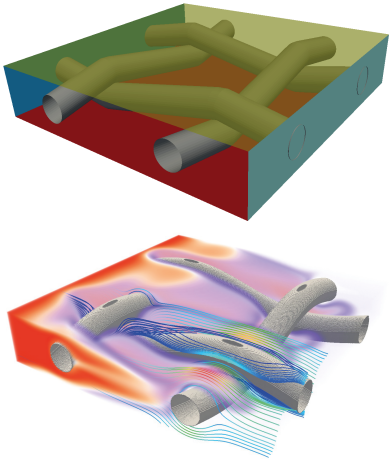
Simulation Techniques in Siegen
Vol 4

*Kannan Masilamani*

# Framework for Coupled Simulation of Electrodialysis Processes



universi
UNIVERSITÄTSVERLAG SIEGEN

# Framework for Coupled Simulation of Electrodialysis Processes

# Konzept und Umgebung für gekoppelte Simulationen von Elektrodialyse-Prozessen

Von der Fakultät für Maschinenwesen der
Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
genehmigte Dissertation

vorgelegt von

Kannan Masilamani

Berichter: Univ.-Prof. Dr.-Ing. Sabine Roller
Univ.-Prof. Dr.-Ing. Wolfgang Marquardt

Tag der mündlichen Prüfung: 09.06.2020

**Bibliographic information published by the Deutsche Nationalbibliothek**
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at **http://dnb.d-nb.de**.

D 82 (Diss. RWTH Aachen University, 2020)

**Simulation Techniques in Siegen Vol. 4 / STS Vol. 4 (2021)**
Editors: Sabine Roller and Harald Klimach



1st edition univers*i* – Universitätsverlag Siegen 2021

Printing: UniPrint, Universität Siegen
printed on wood- and acid-free paper

univers*i* – Universitätsverlag Siegen
Am Eichenhang 50
57076 Siegen
Germany
info@universi.uni-siegen.de
www.uni-siegen.de/universi

ISBN: 978-3-96182-077-1

## Acknowledgements

This dissertation is the outcome of several years of my work, actually much more than expected. Initiated in SIEMENS AG, Erlangen, Germany as part of the HISEEM project funded by the BMBF and then continued in the research group Simulation Techniques and Scientific Computing (STS) at the University of Siegen, Germany. During this long journey, there are lot of people who helped, supported, encouraged and supervised me to achieved this work in one way or another and I would like to take this opportunity to express my gratitude to all of them.

First and foremost, I would like to give my heartfelt gratitude to Prof. Dr.-Ing. Sabine Roller for her support and supervision during my time in SIEMENS AG and then offering me the opportunity to continue this work in her research group STS even after the end of the HISEEM project. This work became possible mainly because of her constant trust, patience, motivation and understanding over these years. Sabine's meticulous comments were an enormous help to improve the quality of this thesis. She has been an amazing mentor for my doctorate study beyond my imagination. I am deeply grateful to Dr.-Ing. Harald Klimach for his support and guidance in developing efficient algorithms and their implementation. The codes developed in this work were possible only from several fruitful discussion with him over these years.

I am deeply grateful to Prof. Dr. Pietro Asinari from Politecnico Torino for his support in developing the multicomponent lattice Boltzmann method for liquid mixtures, which is an essential part of this work.

My special thanks goes to Dr. Andreas Hauser, who offered me the opportunity to represent SIEMENS AG in the HISEEM project. His supervision from Singapore regarding the flow simulations with spacer geometries helped very much in the initial stage of this thesis. These simulations are the main motivation for the redesign of the octree mesh generator *Seeder* because of their large computational mesh requirements. Also, my sincere thanks goes to Dr. Elvira Maria Fernándes Sanchis, Dr. Sonja Wolfrum and Dr. Manfred Baldauf for their support and supervision during my time in SIEMENS AG.

I would like to thank all my colleagues in STS for their support, patience and understanding which helped me complete this dissertation. They created a friendly work atmosphere and several fun events like game evenings, outdoor events which helped to refresh the mind and increase the focus at work. I am particularly grateful to Neda Ebrahimi Pour for

her assistance in running hydrodynamic flow simulations with different spacer geometries. The comments given by my former colleague Verena Krupp has been a great help in improving the style of this thesis. I would like to offer my thanks to my former colleagues Dr.rer.nat. Jens Zudrop and Dr. Matthias Johannink who collaborated in the HISEEM project. Special thanks to Matthias for sharing his knowledge about the electrodialysis process, especially the laboratory experiments performed by him to measure the pressure drop across the channel with the woven spacer. I also would like to extend my thanks to my other former colleagues Dr.-Ing. Manuel Hasert and Dr.-Ing. Jiaxing Qi for their contribution in the development of the lattice Boltzmann solver.

I would like to express my gratitude to BMBF for funding HISEEM project and all the project partners. I also would like to offer my special thanks to HLRS, Germany for the computing budget in their supercomputers. Without them the large scale flow simulations with spacer geometries wouldn't have been possible.

Finally, I would like to offer my heartfelt thanks to my parents, Masilamani and Shanthi for their patience and support. They gave me the freedom to pursue my dream to come to Germany for master study and then do the doctoral study. Without them I wouldn't be the person who I am today. A very special thanks to my wife Rashmi for her care and support during my doctoral study. I cannot thank her enough for taking care of our daughter Ishanvi all by herself and letting me work at home. Her motivation and encouragement gave me the strength to finish writing this thesis.

*Kannan Masilamani*

## Zusammenfassung

Elektrodialyse ist ein effizientes Verfahren zur Meerwasserentsalzung, in dem verschiedene Phänomene zusammenwirken. Ionen werden dabei durch Strömung, Diffusion und eine elektrische Kraft transportiert und mittels selektiven Membranen getrennt. Für die Optimierung dieses Prozesses ist ein Verständnis der Interaktion dieser Effekte wichtig. Diese Arbeit präsentiert rigorose mathematische Modelle des Gesamtprozesses und erarbeitet eine numerische Strategie zur Simulation Durch diesen Ansatz wird es möglich die auftretenden Effekte mit ihren komplexen Interaktionen detailliert zu simulieren. Dazu werden die Maxwell-Stefan Gleichungen für Gemische verwendet, die auch die elektrische Kraft und Mehrkomponentenwechselwirkungen mit konzentrationsabhängigen Diffusions- und thermodynamischen Faktoren berücksichtigt. Darüber hinaus wird hier ebenfalls nicht die übliche Annahme der lokalen Elektroneutralität getroffen, um die Effekte in der elektrochemischen Doppelschicht an den Membranen zu ermöglichen. Für die numerische Behandlung dieser Gleichungen wird ein Lattice-Boltzmann (LBM) Verfahren im Löser *Musubi* implementiert. Das Modell der Kanaldurchströmung wird mit einem elektrischen Feld und einem Modell für die Membranen gekoppelt. Für das elektrische Feld wird in Musubi eine LBM zur Lösung der Poisson Gleichung implementiert.

Die Kanäle zwischen den Membranen werden durch Abstandshalter mit komplexer Geometrie realisiert. Um eine passende Diskretisierung des Gitters für diese Kanäle zu gewährleisten wird ein Gittergenerator (*Seeder*) auf der Basis von Octrees entwickelt. Ein wesentlicher Teil dieser Arbeit ist der Entwicklung des parallel skalierbaren Kopplungswerkzeugs *APESmate* gewidmet. *APESmate* wird im Rahmen der APES-Suite neben *Seeder* und *Musubi* entwickelt und erlaubt die Interaktion verschiedener Löser auf Basis einer gemeinsamen zentralen Octree-Datenstruktur, die eine effiziente Handhabung der I/O auf großen verteilt parallelen Rechensystemen ermöglicht.

Die entwickelte Software wird verwendet um das nichtideale Mehrkomponentenmodell fir verschiedene Konzentrationen und Oberflächenpotentiale zu vergleichen. Die Ergebnisse belegen, dass nichtideale Effekte, insbesondere in der elektrochemischen Doppelschicht, mit der Konzentration zunehmen. Die Abstandshalter werden mit mehreren hydrodynamischen Winkeln und Einströmgeschwindigkeiten nahe und fern einer Ecke untersucht um die Auslegung mit minimalen Druckabfall und ohne Totwassergebiete zu bestimmen. Diese hochaufgelösten Simulationen zeigen, dass der

Druckabfall mit dem hydrodynamischen Winkel zunimmt, während die Größe der Zonen mit geringer Durchströmung abnehmen.

## Abstract

Electrodialysis is an efficient process for seawater desalination that involves various interacting phenomena. In this process, ions are transported by flow, diffusion and an electric force and separated by selective membranes. For the optimization of this process, it is important to understand these interactions. This work presents rigorous mathematical models to describe the overall process and develops a numerical strategy for its simulation. With this approach it becomes possible to simulate the involved physical effects and their interactions in detail. To achieve this, the Maxwell-Stefan equations for mixtures are used. They take into account the electrical force and the multicomponent interactions with concentration dependent diffusivity coefficients and thermodynamic factors. Additionally, the usual assumption of local electroneutrality is not assumed to allow the nonideal effects in the electrical double layer near the membrane. For the numerical solution of these equations, the multicomponent lattice Boltzmann method (LBM) is developed and implemented in the solver *Musubi*. This model for the channel flow is coupled with an electric field and a model for the membranes. To obtain the electric field, the LBM that solves the Poisson's equation is implemented in *Musubi*.

The channels between the membranes are realized by spacers with complex geometry. A mesh generator (*Seeder*) on the basis of octrees is developed to ensure the appropriate discretization of the mesh for these channels. An essential part of this work is dedicated to the development of the parallel scaling coupling tool *APESmate*. *APESmate* is developed within the APES suite along with *Seeder* and *Musubi* on a central octree data structure that allows efficient handing of I/O on large scale distributed parallel computing systems.

The developed software is used to compare the nonideal multicomponent model for various concentrations and surface potentials. The results show that nonideal effects increase with the concentration, especially in the electrical double layer. The spacers for various hydrodynamic angles and inflow velocities near and away from a sealed corner are investigated to find the design with reduced pressure drop and without low velocity zones. The highly resolved simulations show that the pressure drop increases with the hydrodynamic angle, while the extend of the low flow regions decreases.

# Contents

# Nomenclature

### Symbols

| | |
|---|---|
| $\boldsymbol{g}$ | Gravitational acceleration [$\mathrm{m\,s^{-2}}$] |
| $\boldsymbol{x}_b$ | Position of virtual boundary node |
| $K$ | Bulk modulus of the liquid mixture |
| $z_k$ | Charge number of species $k$ [-] |
| $L_c$ | Characteristics length of the flow [m] |
| $c_\infty$ | Bulk concentration of solvent [$\mathrm{mol\,m^{-3}}$] |
| $\kappa^{mem}$ | Specific conductance of membrane[$\mathrm{S\,m^{-1}}$ or $\mathrm{C^2\,s\,kg^{-1}\,m^{-3}}$] |
| $\boldsymbol{d}_k$ | Mass diffusive driving force of species $k$ [$\mathrm{m^{-1}}$] |
| $d_f$ | Diameter of the spacer filament [m] |
| $\boldsymbol{i}^d$ | Current density [$\mathrm{C\,m^{-2}\,s^{-1}}$] |
| $\bar{i}^{d,mem}$ | Current density transported through membranes along normal direction [$\mathrm{C\,m^{-2}\,s^{-1}}$] |
| $\boldsymbol{D}$ | Electric displacement field [$\frac{As}{m^2}$] |
| $\boldsymbol{E}$ | Electric field [$\mathrm{V\,m^{-1}}$] |
| $\hat{\boldsymbol{E}}$ | Constant electric field [$\mathrm{V\,m^{-1}}$] |
| $\boldsymbol{F}_k$ | External diffusive driving force on species $k$ [$\mathrm{m^{-1}}$] |
| $\hat{\boldsymbol{F}}_k$ | External body force per mole on species $k$ [$\mathrm{m\,s^{-2}}$] |
| $\boldsymbol{F}$ | External force [$\mathrm{m\,s^{-2}}$] |
| $U$ | Flow velocity [$\mathrm{m\,s^{-1}}$] |
| $\boldsymbol{x}_f$ | Position of fluid node next to the boundary |
| $R$ | Universal gas constant 8.3144621 [$\mathrm{J\,mol^{-1}\,K^{-1}}$] [$\mathrm{kg\,m^2\,s^{-2}\,mol^{-1}\,K^{-1}}$] |
| $H$ | Height of the channel [m] |
| $h_{ch}$ | Height of the spacer channel [m] |
| $\boldsymbol{I}$ | Identity matrix |
| $\boldsymbol{M}^{-1}$ | Inverse of transformation matrix |
| $n_i$ | Number of ionic species |
| $\tilde{P}$ | Kinematic pressure [$\mathrm{m^2\,s^{-2}}$] |
| $L^2$ | L2-norm |
| $c_s$ | Lattice speed of sound |
| $\boldsymbol{u}^m$ | Lattice velocity vector along the direction $m$ |
| $c$ | Lattice speed ($\delta$x/$\delta$t) |
| $L$ | Length of the channel [m] |
| $W$ | Width of the channel [m] |
| $l_{ch}$ | Length of the spacer channel [m] |
| $\boldsymbol{A}$ | Linear collision operator |
| $l_m$ | Orthogonal distance between two consecutive parallel filaments [m] |
| $\boldsymbol{H}$ | Magnetization [$\frac{A}{m}$] |
| $\boldsymbol{B}$ | Magnetic field [$\frac{Vs}{m^2}$] |
| $\boldsymbol{j}_k$ | Mass diffusive flux density of species $k$ [$\mathrm{kg\,m^{-2}\,s^{-1}}$] |
| $\boldsymbol{N}_k$ | Mole flux density of species $k$ [$\mathrm{mol\,m^{-2}\,s^{-1}}$] |
| $\boldsymbol{n}_k$ | Mass flux density of species $k$ [$\mathrm{kg\,m^{-2}\,s^{-1}}$] |
| $y_k$ | Mass fraction of species $k$ [-] |
| $\mathbf{v}$ | Mass averaged mixture velocity [$\mathrm{m\,s^{-1}}$] |
| $v_m$ | Maximum velocity at middle of the channel [$\mathrm{m\,s^{-1}}$] |

| | |
|---|---|
| $M$ | Molecular weight of the mixture [$\mathrm{kg\,mol^{-1}}$] |
| $M_k$ | Molar mass or molecular weight of species $k$ [$\mathrm{kg\,mol^{-1}}$] |
| $c_k$ | Mole density or concentration of species $k$ [$\mathrm{mol\,m^{-3}}$] |
| $\boldsymbol{J}_k$ | Molar diffusive flux density of species $k$ [$\mathrm{mol\,m^{-2}\,s^{-1}}$] |
| $\mathbf{w}$ | Molar averaged mixture velocity [$\mathrm{m\,s^{-1}}$] |
| $\mathbf{p}$ | Momentum vector [$\mathrm{kg\,m^{-2}\,s^{-1}}$] |
| $\boldsymbol{m}$ | Moments space vector |
| $nF/inch$ | Number of spacer filaments per inch |
| $nH$ | Number of elements in height |
| $\boldsymbol{n}$ | Normal direction of the boundary wall |
| $n$ | Number of components in the mixture |
| $D$ | Spatial dimension |
| $Q$ | Number of discrete velocity directions |
| $p_k$ | Partial pressure of species $k$ |
| $f$ | Particle distribution function |
| $f^{eq}$ | Equilibrium distribution function |
| $F^m$ | External body force along the direction $m$ |
| $f^{neq}$ | Non-equilibrium distribution function |
| $f^c$ | Post-collision distribution function |
| $f^{pre}$ | Pre-collision distribution function |
| $\boldsymbol{f}$ | Particle distribution function vector |
| $\boldsymbol{x}$ | Physical spatial position vector [m] |
| $P$ | Pressure [$\mathrm{N\,m^{-2}}$] |
| $q^m$ | Normalized distance between exact wall and fluid node along the direction $m$ |
| $r$ | Radial distance from the cylinder center [m] |
| $R^{mem}$ | Specific resistivance of membrane [ m or $\mathrm{kg\,m^3\,C^{-2}\,s^{-1}}$] |
| $f_k^{eq,m}$ | Equilibrium distribution function of species $k$ along the direction $m$ |
| $\mathbf{v}_k^{eq}$ | Equilibrium velocity of species $k$ |
| $d_k^m$ | External driving force on species $k$ along the direction $m$ |
| $\mathbf{v}_k$ | Mass velocity of species $k$ [$\mathrm{m\,s^{-1}}$] |
| $S_{vsp}$ | Specific surface of the spacer |
| $\boldsymbol{t}$ | Tangent direction of the boundary wall |
| $T$ | Temperature [K] |
| $t$ | Physical time [s] |
| $\boldsymbol{M}$ | Transformation matrix |
| $\bar{\boldsymbol{j}}_k$ | Transformed momentum of species $k$ |
| $T_k$ | Transport number of species $k$ |
| $T_{\mathrm{w}}$ | Transport number of water |
| $\bar{f}^m$ | Local transformation of particle distribution function along the direction $m$ |
| $\bar{v}_{in}$ | Mean inflow velocity [$\mathrm{m\,s^{-1}}$] |
| $\tilde{V}_k$ | Molar volume of species $k$ [$\mathrm{m^3\,mol^{-1}}$] |
| $s_m^k$ | Density weight factor of species $k$ along the direction $m$ |
| $w_{ch}$ | Width of the spacer channel [m] |
| $\boldsymbol{x}_w$ | Exact position of boundary wall |

**Greek Symbols**

| | |
|---|---|
| $\gamma_k$ | Mole fractions based activity coefficients of species $k$ |
| $\alpha$ | Hydrodynamic angle or angle between spacer filaments in different layer |
| $\beta$ | Angle between spacer filament and flow direction |
| $\zeta$ | Bulk viscosity [$\mathrm{m^2\,s^{-1}}$] |
| $\rho^e$ | Charge density [$\mathrm{C\,m^{-3}}$] |
| $\overline{\mu}_k$ | Molar chemical potential of species $k$ [J mol] |
| $\Omega$ | Collision operator |
| $\Omega_{k,l}$ | Cross-Collision operator between species $k$ and $l$ |
| $\lambda_D$ | Debye length [m] |
| $\rho$ | Total mixture/fluid density [$\mathrm{kg\,m^{-3}}$] |
| $\rho^0$ | Constant fluid density [$\mathrm{kg\,m^{-3}}$] |
| $\boldsymbol{\sigma'}$ | Deviatoric stress [$\mathrm{N\,m^{-2}}$] |
| $\mu$ | Dynamic viscosity of mixture [$\mathrm{kg\,m^{-1}\,s^{-1}}$] |
| $\varepsilon$ | Electric permittivity [$\frac{As}{Vm}$] |
| $\varepsilon_0$ | Electric permittivity of vaccum $8.85 \times 10^{-12}$[$\frac{As}{Vm}$] |
| $\varepsilon_r$ | Relative electric permittivity |
| $\psi$ | Scalar electric potential [V] |
| $\epsilon$ | Smallness parameter |
| $\nabla_{T,P}$ | Derivative at constant temperature and pressure |
| $\lambda_{hom}$ | Relaxation frequency for higher order moments |
| $\nu$ | Kinematic viscosity of mixture [$\mathrm{m^2\,s^{-1}}$] |
| $\delta_{\alpha,\beta}$ | Kronecker delta, $\delta_{\alpha.\alpha} = 1$ |
| $\bar{\mu}$ | Magnetic permeability [$\frac{Vs}{Am}$] |
| $\chi_k$ | Mole fraction of species $k$ [-] |
| $\boldsymbol{\Pi}$ | Momentum flux tensor [$\mathrm{N\,m^{-2}}$] |
| $\boldsymbol{\Pi}^0$ | Equilibrium momentum flux tensor [$\mathrm{N\,m^{-2}}$] |
| $\boldsymbol{\Pi}^{(1)}$ | non-Equilibrium momentum flux tensor [$\mathrm{N\,m^{-2}}$] |
| $\partial$ | Partial derivative |
| $\Psi$ | Permselectivity of ion exchange membrane |
| $\phi_k$ | Parameter related to species molecular weight |
| $\gamma$ | Potential diffusivity [$\mathrm{m^2\,s^{-1}}$] |
| $\lambda^\gamma$ | Relaxation parameter for potential LBE |
| $\boldsymbol{\Lambda}$ | Relaxation matrix |
| $\lambda$ | Relaxation frequency |
| $\lambda^\nu$ | Relaxation frequency relate to kinematic viscosity |
| $\lambda^\zeta$ | Relaxation frequency relate to bulk viscosity |
| $\tau$ | Relaxation time |
| $\Omega_{k,k}$ | Self-Collision operator of species $k$ |
| $\rho_k$ | Density of species $k$ [$\mathrm{kg\,m^{-3}}$] |
| $\lambda_k^\zeta$ | Relaxation frequency of species $k$ related to bulk viscosity $\zeta$ |
| $\lambda_{k,l}^{\mathcal{D}}$ | Relaxation frequency of species $k$ related to Maxwell-Stefan diffusivity coefficient $\mathcal{D}_{k,l}$ |
| $\lambda_k^\nu$ | Relaxation frequency of species $k$ related to kinematic viscosity $\nu$ |
| $\boldsymbol{\Lambda}_k$ | Relaxation matrix for species $k$ |
| $\delta x$ | Physical element size [m] |
| $\delta x^*$ | Lattice element size |
| $\boldsymbol{\Gamma}$ | Thermodynamic factor |
| $\delta t$ | Physical time step [s] |
| $\delta t^*$ | Lattice time step |

| | |
|---|---|
| $\epsilon$ | Voidage |
| $\omega^m$ | Lattice weight along the direction $m$ |

**Sub-/Superscripts**

| | |
|---|---|
| $conv$ | Convection of species due to fluid velocity |
| $diff$ | diffusion of species due to its concentration gradient |
| $m$ | Lattice direction |
| $\bar{m}$ | Inverse lattice direction |
| $c$ | Post-collision |
| $eq$ | Equilibrium |
| $neq$ | non-Equilibrium |
| $pre$ | Pre-collision |
| $l$,$r$ | left and right |
| $mem$ | membrane |
| $mig$ | migration of species due to an external force |
| $e$ | Electrolyte solution |

**Subscripts**

| | |
|---|---|
| $sp$ | Atmospheric |
| $c$ and $a$ | Cation and anion |
| $ch$ | Channel |
| $Dom$ | Donnan |
| $fix$ | Charges fixed to the membrane polymer |
| $in$ | Inlet |
| $k$, $l$, $j$ | Species counters |
| $o$ | Outlet |
| $x$, $y$, $z$ | Spatial dimension |
| $sp$ | Spacer |
| $t$ | Total |
| w | Water |

**Other Symbols**

| | |
|---|---|
| $\mathcal{D}_{k,l}$ | Maxwell-Stefan diffusion coefficients of species $k$ and $l$ $[\mathrm{m^2\,s^{-1}}]$ |
| $\mathcal{F}$ | Faraday constant 96485.3329 $[\mathrm{C\,mol^{-1}}]$ |
| $\mathcal{D}$ | Free parameter |
| $\mathfrak{L}$ | Refinement level |
| $\mathcal{B}_{k,l}$ | Maxwell-Stefan resistivity coefficients of species $k$ and $l[\mathrm{s\,m^{-2}}]$ |
| $\mathcal{S}^m$ | Source term of potential LBE along the direction $m$ |
| $\mathfrak{tID}$ | TreeID of a node |

**Acronyms**

| | |
|---|---|
| AEM | Anion exchange membrane |
| AoS | Array of Structures |
| $APES$ | Adaptable Poly-Engineering Simulator |
| $BC$ | Boundary condition |
| BCID | Unique identification number for Boundary label |
| BGK | Bhatnagar-Gross-Krook |
| CEM | Cation exchange membrane |
| DG | Discontinuous Galerkin |
| ED | Electrodialysis |
| $EDL$ | Electrical Double Layer |
| EDL | Electrical double layer |
| EDR | Electrodialysis-reversal |
| EWI | Singapore Environment and Water Industry Development Council |
| FVDM | Finite Discrete Velocity Model |
| $IC$ | Initial condition |
| IEM | Ion exchange membrane |
| $IS$ | Ionic strength of the electrolyte solution |
| $Kn$ | Knudsen number |
| LBE | Lattice Boltzmann Equation |
| LBM | Lattice Boltzmann Method |

| | |
|---|---|
| LBM | Lattice Boltzmann Method |
| LGA | Lattice Gas Automata |
| $Ma$ | Mach number |
| MC | Multicompontent |
| MPI | Message Passing Interface |
| MRT | Multiple Relaxation Time |
| MS | Maxwell-Stefan's equation |
| $nBnds$ | Number of fluid elements with boundary neighbor |
| $nFluids$ | Number of fluid elements |
| NP | Nernst-Planck's equation |
| $LW$ | Osmotic water transport coefficient |
| PDF | Particle Distribution Function |
| Re | Reynolds number |
| RO | Reverse Osmosis |
| SC | Single Compontent |
| Sc | Schmidt number |
| SFC | Space Filling Curve |
| SoA | Structure of Arrays |
| SRT | Single Relaxation Time |
| $STfun$ | Space-time function |
| $varSys$ | Variable System |

# Notation

Let us begin with the introduction of some basic notations of variables and operators, which are used in this thesis to support the technical discussion. In $\mathbb{R}^d$, subscripts are used to specify different spatial directions. E.g. a point $\mathbf{x}$ in $\mathbb{R}^d$ is given by the $d$-tuple

$$\mathbf{x} = (x_1, \ldots, x_d)^T .$$

Here, a point $\mathbf{x}$ is written in bold notation to represent the $d$-tuple. Also, vectors, vector-valued functions, matrices and tensors are written in bold notation. E.g. a matrix $\mathbf{c}$ in $\mathbb{R}^d$ is given by

$$\mathbf{c} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} .$$

The dyadic product $\otimes$ of two vectors $\mathbf{a}$ and $\mathbf{b}$ results in a matrix and it is defined as

$$\mathbf{a} \otimes \mathbf{b} = \mathbf{a} \cdot \mathbf{b}^T = \begin{pmatrix} a_1 b_1 & \cdots & a_1 b_d \\ \vdots & & \vdots \\ a_d b_1 & \cdots & a_d b_d \end{pmatrix} .$$

The double inner product : of two tensors $\mathbf{c}$ and $\mathbf{d}$ results in a scalar and it is defined as

$$\mathbf{c} : \mathbf{d} = \sum_i \sum_j c_{ij} d_{ij}.$$

The partial derivatives are denoted by subscripts. E.g. a partial derivative of a function $f$ with respect to $x_i$ is written as

$$\partial_{x_i} f = \frac{\partial f}{\partial x_i}.$$

The nabla operator is used to collect all partial derivatives in $\mathbb{R}^d$ and it is written as

$$\nabla = (\partial_{x_1}, \ldots, \partial_{x_d})^T .$$

The gradient of a scalar function $f$ is denoted by

$$grad(f) = \nabla f = (\partial_{x_1} f, \ldots, \partial_{x_d} f)^T .$$

Finally, the divergence and curl of a vector-valued function are given by

$$div(f) = \nabla \cdot f = \sum_{i=1}^{d} \partial_{x_i} f_i$$

and

$$curl(f) = \nabla \times f = \begin{pmatrix} \partial_{x_2} f_3 - \partial_{x_3} f_2 \\ \partial_{x_3} f_1 - \partial_{x_1} f_3 \\ \partial_{x_1} f_2 - \partial_{x_2} f_1 \end{pmatrix}$$

respectively.

# 1 Introduction

In recent years, the scarcity of clean drinking water has become a growing problem in most parts of the world due to continuous global warming and increasing world population. Drink water scarcity is either due to limited availability of natural water resources referred to as physical water scarcity or due to lack of infrastructure referred to as economic water scarcity (Figure 1.1). In contrast, 2/3 of our world is covered by water. Figure 1.2 shows the breakdown of total world water and also the breakdown of available fresh water. Thus, seawater desalination is the best way to solve the water problem in the world. Among several desalination techniques, the electrodialysis process has proven to be a cost and energy efficient technology. Even though this process is used in practice, the underlying physical phenomenon like transport of ions from the channel to the membranes and the impact of the complex geometry between the membranes on the ions transport and pressure drop across the channel are not well understood. With the help of supercomputers, it is now possible to numerically simulate those phenomena to gain a deeper understanding of this process. In this thesis, the scalable coupled simulation algorithm and the framework developed to investigate this process are presented.

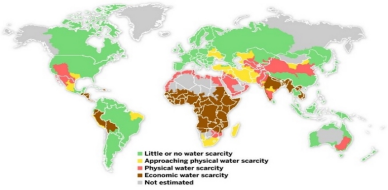



**Figure 1.1** World water scarcity

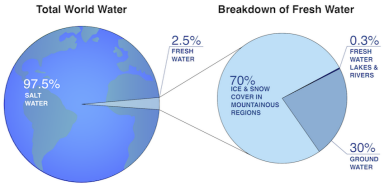



**Figure 1.2** Breakdown of total world water

## 1.1 State of the Art

There are several desalination techniques to remove salt from seawater such as thermal distillation and membrane based technologies. Thermal

distillation methods consume large amount of heat energy to evaporate concentrate water in multiple stages to obtain drink water and due to the high energy consumption, heat processes incur high costs. An alternative technique is the use of membranes and electric fields to separate the salt from seawater. The most commonly used membrane techniques are Reverse Osmosis (RO) and Electrodialysis (ED). Figure 1.3 shows the steps generally involved in a desalination process. The energy is required mostly at the desalination stage. The energy requirement for different industrial desalination techniques is listed in [22]. It is known that the RO process requires electrical energy of roughly $3 - 5.5\ \mathrm{kW\,h\,m^{-3}}$ to pump concentrated water through the osmotic membranes, whereas the ED process uses low pressure flows and selective ion exchange membranes to remove salt ions and requires less energy than RO. In 2007, Siemens won the challenge to produce drink water from seawater at an energy consumption $\leq 1.5\ \mathrm{kW\,h\,m^{-3}}$from the Singapore Environment and Water Industry Development Council (EWI). In 2008, Siemens Water Technology commenced the project to produce $50\ \mathrm{m^3\,d^{-1}}$ with the demonstration plant in Singapore [1]. They worked on the development of an ED prototype using experimental studies. In Germany, SIEMENS Corporate Technology in Erlangen joined a collaboration with German Research School (GRS) and Aachener Verfahrenstechnik (AVT) at RWTH Aachen University to use numerical simulation to understand the fundamental physical phenomena in the ED process.

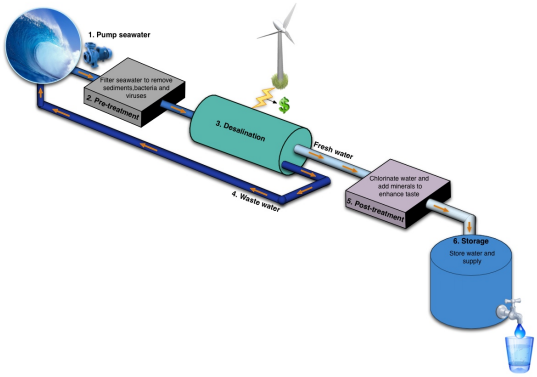



**Figure 1.3** Steps involved in desalination process

The ED process has several advantages over RO:

- high water recovery rate of up to 80%,
- less feed water pretreatment,
- less membrane fouling [1] or scaling[2] due to process reversal,
- easily tunable for feed water concentration,
- easy start-up and shut-down of the process for intermittent operation.

Figure 1.4 illustrates the principle of the electrodialysis process. It consists of selective ion exchange membranes (IEM) such as cation exchange membrane (CEM) and anion exchange membrane (AEM), which are arranged in alternative fashion between anode and cathode to form individual flow channels. Membranes that are swollen with negative and positive charged groups in their polymer structure are called as CEM and AEM respectively. Thus, cation exchange membranes allow only positively charged cations to pass through and retain negatively charged anions in the channel. Likewise, anion exchange membranes allow only negatively charged anions to pass through and retain positively charged cations. In other words, the same charge groups or co-ions are excluded and the opposite charge groups or counter-ions are passed through the membranes. When an electric potential is applied between the electrodes, anions migrate towards anode and cations migrate towards cathode. Due to alternating arrangement of AEM and CEM, ions are depleted and accumulated in alternate channels resulting in alternate dilute and concentrate channels respectively. At the inlet, seawater is pumped between the membranes into the flow channels and at the outlet, dilute channels are connected together to collect potable water and concentrate channels to collect brine or concentrated water. In an ED stack, the dilute and concentrate channels together with AEM and CEM constitute a repeating unit. In practical application, the ED stack contains several hundreds of this repeating unit. This principle of ED is exploited in industrial processes as the desalination of brackish[3] water [99], the concentration of organic or anorganic acids [8] and the demineralization of food products [90].

In ED processes, the electrical resistance of the solution in the channel affects the energy consumption. Therefore, it is necessary to keep the height of the channel as small as possible. In industrial ED stack, the distance between membranes is typically between 0.25 mm to 2 mm. To

[1] accumulation of ions on the surface of the membrane and the spacer
[2] mineral layer that forms on the membrane surface
[3] a mixture of salt water and fresh water

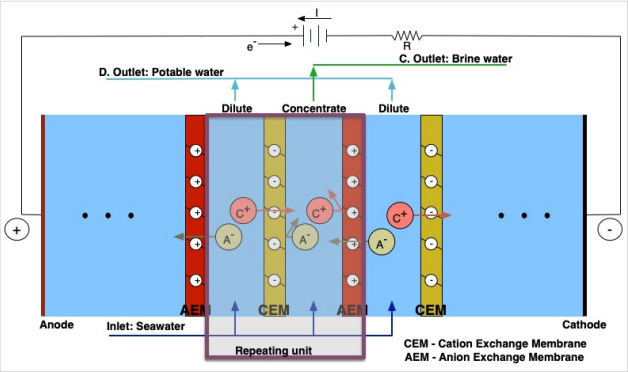



**Figure 1.4** Schematic layout shows the principle of the conventional electrodialysis process

create the flow channel between the membranes, a complex geometry structure called spacer is used as shown in Figure 1.5. The spacer acts as a mechanical stabilizer and turbulence promoter. It is also used to control the flow distribution in the channel. It has influence on the migration of ions in the flow channel and membranes and also the pressure drop along the channel. There are two types of spacer designs used in practice: woven and nonwoven spacers, which are shown in Figure 1.6. The following spacer design parameters like filament diameter, thickness, distance between filaments, angle between filaments and hydrodynamic angle (see Chapter 7) can be used in any combination to create the optimal spacer. The optimal spacer design should provide

- uniform flow distribution,
- improve the mixing of the solutions at the membrane surface,
- maximize effective membrane area,
- reduce low flow zones (areas with very low flow velocities) to avoid scaling and fouling formation in the flow channel,
- maximize the limiting current density[1] and
- minimize the pressure drop.

[1] When an electric current passes through an IEM, salt concentration on the desalting surface of the membrane decreases because of concentration polarization and reduces to zero at the limiting current density [93].

In this work, the impact of the spacer design on the pressure drop across the channel and on the ion distribution for various inflow mean velocities is investigated.

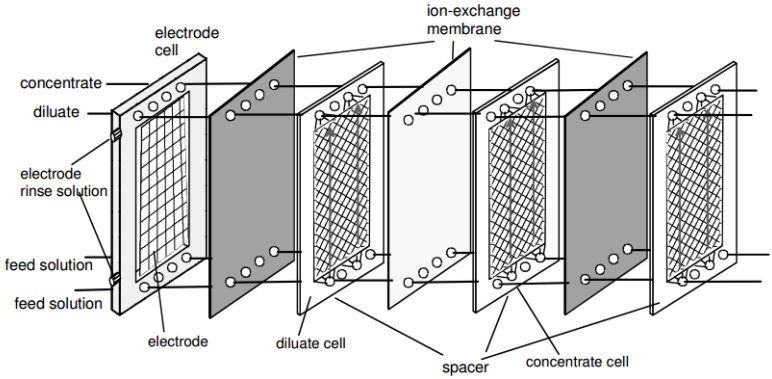



**Figure 1.5** Schematic drawing illustrates the construction of electrodialysis stack [91]

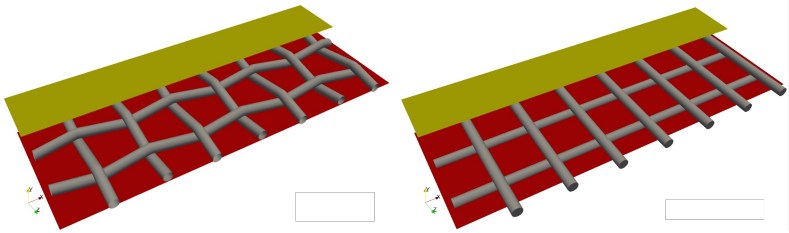


**Figure 1.6** Woven (left) and nonwoven (right) spacers

In the spacer-filled flow channels, the mass transport of ions occurs in the form of convection, diffusion and migration. Convection is due to the pressure drop across the channel along the main flow direction, migration towards membrane is due to the applied electric potential and diffusion happens near the membrane surface due to the concentration gradient. In the flow channels, the current is carried by both cation and anion whereas in IEMs, the current is carried mainly by counter-ions. The transport number [89] expresses the fraction of electric current carried by a certain ion. It is described in Section 2.2. For selective IEM, the transport number of an

ion defines the selective ion transport through the membrane. In addition to the transport number of an ion through membranes, the membrane permselectivity is an important integral membrane parameter that defines the degree at which counter-ions are passed through membranes and co-ions are excluded. Furthermore, the electric membrane resistance also strongly affects the process performance. The membrane resistance characterizes the drop of the electric potential across the membrane. The resistance and the selectivity are governed by the meso- and microscopic morphology of the membrane material [83]. The most desired properties of an IEM are high permselectivity, low electrical resistance, high chemical stability, as well as good mechanical and form stability [92]. The transport properties of the membrane can be numerically simulated by resolving the membrane and solving the transport equation per species. However, in this work, the membranes are not numerically resolved instead they are modeled as black-boxes that are based on an empirical description of the current-voltage characteristics of the membrane. In other words, the selective transport of an ion through the membrane is computed empirically using the transport number of an ion and the current density through the membrane. This approach was chosen because the main focus of this thesis is to develop a coupled simulation framework and it serve as a good starting point to mimic the membrane behavior in initial coupled simulations.

In the flow channel near the membrane, the concentration gradient is established due to the difference in transport properties of ions in the membrane and the solution in the flow channels. This concentration gradient results in diffusive transport in a small region near the surface of a membrane which is referred to as diffusive layer. However, far away from the membranes the bulk solution is well mixed. The existence of concentration gradients on both sides of the membrane leads to a concentration polarization. The concept of concentration polarization is introduced as an integral measure for the intensity of the mass transport in the spacer-filled flow channels. It is understood as the local enrichment or depletion of ions near the membrane surfaces in concentrate and dilute channels respectively. The after-effect of concentration polarization is different in dilute and concentrate channels. In concentrate channels, when the salt concentration reaches its solubility limit of the solution, precipitation of salt occurs resulting in scaling and fouling. In dilute channels, when the salt concentration near the membrane reaches zero, the flow of current through membranes stops, which increases the voltage drop and results in high energy consumption. The current density at this stage where no salt ions are available to transport the electrical current is referred to as limiting current density. In the ED process, the spacer geometries

are used to reduce the concentration polarization near the membranes by promoting mixing of ions and turbulence in the flow channels, which in turn increase the limiting current density.

In industrial applications of the ED processes, the formation of fouling, scaling and concentration polarization constitutes severe problems. These effects are strongly dependent on both the flow distribution adjacent to the membrane and the total flux of ions leaving or entering the flow channel through the membrane. These effects can be reduced by switching the anode and cathode i.e. electrodialysis reversal (EDR). In reversing the polarity, the dilute channel becomes concentrated and the concentrated channel becomes dilute. This increase in durability of the membranes in turn results in an increased durability of the ED stack, which makes this method more environmental friendly. Furthermore, the flow distribution near the membrane can be improved by modifying the spacer geometry and the mass transport of ions through membranes by tuning the membrane transport properties.

The mechanisms of the transport in the spacer-filled flow channels and in the nano-structured membranes are not well understood. A major difficulty in the experimental investigation of integral membrane properties is the discrimination of phenomena occurring in the membrane from those occurring in the adjacent electrolyte solution [23]. The coupled simulation of the transport phenomena in the membrane and the spacer-filled flow-channels can contribute to overcome this unsatisfactory situation.

## 1.2 Related works

The ED process has been studied experimentally and some modeling approaches have been developed [25, 81, 92, 93] to study the ion transport in the flow channel and the membranes. Costa et al [18] characterized the spacer geometry and its effect on fluid mixing by determining the flow-path in the spacer-filled channel. Additionally, they developed a semi-empirical pressure model using experiments to determine the optimal spacer design for ultrafiltration process. Few to investigate the effect of spacer geometry on the mass flow rate and the pressure drop across the channel using numerical simulations. Schwinge et al [84] simulated the flow around different spacer configurations like the cavity, zigzag in 2D (two-dimensional) for a range of Reynolds (Re) numbers from 100 to 1000 and they found that filaments near the wall created larger recirculation than a filament in the center and for large Re numbers the recirculation between the successive filaments influences each other. Karode et al [82] used

3D (three-dimensional) simulations to analyze the flow pattern through symmetric (spacer with equal filament diameter) and asymmetric (spacer with unequal filament diameter) nonwoven spacer channels. They found that the pressure drop across the channel was mainly governed by the loss of fluid momentum at the intersection of the spacer filaments that causes abrupt change in flow directions. Koutsou et al [51] performed simulations on 3D nonwoven spacer geometries on various Re numbers and identified that transition to unsteady flow occurs at very low Re number. In another paper [50], they also studied the effect of both Re and Schmidt (Sc) numbers and calculated the local time-averaged mass transfer coefficient from time-averaged wall mass flux for fixed concentration at the wall. Shakaib et al [85] simulated flow dynamics and mass transfer on parallel spacers (spacer filament aligned with channel axis) and diamond spacer with various flow attack angles. For both kinds of spacers they also varied filament spacing and thickness. They found that the distribution of the mass transfer coefficient is uniform when the spacing between filaments is increased and the fluid flows in a zigzag pattern. The fluid flows in parallel to the filaments when spacing decreases or when the flow attack angle increases. Li et al [59] performed simulations on different periodic unit cells i.e. the smallest possible flow domain with the minimum number of spacer filaments to form periodicity and proposed a periodic cell which can be used to predict the pressure drop through the full-scale spacer. Picioreanu et al [73] simulated the flow in a 3D nonwoven spacer channel and modeled the biofilm formation, accumulation and its growth in the spacer channels. All the previous simulations of flow in spacer filled channels are limited to small scales i.e they considered either 2D cross sections or 3D with just a few filaments. For the first time in 2015, Johannink et al [39] used our open-sourced lattice Boltzmann solver *Musubi* [32] to develop a pressure drop prediction model for woven and nonwoven spacers and also validated the numerical result with experiments.

Regarding the mathematical model, the Nernst-Planck (NP) equation is most widely used to describe the mass transport of ionic species in electrolyte solutions. The limitation of this equation is the fact that they are valid only for infinitely dilute electrolyte solutions due to Fick's law of diffusion description. Fick's law of diffusion considers only the diffusion between an ion and solvent and not between ions. A more accurate description of charge transport in electrolyte solution is given by the Maxwell-Stefan (MS) equations [72, 96]. Unlike Fick's law of diffusion, the MS equations include the multicomponent interaction that occurs in concentrated electrolyte solution. Usually, only the MS equations for a ideal mixture were used for numerical simulation and nonideal effects

were neglected. However, in electrolyte solutions due to the nonlinear concentration gradient near the electrode/electrolyte interface, the nonideal effects must be considered. Psaltis [78] investigated the application of the Nernst-Planck equations and the Maxwell-Stefan equations with ideal and nonideal effects on the electrolyte solution. He found that the concentration predicted by the Maxwell-Stefan is lower than the Nernst-Planck prediction for a particular species and with ternary mixture, the behavior predicted by the Maxwell-Stefan equation was not obtained by the Nernst-Planck equations. Furthermore, he also observed that even though their steady state solution was the same, their transient behavior was different.

## 1.3 Goal of this Thesis

As pointed out in the previous sections, the coupled transport phenomena governing the performance of industrial ED processes are complex and not sufficiently understood. One reason for the limiting insight into the phenomena can be found in the compact design of ED modules prohibiting an experimental in-situ observation. The development of simulation tools allowing the coupled simulation of the transport phenomena can contribute to overcome this unsatisfactory situation.

The goal of this thesis is the development and implementation of an efficient coupled simulation tool for the integrated simulation of the transport phenomena in electrodialysis modules. The development of such a simulation tool requires the derivation of rigorous mathematical models describing the local transport processes with a high accuracy. As the resulting models are large and strongly coupled, appropriate numerical schemes are to be developed allowing for an efficient numerical treatment of the models. The huge computational effort resulting from the fine local resolution of the multiple physical phenomena in large-scale industrially relevant settings is addressed by using massively parallel high performance computing systems.

In this work, the lattice Boltzmann method (LBM) is chosen as the numerical scheme to simulate fluid flow, multicomponent flow and electric potential. The LBM was chosen due to its efficiency in supercomputers and ease in handling complex geometries like spacer structure in the flow channels [12]. To improve the prediction of the behavior of mass transport of ionic species in the flow channels, a sophisticated multicomponent LBM model for nonideal liquid mixture with external diffusive force was developed. This model recovers the Maxwell-Stefan equations for nonideal mixture under the asymptotic limit. This nonideal model is compared

against the ideal model to show the necessity of nonideal models to simulate the charge transport in electrolyte solutions.

The LBM method requires meshes for computation i.e. elemental representation of the computational domain. Due to the spacer geometry in the flow channels, the accuracy of the simulation greatly depends on the approximation of the spacer geometry. Furthermore, a fine resolution is required to resolve the diffusive layer near the membranes. Therefore, an efficient parallel mesh generator was also developed as part of this thesis to generate very large computational meshes for supercomputing needs.

The developed tools (mesh generator *Seeder*, multicomponent extension of the LBM solver *Musubi*, and coupling tool *APESmate*) are implemented as part of an in-house scalable simulation framework called Adaptable Poly-Engineering Simulator (*APES*). These tools allow for the thorough investigation of the transport processes in ED modules by means of dedicated simulation studies and sensitivity analysis. The resulting insight can be directly used to improve the process performance by revising the design of the module or the spacers. Further, the detailed simulation results provide a solid basis for a more accurate detail engineering in industrial applications. The coupling tool *APESmate* is designed in a way that it can couple different solvers in *APES* framework in any combination and thus allows for applications in a number of computational applications.

In this thesis, the flow structure in both woven and nonwoven spacers for various flow attack angles and velocities are investigated to identify the relationship between spacer design and the pressure drop across the channel. Furthermore, the flow structure near the seal corner of the industrial nonwoven spacer is investigated for various velocities and angles between spacer filaments to determine the optimal spacer design with reduced low flow zones. This investigation is performed on the nonwoven spacer because the industrial partner SIEMENS used only this spacer in the prototype of ED module. In addition to pure hydrodynamic simulations, the effects of the spacer geometries on ions transport in the flow channels are also addressed in this work. Furthermore, the coupled simulations were performed to show the effect of ion transport on the electric potential near the membranes and vice versa. Besides investigations, the major contribution of this thesis are the development of the mesh generator *Seeder*, a multicomponent LBM model for nonideal liquid mixtures and a scalable coupling tool *APESmate*. They are presented in detail with their algorithms.

## 1.4 Structure of this Thesis

This thesis is structured as follows: In the next Chapter 2, the governing equations to define various physical processes like ion transport, interaction between ions, bulk flow and electrodynamics within the ED stack are presented. In Chapter 3, the numerical scheme namely the lattice Boltzmann method (LBM), chosen to solve those governing equations is introduced. The black box model for membranes that is used as boundary condition for multicomponent flows is also discussed in this chapter. Next, the coupling strategy employed to couple the numerical approaches on surface and volume are given in Chapter 4. Here, the surface balance equations required to be satisfied on the electrolyte-membrane interface are also introduced. In Chapter 5, the highly scalable simulation framework *APES* is introduced. Here, algorithms and implementation details of the mesh generator *Seeder*, the LBM solver *Musubi* and the coupling tool *APESmate* are presented in detail since they are major contributions of this thesis. In addition to the implementation details, the performance of *Seeder* and *Musubi* are also presented. Then, the presented numerical approaches and implementations are validated using several numerical test cases against analytical solution or experiment in Chapter 6. Here, coupling between multicomponent LBM and LBM for electric potential is validated and coupling between multicomponent LBM and membrane black box model is verified. In Chapter 7, the results of large scale flow simulations with different spacer geometries are presented. The effect of spacer design on flow structures in the middle of the channel and near the sealed corner with low flow zones are discussed in detail. In addition to pure hydrodynamic flow, the results of multicomponent flow simulations performed using multicomponent LBM are also presented. Finally, this chapter is concluded with the simulation results of a repeating unit in the ED process consisting of a dilute and a concentrate channel with membranes. Here, the electric effects on the flow are added to the physical phenomena that are simulated. In the last Chapter 8, a short summary of contributions and conclusion drawn from this work are listed along with list of future work.

# 2 Mathematical models

This chapter introduces the fundamental governing equations to represent different physics involved in the ED process. An overview of the physical subsystems in the ED process and their interactions are shown in Figure 2.1. The multicomponent transport in the spacer-filled flow channel requires a description of mass balance, momentum balance for bulk mixture, movement of ionic species. Furthermore, the description of interaction between species in the electroneutral region in the bulk and the nonelectroneutral region near the membrane is required. In membranes, where only the counter-ions are transported, a description of species mass balance and its transport is required. In both, flow channels and membranes, the electrodynamic is required to describe the interaction between migration of ions and the applied electric potential. Each of these physical systems must be coupled with each other to describe the ED process completely. The coupling approach used to couple these subsystems is presented in Chapter 4. All the descriptions in this section are restricted to isothermal conditions and we neglect any chemical reactions and energy dissipation effects. Thus, energy balance equations are not considered.

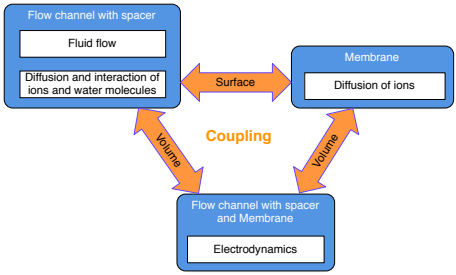



**Figure 2.1** Multiphysical heterogeneous system representing different physics involved in ED process [65].

The mathematical model is introduced for a general homogeneous phase in a three dimensional spatial domain for a set $S = \{1, ..., n\}$ of charged or neutral species. Without a loss of generality, the first $n - 1$ species

represent ionic species of arbitrary charge and species $n$ represents a neutral solvent i.e water molecules in an electrolyte solution. Before introducing the governing equations, the fundamental measures for concentration, velocities and diffusive fluxes are introduced.

## 2.1 Concentration measures, velocities and diffusive fluxes

The set of modeling equations is based on the definition of the *mass averaged mixture velocity* $\mathbf{v}$ as the reference velocity

$$\mathbf{v} = \sum_{k=1}^{n} y_k \mathbf{v}_k, \tag{2.1}$$

where, $y_k = \rho_k/\rho$ is the *mass fraction* of species $k$ and $\mathbf{v}_k$ is the *velocity* of species $k$. $\rho_k$ and $\rho = \sum_k \rho_k$ are *mass density* of species $k$ and *total mixture mass density* of the electrolyte solution respectively. $n$ is the *number of species* in the mixture. The *mass flux* density $\boldsymbol{n}_k$ of species $k$ is defined by

$$\boldsymbol{n}_k = \rho_k \mathbf{v}_k \tag{2.2}$$

and summing up these component fluxes results in the *total mass flux*

$$\boldsymbol{n}_t = \sum_{k=1}^{n} \boldsymbol{n}_k = \rho \mathbf{v}. \tag{2.3}$$

Above measures can also be defined in molar form for liquid mixtures. The *molar density or molar concentration* $c_k$ of species $k$ is related to the mass density $\rho_k$ of species $k$ by

$$c_k = \frac{\rho_k}{M_k} \tag{2.4}$$

where, $M_k$ is the *molecular weight* of species $k$. The *molar flux density* $\boldsymbol{N}_k$ of species $k$ is defined by

$$\boldsymbol{N}_k = c_k \mathbf{v}_k \tag{2.5}$$

and summing up these component molar fluxes gives the *total molar flux*

$$\boldsymbol{N}_t = \sum_{k=1}^{n} \boldsymbol{N}_k = c_t \mathbf{w}. \tag{2.6}$$

Here, $c_t = \sum_{k=1}^{n} c_k$ is the *total mixture molar concentration* and $\mathbf{w} = \sum_{k=1}^{n} \chi_k \mathbf{v}_k$ is the *molar averaged mixture velocity.* $\chi_k = c_k/c_t$ is the *mole fraction* of species $k$.

In addition to mass and molar fluxes, the diffusion flux is introduced which defines the flux of species $k$ in relation to the reference mixture velocity. The *mass diffusion flux density* $\boldsymbol{j}_k$ with respect to the mass averaged mixture velocity is

$$\boldsymbol{j}_k = \rho_k \left(\mathbf{v}_k - \mathbf{v}\right), \tag{2.7}$$

with the closure relation

$$\sum_{k=1}^{n} \boldsymbol{j}_k = 0. \tag{2.8}$$

Eq. 2.7 can be rewritten in terms of the *mass flux density* $\boldsymbol{n}_k$ as

$$\boldsymbol{j}_k = \boldsymbol{n}_k - \rho_k \mathbf{v} \tag{2.9}$$

Similarly, the *molar diffusion flux density* $\boldsymbol{J}_k$ is defined with respect to mole average mixture velocity by

$$\boldsymbol{J}_k = c_k \left(\mathbf{v}_k - \mathbf{w}\right) = \boldsymbol{N}_k - c_k \mathbf{w}, \tag{2.10}$$

and with the closure relation

$$\sum_{k=1}^{n} \boldsymbol{J}_k = 0. \tag{2.11}$$

The relation between the *mole fraction* $\chi_k$ and the *mass fraction* $y_k$ of a species $k$ is expressed by

$$\chi_k = y_k \frac{M}{M_k}, \tag{2.12}$$

where $M$ is the molecular weight of the mixture which can be determined by

$$M = \sum_{k=1}^{n} \chi_k M_k. \tag{2.13}$$

By definition of mole fraction $\chi_k$ and mass fraction $y_k$, their sum reduces to unity i.e.

$$\sum_{k=1}^{n} y_k = 1,$$

$$\sum_{k=1}^{n} \chi_k = 1$$

respectively.

## 2.2 Multicomponent transport equations

In this section, the governing equations to describe the mass transport of ionic species or components in the spacer-filled flow channels and membranes are introduced. Before getting into equations, lets look at different fluxes that transport ions in the flow channels and membranes. Figure 2.2 illustrates the flux densities of cation and anion in the flow channel and membrane and the concentration profiles of the ions and salt near the surface of CEM under steady state conditions [91]. In the figure, the symbols $N$ and $c$ denote the flux density and the concentration, the superscripts $mig$ and $diff$ refer to migration and diffusion, the superscripts $d$ and $c$ refer to diluate and concentrate solution, and the superscripts $b$ and $m$ refer to bulk phase and membrane surface respectively, and the subscripts $c$, $a$ and $s$ refers to cation, anion and salt respectively. In the flow channels, the bulk mixture with ions flows along the length of the channel by an applied pressure drop and is described by the convective flux $\boldsymbol{N}_k^{conv}$. In both, flow channels and membranes, the migration of ions towards respective electrodes due to an applied electric potential is described by migrative flux $\boldsymbol{N}_k^{mig}$. Finally, the concentration gradient which gets established in the channel near the membrane surface due to the difference in the transport number of ions in the solution and the membrane results in the diffusive flux $\boldsymbol{N}_k^{diff}$. Therefore, while selecting the governing equations to describe the mass transport of ionic species, all those above mentioned fluxes need to be considered. Additionally, the interaction between ions must also be considered due to an increase in ion concentration in the concentrate channel.

One more physics to consider is the interaction of ion concentration with an electric potential in the diffusive boundary layer near both sides of the membrane surface. In dilute channels, the concentration of salt decreases near the membrane surface while the concentration of salt on the other side of the membrane facing the concentrate channel increases. This creates a concentration gradient of salt between the channels, which results in diffusive transport of salt from the concentrate to the dilute channel. Due to this concentration gradient in the flow channel near the membrane surface, the solution is locally nonelectroneutral. However, the bulk solution remains electroneutral for all times. The local nonelectroneutral distribution of ions results in a potential gradient that acts as an additional migrative force on the ions. Therefore, in the diffusive region,

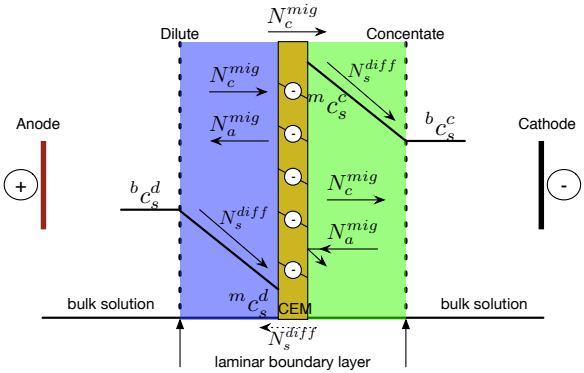



**Figure 2.2** Illustration of concentration profiles of a salt in the boundary layer on both sides of the CEM and the fluxes of cations and anions in the boundary layer and in the membrane.

the interaction of ions and electric field must be considered. Thus, the effective electric field on the ions consists of the applied external electric field and the induced electric field of the ions.

In this work, the more sophisticated Maxwell-Stefan Eq. 2.26 with an external driving forces is considered to describe the mass transport of ions in both dilute and concentrate flow channels. For infinitely dilute solutions, the Maxwell-Stefan (MS) equation can be simplified to the Nernst-Planck Eq. 2.41 to model the flux transport of species [72]. However, in the ED setup along the length of both dilute and concentrate flow channels, the concentration decreases in dilute channels and increases in concentrate channels. Thus, the solution in concentrate channels is not sufficiently dilute anymore to be described by the Nernst-Planck equation. Furthermore, for concentrate solutions with more than two components, the interaction between the components must be considered. Additionally, the nonideal liquid model must be considered for concentrate solutions to include the effects of intermolecular forces between same and different components. Therefore, the Maxwell-Stefan equation for nonideal liquid model is chosen for flow channels to incorporate the interaction between ionic components. On the other hand, in membranes where only counter-ions are migrated, the interaction between ions can be neglected resulting in ideal liquid model. Hence, the Nernst-Planck (NP) relation with external driving forces for ideal liquid is considered to describe the mass transport of ions through membranes. The equations presented in this section are

based on [96].

In general, the mass conservation equation for $k \in \{1, ..., n\}$ component when neglecting chemical reactions is given by

$$\frac{\partial \rho_k}{\partial t} + \nabla \cdot (\rho_k \mathbf{v}_k) = 0. \tag{2.14}$$

Substituting Eq. 2.9 in above equation leads to

$$\frac{\partial \rho_k}{\partial t} + \nabla \cdot (\rho_k \mathbf{v}) = -\nabla \cdot \boldsymbol{j}_k \tag{2.15}$$

The above equation can also be expressed in molar form as

$$\frac{\partial c_k}{\partial t} + \nabla \cdot (c_k \mathbf{w}) = -\nabla \cdot \boldsymbol{J}_k \tag{2.16}$$

Both Eq. 2.15 and Eq. 2.16 are equivalent in nature defining the transport of individual species. Since our focus is on flow in electrolyte solutions, all equations are presented in molar form in the rest of this section. The second term in above transport equations define the convective transport of component $k$ and the right hand side defines the diffusive transport of component $k$. The diffusive flux $\boldsymbol{J}_k$ or $\boldsymbol{j}_k$ is provided by some phenomenological models: the Fick model and the Maxwell-Stefan model [96]. The Fick's diffusion model is a very popular phenomenological model for expressing the diffusion fluxes for binary mixtures and it is also valid for dilute mixtures. The Nernst-Planck (NP) equation is an extension of the Fick's diffusion to include the migration of charged particles due to electrostatic forces. The Maxwell-Stefan model is a model for describing diffusion in multicomponent systems. In the following, both these models are discussed.

### 2.2.1 Maxwell-Stefan equations

In the spacer-filled flow channel with an electrolyte solution, the interaction between species are given by the generalized Maxwell-Stefan's equations which relates the driving force $\boldsymbol{d}_k$ for diffusion of species k and the relative velocity between species $(\mathbf{v}_k - \mathbf{v}_l)$ as

$$\boldsymbol{d}_k = -\sum_{l=1, l\neq k}^{n} \frac{\chi_k \chi_l \left(\mathbf{v}_k - \mathbf{v}_l\right)}{\mathcal{D}_{k,l}} \tag{2.17}$$

where $\mathcal{D}_{k,l}$ is the Maxwell-Stefan diffusion coefficient between species $k$ and $l$ which represent the friction between those species. Eq. 2.17 can also

be written in terms of the molar diffusive flux as

$$\boldsymbol{d}_k = \sum_{l=1, l\neq k}^{n} \frac{(\chi_k \boldsymbol{J}_l - \chi_l \boldsymbol{J}_k)}{c_t \mathcal{D}_{k,l}}. \tag{2.18}$$

The molar diffusive flux $\boldsymbol{J}_k$ and the molar fraction $\chi_k$ are related by the driving force $\boldsymbol{d}_k$. From the theory of irreversible thermodynamics, the driving force $\boldsymbol{d}_k$ for diffusion of species $k$ in nonideal fluids is given by [96]

$$\boldsymbol{d}_k = \frac{\chi_k}{RT} \nabla_{T,P} \overline{\mu}_k + \frac{c_k \tilde{V}_k - y_k}{c_t RT} \nabla P - \boldsymbol{F}_k \qquad k = 1, ..., n. \tag{2.19}$$

The physical interpretation of $c_t RT \boldsymbol{d}_k$ is that it represents the force acting on species $k$ per unit volume of mixture tending to move species $k$ relative to the solution. $\overline{\mu}_k$ is the *molar chemical potential*. $\tilde{V}_k$ is the *specific molar volume*. $R$ is the universal gas constant. $P$ and $T$ are the mixture pressure and temperature respectively. $\nabla_{T,P} \overline{\mu}_k$ represents the gradient of molar chemical potential calculated at constant temperature and pressure and it signifies the diffusion process where species move from higher concentration to lower concentration. The second term in Eq. 2.19 describes the influence of pressure gradients with respect to the difference of the volume fraction $c_k(\boldsymbol{x}, t)\ \tilde{V}_k(\boldsymbol{x}, t)$ and the mass fraction $y_k(\boldsymbol{x}, t)$. The last term $\boldsymbol{F}_k$ represents the external diffusive driving force on species $k$ and it is given as [72, 96]

$$\begin{aligned} \boldsymbol{F}_k &= \frac{1}{c_t RT} \rho_k \left( \frac{\hat{\boldsymbol{F}}_k}{M_k} - \sum_{l=1}^{n} y_l \frac{\hat{\boldsymbol{F}}_l}{M_l} \right) \\ &= \frac{1}{c_t RT} \left( c_k \hat{\boldsymbol{F}}_k - y_k \sum_{l=1}^{n} c_l \hat{\boldsymbol{F}}_l \right) \end{aligned} \tag{2.20}$$

where $\hat{\boldsymbol{F}}_k$ is the external body force per mole acting on species $k$. For electrolyte systems like ED, the ionic species migrates due to an applied external electrical field and the force $\hat{\boldsymbol{F}}_k$ is given as

$$\hat{\boldsymbol{F}}_k = z_k \mathcal{F} \boldsymbol{E}. \tag{2.21}$$

Here, $\boldsymbol{E}$ represents applied electric field and its related a scalar electric potential $\psi$ as $\boldsymbol{E} = -\nabla\psi$. $z_k$ is the specific charge number of species $k$ and $\mathcal{F}$ is the Faraday constant. In general, the external force term in Eq. 2.20 can be decomposed into two parts: The first part represents the total external force which acts on species $k$ and the second part represents the

fraction of total mixture force acting on species $k$. It is worth mentioning that in the ED process, the external electrical force which drives the ions through the membranes must be high enough to overcome the friction or the resistance from the electrolyte solution on the movement of ionic species.

For nonideal fluids, the gradients of the molar chemical potential $\overline{\mu}_k$ in Eq. 2.19 can be expressed in terms of molar fraction $\chi_k$ and activity coefficients $\gamma_k$ based on mole fractions as [96]

$$\begin{aligned}\frac{\chi_k}{RT}\nabla\overline{\mu}_k &= \sum_{j=1}^{n-1}\left(\delta_{k,j}+\chi_k\frac{\partial ln(\gamma_k)}{\partial\chi_j}\bigg|_{T,p,\Sigma}\right)\nabla\chi_k \\ &= \sum_{j=1}^{n-1}\Gamma_{k,j}\nabla\chi_j. \end{aligned} \tag{2.22}$$

Here, $\Gamma_{k,j}$ is the thermodynamic factor and it is given as

$$\Gamma_{k,j} = \delta_{k,j}+\chi_k\frac{\partial ln(\gamma_k)}{\partial\chi_j}\bigg|_{T,p,\Sigma} \tag{2.23}$$

where, $\delta_{k,j}$ is the Kronecker delta. For the evaluation of $\Gamma_{k,j}$ from molar based activity coefficients $\gamma_k$ see [60, 96]. This description of the chemical potential gradients is of significant importance since it allows a straightforward inversion of the flux-force relationship. That way, an explicit expression of the flux or composition vector can be obtained. The symbol $\Sigma$ in Eq. 2.23 means that the differentiation of $ln(\gamma_k)$ with respect to mole fraction $\chi_j$ must be carried out by keeping the mole fraction of all other species constant except the $n^{th}$. The subscripts $T$ and $P$ denote that this differentiation is to be calculated at constant temperature and pressure. Substituting Eq. 2.22 in Eq. 2.19 and neglecting the pressure diffusion term since its effect is negligible [38], results in

$$\boldsymbol{d}_k = \sum_{j=1}^{n-1}\Gamma_{k,j}\nabla\chi_j - \boldsymbol{F}_k. \tag{2.24}$$

For ideal fluids, the activity coefficients becomes unity $\gamma_k = 1$ and the thermodynamic factor reduces to identity matrix $\boldsymbol{\Gamma} = \boldsymbol{I}$. Thus, the driving force $\boldsymbol{d}_k$ for diffusion of species $k$ in ideal fluids can be written as

$$\boldsymbol{d}_k = \nabla\chi_k - \boldsymbol{F}_k. \tag{2.25}$$

Relating Eq. 2.24 and Eq. 2.18 gives the closure relation of Maxwell-Stefan equation for nonideal fluids with an external force as

$$\sum_{j=1}^{n} \Gamma_{k,j} \nabla \chi_j - \boldsymbol{F}_k = \sum_{l=1, l\neq k}^{n} \frac{(\chi_k \boldsymbol{J}_l - \chi_l \boldsymbol{J}_k)}{c_t \mathcal{D}_{k,l}} \tag{2.26}$$

In case of ideal fluids, the above equation can be written as

$$\nabla \chi_k - \boldsymbol{F}_k = \sum_{l=1, l\neq k}^{n} \frac{(\chi_k \boldsymbol{J}_l - \chi_l \boldsymbol{J}_k)}{c_t \mathcal{D}_{k,l}} \tag{2.27}$$

which is the Maxwell-Stefan relation with external force for ideal fluids.

Only $n-1$ of Eqs. 2.26 and 2.27 are independent because the $\nabla \chi_k$ sum to zero; the $n^{th}$ component gradient is given by

$$\nabla \chi_n = -\nabla \chi_1 - \nabla \chi_2 - \nabla \chi_3 \cdots - \nabla \chi_{n-1} \tag{2.28}$$

$$= -\sum_{k=1}^{n-1} \nabla \chi_k. \tag{2.29}$$

There are also other important constraint to be considered when using the Maxwell-Stefan formulation Eqs. 2.26 and 2.27. The Maxwell-Stefan diffusion coefficients are symmetric

$$\mathcal{D}_{k,l} = \mathcal{D}_{l,k}$$

to fulfill the Onsager reciprocal conditions [68]. The concentration dependent binary Maxwell-Stefan diffusion coefficients have been estimated by [17] using the equation

$$\begin{aligned} \mathcal{D}_{k,l} = \tilde{\mathcal{D}}_1(k,l) + \tilde{\mathcal{D}}_2(k,l) \cdot I_s + \tilde{\mathcal{D}}_3(k,l) \cdot I_s^{3/2} \\ + \tilde{\mathcal{D}}_4(k,l) \cdot I_s^2 + \tilde{\mathcal{D}}_5(k,l) \cdot \sqrt{I_s}, \end{aligned} \tag{2.30}$$

and experimental data from [100]. Parameters $\tilde{\mathcal{D}}_1(k,l), \cdots, \tilde{\mathcal{D}}_5(k,l)$ are species dependent coefficients and $I_s$ is the molar ionic strength of the solution in $\mathrm{mol\,m^{-3}}$

$$I_s = \frac{1}{2} \sum_{k=1}^{n} c_k z_k^2. \tag{2.31}$$

For a binary (1:1) electrolyte solution like NaCl where each ion is singly-charged, the ionic strength is equal to the concentration of ions i.e. $I_s = c_{NaCl} = c_{Na} = c_{Cl}$. For a liquid NaCl solution, the species diffusivity

| (k,l) | $\tilde{D}_1$ | $\tilde{D}_2$ | $\tilde{D}_3$ | $\tilde{D}_4$ | $\tilde{D}_5$ |
|---|---|---|---|---|---|
| | $1 \times 10^{-9}$ | $1 \times 10^{-13}$ | $1 \times 10^{-15}$ | $1 \times 10^{-17}$ | $1 \times 10^{-12}$ |
| | | | $[\mathrm{m^2\,s^{-1}}]$ | | |
| Na,Cl | 0 | 0.802 | −0.209 | −0.703 | 2.18 |
| Na,$H_2O$ | 1.34 | −0.306 | −3.91 | 3.77 | −1.77 |
| Cl,$H_2O$ | 2.04 | −2.24 | −3.79 | 3.78 | 8.32 |

**Table 2.1** Parameter values $\tilde{D}_1(k,l), \cdots, \tilde{D}_5(k,l)$ for a liquid NaCl solution as reported in [17].

coefficients are given in Table 2.1 and the correlation Eq. 2.30 can be applied for a concentration range of $c_{NaCl} = 0 \ldots 5000\ \mathrm{mol\,m^{-3}}$.

Last, for ideal fluids, by definition of $\boldsymbol{d}_k$, the sum of the diffusive driving forces vanishes

$$\sum_{k=1}^{n} \boldsymbol{d}_k = 0 \tag{2.32}$$

due to the Gibbs-Duhem restriction [96], which results in only $n-1$ independent forces.

### 2.2.2 Nernst-Planck equations

In membranes, where only the counter-ions are transported, the interaction between ions is neglected and only the interaction between ion and solvent is considered. Therefore, the Maxwell-Stefan diffusive driving force given in Eq. 2.18 is reduced to

$$\boldsymbol{d}_k = -\boldsymbol{J}_k/(c_t \mathcal{D}_k). \tag{2.33}$$

Substituting the above equation in Eq. 2.24 gives the direct relation between diffusive flux and mole fraction of species as

$$\nabla \chi_k - \boldsymbol{F}_k = -\boldsymbol{J}_k/(c_t \mathcal{D}_k) \tag{2.34}$$

where $D_k$ is the effective MS diffusion coefficient of species $k$ in the multicomponent mixture. Above equation can also be rewritten as

$$\boldsymbol{J}_k = -\mathcal{D}_k \nabla c_k + c_t \mathcal{D}_k \boldsymbol{F}_k \tag{2.35}$$

which defines the diffusive flux $\boldsymbol{J}_k$ of species $k$. Once again there are only $n-1$ independent diffusive fluxes since the sum of diffusive fluxes vanishes. Thus, the flux density $\boldsymbol{N}_k$ of species $k$ can be explicitly written as

$$\boldsymbol{N}_k = -\mathcal{D}_k \nabla c_k + c_k \mathbf{w} + c_t \mathcal{D}_k \boldsymbol{F}_k. \tag{2.36}$$

With an applied electrical field $\boldsymbol{E}$ as an external force, above equation can be written as

$$\boldsymbol{N}_k = \underbrace{-\mathcal{D}_k \nabla c_k}_{\text{diffusion}} + \underbrace{c_k \mathbf{v}}_{\text{convection}} + \underbrace{\left( z_k c_k - y_k \sum_{l=1}^{n} z_l c_l \right) \frac{\mathcal{D}_k \mathcal{F}}{RT} \boldsymbol{E}}_{\text{electro-migration}}. \tag{2.37}$$

Here, the first term represents the diffusion of species due to the concentration gradient and the second term represents the convection of species due to the pressure gradient across the stack and the last term represents the electro-migration of ionic species due to an applied external electric field. Note that both diffusion and convection transport of cations and anions are in the same direction and only the electro-migration term discriminates ions by their charge number $z_k$ and transport them in different directions. In the ED process, the nonelectroneutral region exists in the flow channels only near the membranes due to charge separation and formation of diffusion layer. However, far from membrane the bulk solution remains electroneutral and satisfies the electroneutrality condition

$$\sum_{k=1}^{N} z_k c_k = 0. \tag{2.38}$$

Additionally, the transport of counter ions in the ion exchange membrane satisfies the electroneutrality condition

$$\sum_{\substack{k=1 \\ k \neq fix}}^{N} z_k c_k^{IEM} + z_{fix} c_{fix}^{IEM} = 0 \tag{2.39}$$

where, the superscript $IEM$ refers to the ion-exchange membrane, the subscript $fix$ refers to charges fixed to the membrane polymer. With an electroneutrality condition in bulk mixture and in membranes, there is no net electrical force acting as a whole. Thus, for electroneutral solution, the flux density $\boldsymbol{N}_k$ of species given in Eq. 2.36 can be simplified to

$$\boldsymbol{N}_k = -\mathcal{D}_k \nabla c_k + c_k \mathbf{w} + \frac{z_k \mathcal{D}_k \mathcal{F}}{RT} c_k \boldsymbol{E}. \tag{2.40}$$

Combining Eq. 2.16 and Eq. 2.40 results in

$$\frac{\partial c_k}{\partial t} = -\nabla \cdot \left( -\mathcal{D}_k \nabla c_k + c_k \mathbf{w} + \frac{z_k \mathcal{D}_k \mathcal{F}}{RT} c_k \boldsymbol{E} \right) \tag{2.41}$$

which is the Nernst-Planck equation with an external electrical force. Its worth mentioning that the Nernst-Planck equation is generally used to describe mass transport of species in an infinitely dilute electrolyte solution.

### 2.2.3 Electrical current and Ion transport number

The current in the ED process is carried by the charged ionic species in the electrolyte solution. Thus, the electrical current density $\boldsymbol{i}^d$ can be expressed in terms of the flux density $\boldsymbol{N}_k$ of the species as

$$\boldsymbol{i}^d = \mathcal{F}\sum_{k=1}^{n} z_k \frac{\rho_k}{M_k}\mathbf{v}_k = \mathcal{F}\sum_{k=1}^{n} z_k c_k \mathbf{v}_k = \mathcal{F}\sum_{k=1}^{n} z_k \boldsymbol{N}_k. \tag{2.42}$$

For the electrolyte solution with ionic species, the species balance Eq. 2.15 and Eq. 2.16 are completed by the charge balance equation

$$\frac{\partial \rho^e}{\partial t} + \nabla \cdot \boldsymbol{i}^d = 0, \tag{2.43}$$

where $\rho^e$ is the electrical charge density defined as

$$\rho^e = \mathcal{F}\sum_{k=1}^{n} z_k \frac{\rho_k}{M_k} = \mathcal{F}\sum_{k=1}^{n} z_k c_k. \tag{2.44}$$

For electroneutral solutions, the charge density becomes zero. Therefore, the charge balance equation reduces to

$$\nabla \cdot \boldsymbol{i}^d = 0. \tag{2.45}$$

Further relations between charge density, current density with electric and magnetic field are given in Section 2.4. Substituting the flux density $\boldsymbol{N}_k$ of species Eq. 2.40 in Eq.2.42 and setting $\boldsymbol{E} = -\nabla\psi$, where $\psi$ is the scalar electric potential, the current density $\boldsymbol{i}^d$ can be written as

$$\boldsymbol{i}^d = -\mathcal{F}\sum_{k=1}^{n} z_k \mathcal{D}_k \nabla c_k + \mathcal{F}\mathbf{w}\sum_{k=1}^{n} z_k c_k - \frac{\mathcal{F}^2}{RT}\nabla\psi \sum_{k=1}^{n} z_k^2 \frac{\mathcal{D}_k}{RT} c_k. \tag{2.46}$$

In the flow channels, the current is carried by both cation and anion while in the IEM the current is mostly carried by counter-ions. Even though the current is carried by both ionic species, the fraction of overall current carried by individual ionic species differs. Therefore, the fraction of current carried by specific ionic species $k$ is expressed by the ion transport number or transference number $T_k$ and it is given as [92, 93]

$$T_k = \frac{z_k \boldsymbol{N}_k}{\sum_l z_l \boldsymbol{N}_l} \tag{2.47}$$

with the constraint that the sum of the ion transport number is unity. The transport number can also be expressed in terms of current density $\boldsymbol{i}^d$

by multiplying and dividing Faraday's constant to above equation, which results in

$$T_k = \frac{z_k \mathcal{F} \boldsymbol{N}_k}{\mathcal{F} \sum_l z_l \boldsymbol{N}_k} = \frac{z_k \mathcal{F} \boldsymbol{N}_k}{\boldsymbol{i}^d}. \tag{2.48}$$

The transport number of cations and anions in the flow channels does not differ much since the current is carried by both ions. But due to the high concentration of counter-ions in the IEM, the transport number of counter-ions is close to 1. Another important parameter of the membrane is the permselectivity $\Psi$ that describes the degree to which a membrane passes an ion of one charge and retains an ion of opposite charge. Therefore, it defines the performance of a membrane in ED processes. The permselectivity of cation and anion exchange membranes are given by [92, 93]

$$\Psi^{CEM} = \frac{T_c^{CEM} - T_c^e}{T_a^e} \quad \text{and} \quad \Psi^{AEM} = \frac{T_a^{AEM} - T_a^e}{T_c^e}. \tag{2.49}$$

The subscripts $c$ and $a$ refer to cation and anion respectively. The superscripts *cem*, *aem* and $e$ refer to CEM, AEM and an electrolyte solution respectively.

Assuming that the membrane is very thin and there exists no concentration gradient, the first term in Eq. 2.46 will vanish. Additionally, due to the electroneutrality condition in membranes, the second term in Eq. 2.39 becomes zero. Thus, the current density $\boldsymbol{i}^{d,mem}$ through membranes can be defined as

$$\boldsymbol{i}^{d,mem} = -\mathcal{F}^2 \nabla\psi \sum_{k=1}^{n} z_k^2 \frac{\mathcal{D}_k}{RT} c_k \tag{2.50}$$

which can be compared to the well-known Ohm's law expression

$$\boldsymbol{i}^{d,mem} = -\kappa^m \nabla\psi. \tag{2.51}$$

Here, $\kappa^{mem}$ is the specific membrane conductance

$$\kappa^{mem} = \frac{1}{R^{mem}} = \mathcal{F}^2 \sum_{k=1}^{n} z_k^2 \frac{\mathcal{D}_k}{RT} c_k \tag{2.52}$$

and $R^{mem}$ is the specific membrane resistance. With no concentration gradient and electroneutrality in the membranes, there is only a migrative flux $\boldsymbol{N}_k^{mig,mem}$ which drives the ionic species in the membranes and it can be explicitly expressed as

$$\boldsymbol{N}_k^{mig,mem} = \frac{T_k^{mem}}{z_k \mathcal{F}} \boldsymbol{i}^{d,mem} = -\frac{z_k \mathcal{D}_k \mathcal{F}}{RT} c_k^{mem} \nabla\psi \tag{2.53}$$

where $c_k^{mem}$ is the molar concentration of species $k$ in the membrane and $T_k^{mem}$ is the ion transport number of species $k$ in the membrane

$$T_k^{mem} = \frac{z_k^2 \mathcal{D}_k c_k^{mem}}{\sum_l z_l^2 \mathcal{D}_l c_l^{mem}}. \tag{2.54}$$

## 2.3 Mixture flow - Incompressible Navier-Stokes equations

In the spacer-filled flow channel, the mixture flow is considered to be an isothermal, incompressible and in the low-Mach regime. The mass and momentum conservation equation with external forces are given by the incompressible Navier-Stokes equation of the form

$$\nabla \cdot \mathbf{v} = 0 \tag{2.55}$$

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\nabla \cdot \mathbf{v} = -\nabla \tilde{P} + \nu \nabla^2 \mathbf{v} + \underbrace{\boldsymbol{g} + \frac{\rho^e}{\rho}\hat{\boldsymbol{E}}}_{=:\boldsymbol{F}} \tag{2.56}$$

respectively for a Newtonian fluid. The external force field $\boldsymbol{F}$ is the mixture force that acts on all the components in same way. $\boldsymbol{g}$ and $\boldsymbol{E}$ are forces due to the gravitational field and the electrical field respectively. The mass conservation equation or continuity Eq. 2.55 can be obtained by summing the species balance Eq. 2.14 over $n$ species and mixture density as constant due to incompressibility i.e. $\rho = \rho^0$, where $\rho^0$ is the constant fluid density. The momentum conservation Eq. 2.56 is obtained from Newton's second law of motion which states that the rate of change of momentum of the fluid is equal to the sum of the forces applied on it. Here, $\tilde{P} = P/\rho$ is the kinematic pressure, $\nu = \mu/\rho$ is the kinematic viscosity. $\mu$ is the dynamic viscosity, $\boldsymbol{F}$ denotes the external force and $\rho^e$ the charge density of the mixture.

In the flow channel, the multicomponent flow can be characterized by dimensionless quantities like the Reynolds and Schmidt number [14]. The Reynolds number can be interpreted as the ratio of inertial forces to viscous forces as

$$Re = \frac{\mathbf{v} L_c}{\nu} \tag{2.57}$$

where $L_c$ is the characteristic length of the flow. The Schmidt number is defined as the ratio of viscous diffusion rate to molecular mass diffusion rate as

$$Sc = \frac{\nu}{D_{k,l}}. \tag{2.58}$$

## 2.4 Electrodynamics - Maxwell's equations

In the ED process, the ionic species are transported towards and through the membrane by the applied electric potential between the anode and cathode. The mixture in the bulk region of the flow channel remains electroneutral $\sum_k z_k c_k = 0$ but near the membrane it becomes nonelectroneutral and creates a concentration gradient. The nonelectroneutral region is referred as the diffusion layer and it is usually in the range of nanometers. The concentration gradient in the diffusion layer affects the potential field. Furthermore, the movement of ionic species might also induce some magnetic field but usually this is neglected assuming it is very small. Since the effect of magnetic field in the ED stack is unknown, the full Maxwell equations of the form

$$\nabla \cdot \boldsymbol{D} = \rho^e, \tag{2.59a}$$

$$\nabla \cdot \boldsymbol{B} = 0, \tag{2.59b}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} + \nabla \times \boldsymbol{E} = 0, \tag{2.59c}$$

$$\frac{\partial \boldsymbol{D}}{\partial t} - \nabla \times \boldsymbol{H} = -\boldsymbol{i}^d \tag{2.59d}$$

are considered at first to describe the spatial and temporal evolution of electro-magnetic fields in the entire ED unit. Here, $\boldsymbol{D}$ denotes the electric displacement field, $\boldsymbol{E}$ the electric field, $\boldsymbol{H}$ the magnetization and $\boldsymbol{B}$ the magnetic field. The charge density $\rho^e$ and the current density $\boldsymbol{i}^d$ are given in Eq. 2.44 and Eq. 2.42. Note that $\rho^e$ and $\boldsymbol{i}^d$ must satisfy the charge conservation Eq. 2.43 for all times $t \geq 0$ and they are implicitly satisfied when the displacement field $\boldsymbol{D}$ fulfills Gauss' law Eq. 2.59a and the magnetic field $\boldsymbol{B}$ fulfills the magnetic monopole constraint Eq. 2.59b for all times. In case of a simple dielectric, where the electric permittivity $\varepsilon$ and the magnetic permeability $\tilde{\mu}$ are considered constant, the displacement field, electric field, magnetic field and magnetization are correlated by

$$\boldsymbol{D} = \varepsilon\, \boldsymbol{E}, \tag{2.60}$$

$$\boldsymbol{B} = \tilde{\mu}\, \boldsymbol{H}. \tag{2.61}$$

In this work which was part of the HISEEM [1] project, initially the full Maxwell equations were considered to couple electrodynamics with multicomponent flows. However, some complications were encountered due

[1] Hocheffiziente Integrierte Simulation von Elektromembranverfahren zur Entsalzung von Meerwasser

to the difference in time scales between the Maxwell solver and the multicomponent solver. The time step size of the Maxwell solver is much smaller than that of the multicomponent solver since the Maxwell solver time step depends on the speed of light and the multicomponent solver time step depends on the speed of sound. Another problem with this coupling is that it is costly to satisfy the Gauss law at all times since ions are transported with much larger time scale than the electric field. Therefore, to circumvent this problem, the magnetic effect was neglected which reduces the full Maxwell equations to the Poisson equation. When the magnetic induction of the electrolyte solution is neglected, which is reasonable in most electrochemical systems, the electric field $\boldsymbol{E}$ can be assumed to be irrotational. This simplifies Faraday's law according to

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \approx 0. \tag{2.62}$$

That way, the electric field can be interpreted as negative gradient of the scalar electric potential $\psi$

$$\boldsymbol{E} = -\nabla\psi. \tag{2.63}$$

Additionally, if the dielectric permittivity $\varepsilon$ is assumed to be constant, Gauss Law $\nabla \cdot (\varepsilon \boldsymbol{E}) = \rho^e$ can be simplified to the well known Poisson's equation,

$$\nabla^2\psi = -\frac{1}{\varepsilon}\rho^e. \tag{2.64}$$

Note, that the Poisson's equation Eq. 2.64 and the charge balance in form of Eq. 2.44 already imply Ampere's law. The description of the electric field can be simplified by using the assumption of a locally electroneutral solution. This leads to the algebraic constraint

$$0 = \rho^e(\boldsymbol{x}, t) = \mathcal{F}\sum_{k=1}^{N} z_k \frac{\rho_k(\boldsymbol{x}, t)}{M_k}, \tag{2.65}$$

also known as the electroneutrality condition. In a simplified model of ionic mass transfer Eq. 2.65 can be used instead of the Poisson equation. Thus, in the electroneutral region, the potential gradient is linear. However, in the flow channel near the membrane where there is a concentration gradient, a nonelectroneutral region exists, usually referred to as diffusive region. Here, the charge density is non-zero and contributes to the nonlinear potential gradient. This additional potential gradient in the diffusion region is called the diffusion potential and the current in this region is called diffusion current.

## 2.5 Conclusion

In this chapter, the mathematical equations describing the various physical phenomena like transport of ionic species, bulk fluid and interaction between ionic species and electric potential in ED processes were presented in detail. Additionally, the measurement for fundamental properties of the ionic species like concentration, velocity and the mole flux were introduced. The transport of ionic species in ED processes are described by the species transport equations and the phenomenological relationship between mole flux and concentration of the species. They are given by two different equations: the Maxwell-Stefan and Nernst-Planck equations. The Maxwell-Stefan equation describes the interaction of ionic species in the mixture where as the Nernst-Planck equation is a simplification of the Maxwell-Stefan equation for infinitely dilute mixtures where species interact only with the solvent and not with other ionic species. Due to the high salinity level of sea water around 35%, the Maxwell-Stefan equation is used to model the transport of ions in the dilute and concentrate flow channels. Furthermore, since seawater is a nonideal liquid mixture, the thermodynamic factors are introduced in the Maxwell-Stefan equations to describe nonideal behavior. The Nernst-Planck equations are used to model the transport of ions in the membrane since only specific ions are transported through membranes. In both flow channels and membranes, the ions are driven mainly due to the applied external electric field. Therefore, the external driving force is introduced in both Maxwell-Stefan and Nernst-Planck equations. In addition to equations describing ionic transport, the incompressible Navier-Stokes equation with external body force is introduced to describe the mixture transport. Finally, this chapter was concluded with the introduction of the electric potential equation to model the interaction between the electric potential and the local concentration of ionic species in the flow channels.

# 3 Numerical approaches

In this chapter, the numerical approaches chosen to simulate different physical subsystems shown in Figure 2.1 are presented in detail. The lattice Boltzmann method (LBM) is chosen to simulate the fluid flows, multicomponent flows and electric potential in the spacer-filled flow channel. The LBM is chosen due to its advantage in ease to integrate complex geometries like spacers and its simple two-step stream-collide algorithm that has high performance on supercomputers [12]. In Section 3.1, the lattice Boltzmann equation (LBE) for fluid flows is presented in detail along with the LBM algorithm and the boundary conditions. The asymptotic analysis which recovers the incompressible Navier-Stokes equations in the limit of low Mach numbers from the semi-discrete Boltzmann equation under diffusive scaling $\delta t \propto \delta x^2$ is given in App. A.1.1. The semi-discrete Boltzmann equation is the Boltzmann equation discretized in velocity space but still continuous in space and time coordinates. The LBE is the fully discrete Boltzmann equation discretized also in space and time coordinates.

In Section 3.2, the multicomponent LBE for nonideal fluids to simulate the multicomponent flow in spacer-filled flow channels is presented. Its corresponding asymptotic analysis, which recovers the incompressible Navier-Stokes equations for mixture and the Maxwell-Stefan equations for species transport are detailed in App. A.1.2. For the transport process in the ion exchange membranes, the simplified black box model given in Section 3.3 is used. The black box model is implemented as source/sink boundary for the multicomponent LBM as described in Section 3.2.2.

To simulate the electrodynamics defined by the Maxwell equations, the discontinuous Galerkin (DG) scheme was deployed [103] at first. From preliminary simulations, it was found that it is difficult to couple electrodynamics with multicomponent LBM due to the very small time step size in electrodynamics. Thus, even the smallest movement of ions leads to a large electric field that leads to instability of the multicomponent LBM. Therefore, the lattice Boltzmann method for the electric potential described in Section 3.4 was deployed to solve the Poisson's equation Eq. 2.64. This equation is solved for steady state solution at every physical coupling time step in the entire ED stack. The asymptotic analysis re-

covering the Poisson's equation from the discrete Boltzmann equation for electric potential is given in App. A.1.3. Finally, this chapter is concluded with the parameterization Section 3.5 on how to convert various quantities from physical SI units to lattice units.

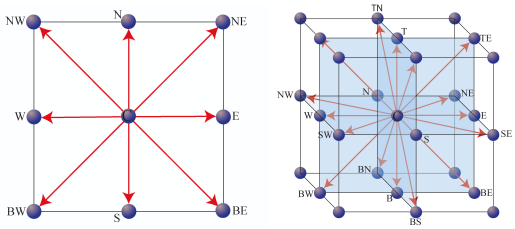
## 3.1 Lattice Boltzmann method for fluid flows

In this section, the lattice Boltzmann method (LBM) chosen to simulate hydrodynamics in spacer-filled channels is presented in detail. This method is an alternative to classical fluid dynamics methods like finite difference, finite volume, finite element, etc. that use fluid density, velocity and pressure as the primary variables on macroscopic level. LBM is a statistical method originated from the lattice gas automata, which is a discrete kinetics utilizing a discrete lattice and discrete time. LBM uses the lattice Boltzmann equation (LBE) that can be obtained from the Boltzmann equation with finite discrete velocities. In this work, the asymptotic analysis by Junk [41] is used to recover the incompressible Navier-Stokes equations from the discrete Boltzmann equation under diffusive scaling. The Boltzmann equation discretized with a finite set of discrete velocities $\boldsymbol{u}^m$ with an external body force term $F^m$ for $m \in \{1, \cdots Q\}$ is given as

$$\partial_t f^m(\boldsymbol{x}, t) + \boldsymbol{u}^m \cdot \nabla f^m(\boldsymbol{x}, t) = \Omega^m(\boldsymbol{x}, t) + F^m(\boldsymbol{x}, t) \tag{3.1}$$

where $f^m(\boldsymbol{x}, t)$ is the particle distribution function (PDF) at position $\boldsymbol{x}$ and at time $t$. This equation is referred to as the semi-discrete Boltzmann equation. The left side of this equation describes the transport of a particle and the right side $\Omega^m$ describes the local collision interaction between particles, and $F^m$ describes the body force. $Q$ is the number of lattice velocity directions that is dependent on the velocity model. The velocity model is also referred to as stencil. Here, in this work, the most commonly used stencils D2Q9 for 2 dimensions and D3Q19 for 3 dimensions are used (see Figure 3.1).

The fully discrete lattice Boltzmann equation (LBE) can be obtained by integrating the Eq. 3.1 along its characteristic velocities $\boldsymbol{u}^m$ and approximating the integral with the trapezoidal rule App. A.1.1.1 [35, 41]. In this work, the integrand of the body force term is approximated by forward Euler, i.e. first-order in time because of the algorithm used in our LB solver *Musubi*. In App. A.1.1.1, a new variable $\bar{f}$ was introduced which is defined by Eq. A.62 to convert the second-order implicit LBE to second-order explicit LBE. Thus, the LB algorithm is carried out in $\bar{f}$ instead of $f$. Thus, all equations presented in here, are in transformed




(a) D2Q9 model (b) D3Q19 model

**Figure 3.1** Commonly used velocity models in 2D and 3D.

variable $\bar{f}$ but for simplicity reasons the bar is neglected. The fully discrete LBE with the multiple relaxation time (MRT) [19, 41, 56] collision model with first-order body force term is given as

$$f^m(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t + \delta t) - f^m(\boldsymbol{x}, t) = \underbrace{\boldsymbol{A}(f^{eq,m}(\boldsymbol{x}, t) - f^m(\boldsymbol{x}, t))}_{=\Omega^m} + \delta_t F^m(\boldsymbol{x}, t). \tag{3.2}$$

Here, $\boldsymbol{A}$ is the linear collisional operator and $f^{eq,m}$ is the thermodynamic equilibrium distribution function in $m^{th}$ direction. For the isothermal incompressible case, $f^{eq,m}$ is given by the Maxwell-Boltzmann distribution function for low Mach numbers as

$$f^{eq,m}(\rho^0, \rho, \boldsymbol{v}) = \omega^m \left( \rho + \rho^0 \left( \frac{1}{c_s^2}(\boldsymbol{u}^m \cdot \boldsymbol{v}) + \frac{1}{2c_s^4}(\boldsymbol{u}^m \cdot \boldsymbol{v})^2 - \frac{1}{2c_s^2}(\boldsymbol{v} \cdot \boldsymbol{v}) \right) \right). \tag{3.3}$$

where $\rho$ and $\boldsymbol{v}$ are macroscopic density and velocity of the fluid respectively. $\rho^0 = 1$ is the constant mass density. $c_s$ is the speed of sound and it is set to $c_s = c/\sqrt{3}$ according to the isotropic condition given in Eq. 3.13 for $D2Q9$ and $D3Q19$ velocity models. $c = \delta x/\delta t$ is the lattice velocity, $\delta x$ is the lattice size and $\delta t$ is the lattice time. Here, the lattice cell size $\delta x$ and lattice time step size $\delta t$ are scaled to unity ($\delta x = \delta t = 1$) which results in the lattice velocity to be $c = 1$ representing that at every time step the PDF is streamed only to neighbor lattice. It's worth mentioning here that the subsequent descriptions are given in lattice units and conversion of relevant physical quantities from physical to lattice units are given in

Section 3.5. The macroscopic moments like the density $\rho$, the momentum $\mathbf{p} = \rho^0 \boldsymbol{v}$ and the total momentum flux tensor $\mathbf{\Pi}$ are related to the PDF $f^m$ by,

$$\rho = \sum_{m=1}^{Q} f^m, \tag{3.4}$$

$$\mathbf{p} = \rho^0 \boldsymbol{v} = \sum_{m=1}^{Q} \boldsymbol{u}^m f^m \implies \boldsymbol{v} = \frac{1}{\rho^0} \sum_{m=1}^{Q} \boldsymbol{u}^m f^m \tag{3.5}$$

and

$$\mathbf{\Pi} = \sum_{m=1}^{Q} \boldsymbol{u}^m \boldsymbol{u}^m f^m. \tag{3.6}$$

As shown in [21], the difference of the total momentum flux $\mathbf{\Pi}$ and the equilibrium momentum flux

$$\mathbf{\Pi}^0 = \sum_{m} \boldsymbol{u}^m \boldsymbol{u}^m f^{eq,m} = \rho^0 (c_s^2 \boldsymbol{I} + (\boldsymbol{v} \otimes \boldsymbol{v})) \tag{3.7}$$

gives rise to the deviatoric stress tensor $\boldsymbol{\sigma}'$, which contributes to the viscous term in the incompressible Navier-Stokes equations. As mentioned before, the LB algorithm is carried out in the transformed PDF $\bar{f}$ but the macroscopic moments must be computed from the original $f$. The conserved moments like density and momentum can be directly computed from $\bar{f}$, but the non-conserved moments like the deviatoric stress tensor $\boldsymbol{\sigma}'$ is calculated from

$$\boldsymbol{\sigma}' = -\left(1 - \frac{\lambda^\nu}{2}\right) \left(\mathbf{\Pi} - \mathbf{\Pi}^{(0)}\right). \tag{3.8}$$

The above equation can also be written in terms of the non-equilibrium PDF, $f^{neq,m} = f^m - f^{eq,m}$ as

$$\boldsymbol{\sigma}' = -\left(1 - \frac{\lambda^\nu}{2}\right) \sum_{m=1}^{Q} \boldsymbol{u}^m \boldsymbol{u}^m f^{neq,m}. \tag{3.9}$$

Thus, in LBM, the deviatoric stress tensor $\boldsymbol{\sigma}'$ can be computed locally using the non-equilibrium PDF $f^{neq,m}$, while other numerical methods require the gradient of velocity to compute this tensor.

The pressure $P$ is calculated from the density using the equation of state relation as

$$P = c_s^2 \rho = \frac{1}{3} \rho. \tag{3.10}$$

For the incompressible LBM model, the pressure $P$ is calculated by (see App. A.1.1)

$$P = c_s^2(\rho - \rho^0) = \frac{1}{3}(\rho - \rho^0). \tag{3.11}$$

Furthermore, the body force term $F^m$ is related to the macroscopic force $\boldsymbol{F} = \boldsymbol{g} + \frac{\rho^e}{\rho^0}\boldsymbol{E}$ as

$$F^m = \frac{\omega^m}{c_s^2}\boldsymbol{u}^m \cdot (\rho^0 \boldsymbol{F}). \tag{3.12}$$

The discrete lattice velocity vector $\boldsymbol{u}^m$ and weights $\omega^m$ are chosen to satisfy the following isotropy conditions

$$\langle 1, \omega \rangle = \sum_m \omega^m = 1 \tag{3.13a}$$

$$\langle 1, u_\alpha \omega \rangle = \sum_m u_\alpha^m \omega^m = 0 \tag{3.13b}$$

$$\langle 1, u_\alpha u_\beta \omega \rangle = \sum_m u_\alpha^m u_\beta^m \omega^m = c_s^2 \delta_{\alpha,\beta} \tag{3.13c}$$

$$\langle 1, u_\alpha u_\beta u_\gamma \omega \rangle = \sum_m u_\alpha^m u_\beta^m u_\gamma^m \omega^m = 0 \tag{3.13d}$$

$$\begin{aligned} \langle 1, u_\alpha u_\beta u_\gamma u_\delta \omega \rangle &= \sum_m u_\alpha^m u_\beta^m u_\gamma^m u_\delta^m \omega^m \\ &= \kappa c_s^4 \left(\delta_{\alpha,\beta}\delta_{\gamma,\delta} + \delta_{\alpha,\gamma}\delta_{\beta,\delta} + \delta_{\alpha,\delta}\delta_{\beta,\gamma}\right) \end{aligned} \tag{3.13e}$$

with $\kappa \neq D/(D+2)$ [41]. The upper restriction on $\kappa$ allows us to make use of the well-known D2Q9, D3Q15 or D3Q19 model where $\kappa = 1$. The Greek letters $\alpha, \beta, \gamma, \delta$ in the velocities subscripts represents dimension $1, 2, \ldots, D$. The discrete lattice velocity vector $\boldsymbol{u}^m$ and weights $\omega^m$ for $D2Q9$ are

$$\begin{aligned} \boldsymbol{u}^m &= (u_x^m, u_y^m) \\ &= \begin{cases} (-1,0), (0,-1), (1,0), (0,1) & \text{for } m = 1,2,3,4 \\ (-1,-1), (-1,1), (1,-1), (1,1) & \text{for } m = 5,6,7,8 \\ (0,0) & \text{for } m = 9 \end{cases} \end{aligned} \tag{3.14}$$

$$\omega^m = \begin{cases} 1/9 & \text{for } m = 1 \cdots 4 \\ 1/36 & \text{for } m = 5 \cdots 8 \\ 4/9 & \text{for } m = 9 \end{cases} \tag{3.15}$$

and for $D3Q19$, they are given in App. A.2.1. The linear collision operator $\boldsymbol{A}$ is defined as

$$\boldsymbol{A} = \boldsymbol{M}^{-1}\boldsymbol{\Lambda}\boldsymbol{M} \tag{3.16}$$

where $\boldsymbol{M}$ is the transformation matrix of size $Q \times Q$ and composed of a combination of the discrete lattice velocity vectors $\boldsymbol{u}^m$. The stability of the LBM can be increased if combinations of the discrete velocity vectors are chosen to be orthonormal [19, 56]. The choice of orthogonal vectors yield moments, which are linearly independent of each other. The transformation matrix $\boldsymbol{M}$ for $D2Q9$ model is chosen as

$$\boldsymbol{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 0 & -1 & -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 1 & -1 & 1 & -1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{3.17}$$

This matrix can be used to convert the particle distribution function vector $\boldsymbol{f}$ to the moments space vector $\boldsymbol{m}$ and vice versa using

$$\boldsymbol{m} = \boldsymbol{M}\boldsymbol{f}, \qquad \boldsymbol{f} = \boldsymbol{M}^{-1}\boldsymbol{m}. \tag{3.18}$$

The nine linearly independent moments for the $D2Q9$ model can be arranged in vector form as

$$\boldsymbol{m} = \begin{pmatrix} \rho & p_x & p_y & \Pi_{xx} & \Pi_{yy} & \Pi_{xy} & m_{xyy} & m_{xxy} & m_{xxyy} \end{pmatrix}^T. \tag{3.19}$$

The first three moments $\rho$, $p_x$, $p_y$ are locally conserved while the other six moments are not conserved. Multiplying the transformation matrix $\boldsymbol{M}$ with the equilibrium distribution function $f^{eq,m}$ Eq. 3.3 results in the equilibrium moments $\boldsymbol{m}^{eq}$ vector as

$$\begin{aligned} \boldsymbol{m}^{eq} = \Big( & \rho \quad \rho_0 v_x \quad \rho_0 v_y \quad P + \rho_0 v_x^2 \quad P + \rho_0 v_y^2 \quad \rho_0 v_x v_y \\ & c_s^2 \rho_0 v_y \quad c_s^2 \rho_0 v_x \quad c_s^2 \left( P + \rho_0 v_x^2 + \rho_0 v_y^2 \right) \Big)^T. \end{aligned} \tag{3.20}$$

$\boldsymbol{\Lambda} = diag(\lambda^m, m = 1, \cdots Q)$ is the non-negative diagonal collision matrix with relaxation parameter $\lambda^m$ for each moment $m_i$. For the $D2Q9$ model,

the diagonal entries are chosen as

$$\begin{aligned}
&\lambda^1 = \lambda^4 = \lambda^6 = \lambda^a, \\
&\lambda^2 = \lambda^\zeta, \\
&\lambda^3 = \lambda^b, \quad \lambda^5 = \lambda^7 = \lambda^c \\
&\lambda^8 = \lambda^9 = \lambda^\nu.
\end{aligned} \tag{3.21}$$

The relaxation parameters $\lambda^a$, $\lambda^b$ and $\lambda^c$ are tuned between $[0, 2]$ for stability reasons and in this work, they are set to $\lambda^a = 0$, $\lambda^b = 1.14$ and $\lambda^c = 1.92$. The relaxation parameter $\lambda^\nu$ is related to the kinematic viscosity $\nu$ as

$$\nu = c_s^2 \delta t \left( \frac{1}{\lambda^\nu} - \frac{1}{2} \right) \tag{3.22}$$

and the relaxation parameter $\lambda^\zeta$ is related to bulk viscosity $\zeta$ as [19]

$$\zeta = \frac{(5 - 9c_s^2)\delta t}{9} \left( \frac{1}{\lambda^\zeta} - \frac{1}{2} \right) \tag{3.23}$$

The one by two factor in the relation between kinematic viscosity and relaxation frequency comes from the second-order time discretization of the semi-discrete Boltzmann equation and is shown in App. A.1.1.1.

The transformation matrix $\boldsymbol{M}$, moments vector $\boldsymbol{m}$ and relaxation matrix $\boldsymbol{\Lambda}$ for the $D3Q19$ model are given in App. A.2.1. The MRT collision model reduces to the most commonly used single relaxation time Bhatnagar-Gross-Krook (BGK) [13] collision model when the relaxation parameter $\lambda^\nu$ is used for all entries in the diagonal collision matrix $\boldsymbol{\Lambda}$. Thus, the fully discrete LBE with the single relaxation time BGK collision with first-order force body term is

$$f^m(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t + \delta t) - f^m(\boldsymbol{x}, t) = \lambda^\nu (f^{eq,m} - f^m) + \delta_t F^m(\boldsymbol{x}, t). \tag{3.24}$$

The MRT model overcomes the limitations of the single relaxation time BGK model such as fixed Prandtl number ($Pr = 1$) and fixed ratio between kinematic and bulk viscosity [19]. Additionally, the MRT model is more stable than BGK since each moment can be relaxed differently to obtain stability.

In the asymptotic limit under diffusive scaling where the space and time step size are scaled as $\delta x = \epsilon$ and $\delta t = \epsilon^2$ [41], Eq. 3.1 recovers the incompressible Navier-Stokes equations (see App. A.1.1). The smallness parameter $\epsilon$ corresponds to the Knudsen number

$$Kn = l_m / L \tag{3.25}$$

with molecular mean free path $l_m$ and physical length scale $L$. The Mach number is given by

$$Ma = U/c \tag{3.26}$$

with the flow velocity $U$ in $\mathrm{m\,s^{-1}}$ and the speed of sound $c$ in $\mathrm{m\,s^{-1}}$.

### 3.1.1 Initial conditions

For time-dependent flow simulations, the dynamics of the flow are heavily dependent on the initial conditions. Therefore, they must be properly defined. Usually, the initial conditions are defined on macroscopic quantities like density and velocity, so they must be converted into PDF. The PDF can be split into an equilibrium and a non-equilibrium part as $f^m = f^{eq,m} + f^{neq,m}$. The equilibrium part is a function of macroscopic quantities: density $\rho$ and velocity $\boldsymbol{v}$ as in Eq. 3.3 and the non-equilibrium part is a function of shear stress tensor as [52]

$$f^{neq,m} = -\frac{\omega^m}{2c_s^4\left(1-\frac{\lambda_\nu}{2}\right)}\left(u^m u^m - c_s^2\boldsymbol{I}\right) : \boldsymbol{\sigma}'. \tag{3.27}$$

Since the shear stress tensor $\boldsymbol{\sigma}'$ is not straight forward to define at initial time, the PDFs are simply initialized with the equilibrium distribution function as

$$f^m(\boldsymbol{x}, t=0) = f^{eq,m}(\rho(\boldsymbol{x}), \boldsymbol{v}(\boldsymbol{x})). \tag{3.28}$$

This might result in some inconsistencies but they will be washed out after several numbers of time steps. The inconsistencies due to equilibrium based initial conditions can be observed in the flow simulation results with spacer geometry presented in Section 7.1.

### 3.1.2 Boundary conditions

For any numerical simulations, the boundary condition (BC) play an important role in simulating the real world problems in science and engineering with bounded domains. Boundary conditions greatly affect the numerical accuracy and stability of the numerical scheme. Often it is difficult to define BC mathematically and it is even more difficult on the numerical implementation as it may require neighbors of neighbors. In many numerical schemes, the problem arises in setting BC for complex geometries. Fortunately, LBM is very efficient in handling complex geometries due to its link-based approach and its origin from kinetic theory. However, it is not straight forward to impose BC which are defined in macroscopic

variables like pressure ($P$) and velocity ($\boldsymbol{v}$) on the mesoscopic particle distribution function $f^m$.

In the last decade various BCs for the LBM were proposed and analyzed [11, 15, 27, 42–44, 55]; nowadays the majority of BCs appearing in standard incompressible computational fluid dynamics are available for LBM too. The methodology behind specific BC is different: The most simplest of all is the BC based on bounce back technique, which comes from kinetic theory. In this work, the bounce back technique is used to handle wall and inlet BCs. For outlet, the extrapolation BC is used. Here, BCs implemented in the lattice Boltzmann solver *Musubi* [32] for incompressible flows are presented in detail. In the two-step streaming collision implementation style, the post-collision PDF $f^{c,\bar{m}}(\boldsymbol{x}_b, t)$ from the virtual boundary node $\boldsymbol{x}_b$ to fluid node $\boldsymbol{x}_f$ needs to be updated before streaming. Here, $m$ denotes the direction, which points towards the boundary, i.e. the outgoing direction, and $\bar{m}$ is its counter direction or incoming direction such that $\boldsymbol{u}^m = -\boldsymbol{u}^{\bar{m}}$. If BCs are to be applied after the collision step then the BCs must be applied on the incoming direction $\bar{m}$ of the post-collision PDF at the boundary node $\boldsymbol{x}_b$. Therefore, the BCs presented here are applied on $f^{c,\bar{m}}(\boldsymbol{x}_b, t)$

**Wall** The bounce back rule from the kinetic theory is the most simplest BC in the LBM, in which particles are bounced back or reflected when they hit the solid (no-slip) wall. This leads to zero macroscopic velocity $\boldsymbol{v} = 0$ along the wall, assuming the wall is located at half-way between the fluid and boundary nodes. In the LBM, this rule is applied by simply copying the outgoing PDF $f^m$ to incoming PDF $f^{\bar{m}}$ at $\boldsymbol{x}_f$ after the collision step, i.e.

$$f^{c,\bar{m}}\left(\boldsymbol{x}_f, t\right) = f^{c,m}\left(\boldsymbol{x}_f, t\right). \tag{3.29}$$

Figure 3.2 illustrates this BC with blue color representing fluid nodes and grey color representing solid nodes. The left side of the figure shows the PDFs $f^m$ near the wall before the stream step and the right side shows those PDFs after the stream step. This rule was analyzed in various publications and second order velocity and first order pressure accuracy was confirmed theoretically and numerically [42].

The limitation of the simple bounce back rule is that it results in zero velocity at wall only if wall is located at half-way between the nodes. To overcome this limitation of the simple bounce back rule and to improve the accuracy for complex geometries, a concept of $q$-values was proposed by Bouzidi et al. [15]. The q-values $q^m \in [0; 1)$ are the normalized distance between the actual position of wall or physical boundary and the adjacent

(a) Before stream step

(b) After stream step

**Figure 3.2** Particle distribution function links on the cell near the obstacle.

barycenter of the fluid node as in Figure 3.3. Considering $\boldsymbol{x}_w$ to be the exact location of the wall or the physical boundary and $\boldsymbol{x}_f$ to be the location of the barycenter of the adjacent fluid node, such that $\boldsymbol{u}^m$ is the lattice discrete velocity which points towards the boundary, then $q^m$ is defined by

$$\boldsymbol{x}_w = \boldsymbol{x}_f + \delta x \, q^m \boldsymbol{u}^m. \tag{3.30}$$

Clearly, for $q^m = 1/2$ the wall is located exactly $\delta x/2$ away from the element center. The $q$-value bounce back rule is then given by

$$f^{c,\bar{m}}(\boldsymbol{x}_f, t) = \begin{cases} 2q^m f^{c,m}(\boldsymbol{x}_f, t) + (1 - 2q^m) f^{c,m}(\boldsymbol{x}_{ff}, t) & \text{if } q^m < 1/2 \\ \frac{1}{2q^m} f^{c,m}(\boldsymbol{x}_f, t) + \frac{2q^m - 1}{2q^m} f^{c,\bar{m}}(\boldsymbol{x}_f, t) & \text{if } q^m \geq 1/2 \end{cases}. \tag{3.31}$$

Figure 3.3 shows the setup of the q-value bounce back rule for $q^m > 1/2$.

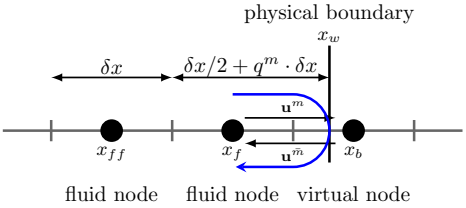



**Figure 3.3** Illustration of $q$-values in 1D. The black circles depict the element centers. The virtual node is behind the boundary and does not exist [103].

This BC improves the accuracy for arbitrary geometries significantly to first-order in pressure and second-order in velocity [42]. This BC is always applied to the spacer geometry in Chapter 7.

**Inlet** At inlet, the velocity bounce back rule is applied with macroscopic velocity $\boldsymbol{v}_{in}(\boldsymbol{x}_w, t)$ according to [44, 55].

$$f^{c,\bar{m}}\left(\boldsymbol{x}_f, t\right) = f^{c,m}\left(\boldsymbol{x}_f, t\right) + \frac{2}{c_s^2}\omega^m \rho^0 \left(\boldsymbol{u}^{\bar{m}} \cdot \boldsymbol{v}_{in}(\boldsymbol{x}_w, t)\right), \tag{3.32}$$

with the velocity index $\boldsymbol{u}^{\bar{m}} = -\boldsymbol{u}^m$ and $\boldsymbol{x}_w$ defines the exact position of wall given by Eq. 3.30. This BC is an extension of the simple bounce back rule Eq. 3.29 by applying velocity $\boldsymbol{v}_{in}$ to the wall. Therefore, this BC can also be used for the moving wall.

**Outlet** At outlet, a macroscopic pressure $P_o$ is imposed by outlet extrapolation boundary condition from Junk et al. [43] which is given as,

$$f^{c,\bar{m}}\left(\boldsymbol{x}_f, t\right) = \begin{cases} \left[f^{eq,\bar{m}}(c_s^{-2}P_o, \boldsymbol{v}_f) + f^{neq,c,m}\right](\boldsymbol{x}_b, t) & \text{if } \boldsymbol{u}^{\bar{m}} = -\boldsymbol{n} \\ 1.5f^{\bar{m}}(\boldsymbol{x}_b - \boldsymbol{n}, t+1) - 0.5f^{\bar{m}}(\boldsymbol{x}_b - 2\boldsymbol{n}, t+1) & \text{if } \boldsymbol{u}^{\bar{m}} \neq -\boldsymbol{n} \end{cases} \tag{3.33}$$

where $n$ is the outer normal direction of the boundary. $\boldsymbol{x}_b - \boldsymbol{n}$ and $\boldsymbol{x}_b - 2\boldsymbol{n}$ are two consecutive fluid nodes in the opposite direction of the boundary normal $\boldsymbol{n}$. The non-equilibrium distribution function $f^{neq,m}$ is given by

$$f^{neq,m} = f^m - f^{eq,m}. \tag{3.34}$$

$\boldsymbol{v}_e$ is the macroscopic velocity extrapolated from the adjacent fluid node of the boundary. In the macroscopic limit, this settings results in the general outflow condition of the form,

$$\frac{\partial \boldsymbol{v} \cdot \boldsymbol{n}}{\partial \boldsymbol{n}} = 0, \quad \frac{\partial \boldsymbol{v} \cdot \boldsymbol{t}}{\partial \boldsymbol{n}} = 0, \quad P = P_o \tag{3.35}$$

where $\boldsymbol{v} \cdot \boldsymbol{n}$ and $\boldsymbol{v} \cdot \boldsymbol{t}$ are macroscopic velocities in normal and tangential direction with respect to boundary. This technique is first-order accurate in pressure [43] and to obtain second-order accuracy in pressure, an alternative approach was proposed in [45].

## 3.2 Multicomponent lattice Boltzmann method

The single component LBM presented in the previous Section 3.1 is used to simulate pure hydrodynamics in the spacer-filled flow channels. To simulate the transport of ionic species in the diluted and concentrated flow compartments, the single component LBM is extended to the multicomponent LBM. The main reason for choosing multicomponent LBM in the

spacer-filled flow channel is its advantage to incorporate complex spacer geometries. In multicomponent, the LBE of the following form is solved for each species $k$,

$$\partial_t f_k^m(\boldsymbol{x}, t) + \boldsymbol{u}^m \cdot \nabla f_k^m(\boldsymbol{x}, t) = \Omega_k^m(\boldsymbol{x}, t) + d_k^m. \tag{3.36}$$

Here, $d_k^m$ is the external driving force for species $k$ in $m^{th}$ direction. The interaction between different species is defined by the collision operator $\Omega_k^m$. Several collision operators have been proposed for multicomponent lattice Boltzmann models [3, 5, 6, 26, 33, 40, 66, 86]. Some of those approaches are shown in Figure 3.4

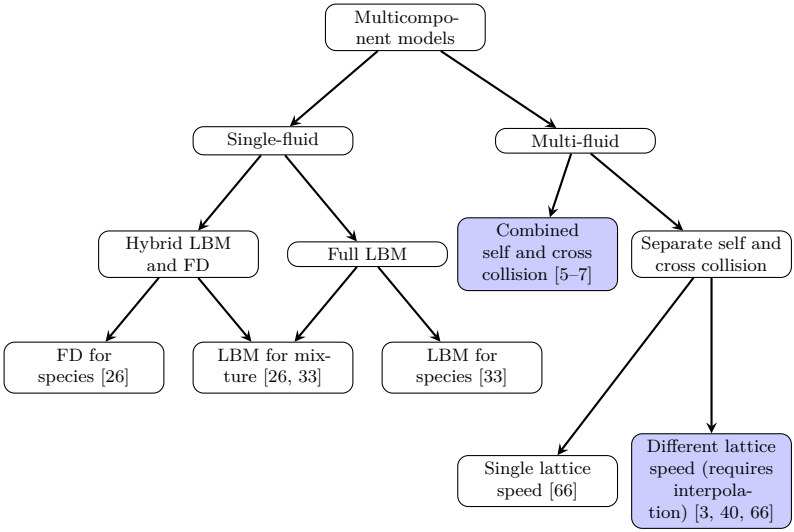



**Figure 3.4** Multicomponent simulation approaches

These approaches can be broadly categorized into single-fluid and multi-fluid model. In single-fluid models, the mixture flow is modeled by single component LBM and ionic species are modeled either by passive-scalar LBM with reduced lattice discrete velocity directions [33] or by conventional numerical schemes like finite differences [26]. Limitation of these approaches is that they solve the Nernst-Planck equations for ionic species in infinitely dilute solutions. Whereas in the multi-fluid model, the LBE is defined for each species and the collision operator defines the interaction between the species. Some authors [3, 40, 66] separated the collision operator into self and cross collision such that each collision term for each species is relaxed

towards its equilibrium according to its relaxation parameter as

$$\begin{aligned}\Omega_k^m &= \Omega_{k,k}^m + \sum_{l,l\neq k} \Omega_{k,l}^m \\ &= \lambda_k^{\nu}(f_k^{eq,m} - f_k^m) \quad + \sum_{l,l\neq k} \lambda_{k,l}^{\mathcal{D}} \Big(\frac{\rho_l}{\rho}\Big) \frac{f_k^{eq,m}}{c_s^2 \delta t} (\boldsymbol{u}^m - \boldsymbol{v}) \cdot (\boldsymbol{v}_l - \boldsymbol{v}_k).\end{aligned}$$

Here, $\lambda_k^{\nu}$ and $\lambda_k^{\mathcal{D}}$ are relaxation parameters related to kinematic viscosity $\nu$ and Maxwell-Stefan diffusion coefficient $\mathcal{D}_{k,l}$ respectively as

$$\nu_k = c_s^2 \Big(\frac{1}{\lambda_k^{\nu}} - \frac{1}{2}\Big)\delta t \tag{3.37a}$$

$$\mathcal{D}_{k,l} = \frac{\rho p}{c_t^2 M_k M_l} \Big(\frac{1}{\lambda_{k,l}^{D}} - \frac{1}{2}\Big)\delta t. \tag{3.37b}$$

In one of those collision models, [66] proposed a single lattice speed (SLS) and a different lattice speed (DLS) [3, 40, 66] approach. In DLS, each species is streamed with a different lattice speed of sound depending on its molecular weights. Thus, in the streaming step of LBM, the species with the least molecular weight is streamed to the neighbor lattice and the other species are streamed to the off-lattice within the lattice grid and an interpolation is used to obtain the species PDF at the lattice nodes. The lattice speed of sound for species $k$ is tuned as

$$c_{s,k} = \frac{1}{\sqrt{3}} \sqrt{\frac{M_k}{min_l(M_l)}}. \tag{3.38}$$

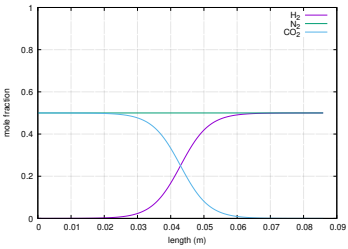
In SLS, the change in the speed of sound for each species is incorporated as some constant in the equilibrium function. The disadvantage of the DLS approach is that it requires interpolation, which is computationally expensive. Nevertheless, both of these approaches recover the Maxwell-diffusion equations and the incompressible Navier-Stokes equations in continuum limit only within the macroscopic mixture averaged velocity in the equilibrium distribution function of all species.

An alternative approach was proposed by Asinari [7], with a single collision operator and the interaction between species defined by a specially tuned velocity term in the equilibrium distribution function that results in the Maxwell-Stefan diffusion equation in continuum limit. A theoretical basis of this approach is based upon a BGK like kinetic model for gas mixtures as proposed by Andries et al. [2]. In [5], Asinari presented explicit and implicit LBE which are derived by integrating the discrete Boltzmann

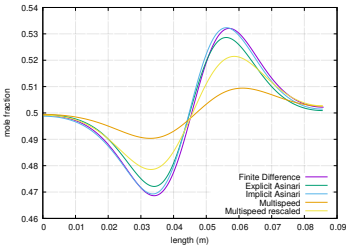
equation by the first-order forward Euler method and the second-order Crank-Nicolson method respectively. He mentioned that the explicit LBE suffers from lack of mass conservation so only the implicit LBE must be used. Similar to single component LBM, the second-order implicit LBE is converted into second-order explicit LBE using the variable transformation technique. The disadvantage of this technique in the multicomponent LBM case is that the linear equation system must be solved to obtain species velocity of the original PDF on each lattice. It is worth mentioning that all above mentioned approaches are proposed especially for ideal gas mixtures. Therefore, a new model for nonideal liquid mixture was developed [38] which is presented in Section 3.2.1. This new model is implemented in the in-house LBM solver *Musubi* which is part of this work. The robustness of the new model is analyzed on several test cases including complex geometry setup and they are published in [104].

To decide which of the different multicomponent models to use, the DLS (without and with scaled molecular weight ratio) and the Asinari (explicit and implicit LBE) approaches highlighted in Figure 3.4 were implemented in an experimental 2D LBM code and results are compared to the finite difference results obtained from gProms [77]. For this experiment, ternary gas mixture with $H_2$, $N_2$ and $CO_2$ in a tube of length $L = 0.0859$ m were considered. The Maxwell-Stefan diffusivity coefficients are $\mathcal{D}_{H_2,N_2} = 8.33 \times 10^{-5}\ \mathrm{m^2\,s^{-1}}$, $\mathcal{D}_{H_2,CO_2} = 6.8 \times 10^{-5}\ \mathrm{m^2\,s^{-1}}$ and $\mathcal{D}_{N_2,CO_2} = 1.68 \times 10^{-5}\ \mathrm{m^2\,s^{-1}}$. The initial mole fraction profile of each species is shown in Figure 3.5a. At $x = 0$ and $x = L$, wall (no-flux) boundary condition is assumed for all species, i.e $\mathbf{v}_k = 0$.

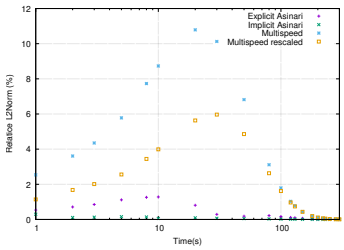
Figure 3.5b shows the mole fraction $\chi_k$ profile of $N_2$ at $t = 1$ s for different multicomponent models: Asinari explicit time stepping [5], Asinari implicit time stepping [5], DLS without scaled molecular weight ratio (multspeed) [40], DLS with scaled molecular weight ratio (multispeed rescaled) [40] and finite difference results from gProms. The figure shows that the result of the Asinari model with implicit time stepping is closer to FD than all other models. The multispeed rescaled shows increased accuracy of result but still is far from FD. The analytical solution is not available for this test case so the relative L2 error for all models is computed w.r.t FD assuming FD result is accurate. Figure 3.5c shows the relative L2 error of all models over time for species $N_2$. The explicit Asinari model is below 2% and the implicit Asinari model shows an error below 0.5% which is the best solution. Whereas, the errors of DLS with and without rescaled molecular weight are much higher. All models reach steady state around the same time and the error reduces towards steady state. From this, it can be concluded that the Asinari approach is better than the multispeed




(a) $t = 0$




(b) Mole fraction of $N_2$ at $t = 1$ s




(c) Relative L2 error

**Figure 3.5** Comparison of different multicomponent approaches on diffusion of ternary gas mixture consists of $H_2, N_2, CO_2$

(DLS) approach. Therefore, the Asinari model was chosen and extended for nonideal liquid mixtures to simulate the ED system [38, 104].

### 3.2.1 Nonideal liquid mixtures

In this section, the multicomponent LBM for nonideal liquid mixtures is presented which was developed to simulate the transport of ionic species and mixture in spacer-filled flow channels. To model nonideal liquid mixtures, thermodynamic factors which are essential to describe electrolytes are included in the Maxwell-Stefan relation. Furthermore, the electric force which drives ionic species towards the membranes is included in the Maxwell-Stefan equation. The boundary conditions for the multicomponent are presented in Section 3.2.2. Similar to the single component MRT Boltzmann equation for fluid flows, the discretized MRT Boltzmann

equation with the external driving force for each species [6] is written as

$$\partial_t f_k^m(\boldsymbol{x},t) + \boldsymbol{u}^m \cdot \nabla f_k^m(\boldsymbol{x},t) = \underbrace{\boldsymbol{A}_k \left(f_k^{eq,m}(\boldsymbol{x},t) - f_k^m(\boldsymbol{x},t)\right)}_{=:\Omega_k^m(\boldsymbol{x},t)} + d_k^m(\boldsymbol{x},t), \tag{3.39}$$

where $\boldsymbol{A}_k = \boldsymbol{M}^{-1}\boldsymbol{\Lambda}_k\boldsymbol{M}$ is the linear collision operator with relaxation matrix $\boldsymbol{\Lambda}_k$ defined for each species $k$. $f_k^m$ and $d_k^m$ are the PDF and the external driving force for species $k$ in discrete velocity direction $m$. The single component equilibrium function given in Eq. 3.3 violates the in-differentiability principle for multicomponents. I.e. for species with similar molecular weight Eq. 3.39 should reduce to single component LBE equation but it does not. The solution to this problem is proposed in [2] by modifying the velocity in the equilibrium function. The modified species equilibrium velocity $\mathbf{v}_k^{eq}$ for nonideal liquid mixture is given as

$$\mathbf{v}_k^{eq} = \mathbf{v}_k + \frac{1}{\rho_k}\sum_l \Gamma_{k,l}^{-1}\rho_l\phi_l \sum_\zeta \chi_\zeta \frac{\mathcal{D}}{\mathcal{D}_{\zeta,l}}\left(\boldsymbol{v}_\zeta - \boldsymbol{v}_l\right). \tag{3.40}$$

Here, $\mathcal{D}_{k,l}$ are the Maxwell-Stefan binary diffusive coefficients between species $k$ and $l$ and $\Gamma_{k,l}^{-1}$ are the inverse of thermodynamic factors. For ideal mixture $\boldsymbol{\Gamma}^{-1} = \boldsymbol{I}$, and the species equilibrium velocity Eq. 3.40 reduces to

$$\mathbf{v}_k^{eq} = \mathbf{v}_k + \phi_k \sum_l \chi_l \frac{\mathcal{D}}{\mathcal{D}_{k,l}}\left(\boldsymbol{v}_l - \mathbf{v}_k\right). \tag{3.41}$$

The thermodynamic equilibrium function in Eq. 3.39 for species $k$ is defined as

$$f_k^{eq,m}(\rho_k, \mathbf{v}_k^{eq}, \boldsymbol{v}) = \omega^m\rho_k\Big(s_m^k + \frac{1}{c_s^2}\left(\boldsymbol{u}^m \cdot \mathbf{v}_k^{eq}\right) + \frac{1}{2c_s^4}\left(\boldsymbol{u}^m \cdot \boldsymbol{v}\right)^2 - \frac{1}{2c_s^2}\left(\boldsymbol{v}\right)^2\Big). \tag{3.42}$$

Notice, that this is essentially the discrete version of the standard Maxwell equilibrium distribution Eq. 3.3, except for the non-standard density and velocity variables. In Eq. 3.42, the modified species velocity $\mathbf{v}_k^{eq}$ in the bilinear part recovers the Maxwell-Stefan equations and the mixture velocity $\boldsymbol{v}$ in the quadratic part recovers the nonlinear term in the momentum equation of the incompressible Navier-Stokes equations under asymptotic analysis, see App. A.1.2. The effect of using the mixture velocity $\boldsymbol{v}$ instead of the species velocity $\mathbf{v}_k^{eq}$ in the quadratic part of $f_k^{eq,m}$ is investigated in

App. A.1.2.5. In equilibrium function $f_k^{eq,m}$, the density is weighted by the parameter $s_m^k$ to incorporate species with different molecular weight as

$$s_m^k = \begin{cases} \frac{1}{\omega_0} + (1 - \frac{1}{\omega_0})\phi_k, & \text{if } m = 0 \\ \phi_k & \text{else} \end{cases}, \tag{3.43}$$

where the parameter $\phi_k$ is related to the molecular weight of species $M_k$ as

$$\phi_k = \frac{\min_l M_l}{M_k} \leq 1 \tag{3.44}$$

and the equation of state for each species is defined as

$$P_k = c_s^2 \phi_k \rho_k. \tag{3.45}$$

Thus, an additional weighting parameter $\phi_k$ allows us to define the equation of state by properly adjusting the effective local speed of sound for each species $c_{s,k} = \sqrt{c_s^2 \phi_k}$. In general, any equation of state is possible in the presented model but the following restrictions have to be considered:

- The equation of state should depend on slow mass diffusion time scales and its corresponding quantities like $\rho_k^0$ (see App. A.1.2).
- The parameter $\phi_k$ should be positive and $\leq 1$ to guarantee stability of the model.

The weights $\omega^m$ are chosen to guarantee the isotropy conditions given in Eq. 3.13 which are the same as in the single component case. For $D2Q9$, the relaxation matrix $\mathbf{\Lambda}_k$ [7] is set to

$$\mathbf{\Lambda}_k = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_k^{\mathcal{D}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_k^{\mathcal{D}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\lambda_k^{\zeta} + \lambda_k^{\nu}}{2} & \frac{\lambda_k^{\zeta} - \lambda_k^{\nu}}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\lambda_k^{\zeta} - \lambda_k^{\nu}}{2} & \frac{\lambda_k^{\zeta} + \lambda_k^{\nu}}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_k^{\nu} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{hom} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{hom} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{hom} \end{pmatrix}. \tag{3.46}$$

The entries in the relaxation matrix are related to the macroscopic quantities by

$$\lambda_k^{\mathcal{D}} = \frac{K}{\rho \mathcal{D}}, \tag{3.47a}$$

$$\lambda_k^{\nu} = \frac{1}{3\nu}, \tag{3.47b}$$

$$\lambda_k^{\zeta} = \frac{2 - \phi_k}{3\zeta}. \tag{3.47c}$$

Where $K$ is the bulk modulus of the liquid mixture, which defines fluid resistance to uniform compression. $\lambda_{hom}$ is the relaxation for higher order moments and if not specified otherwise it is set to 1.0 to maintain stability. Numerical values for some electrolytes relevant in electrodialysis can be found for example in [67]. The parameter $\mathcal{D}$ is free to be chosen for stability. If not mentioned otherwise, $\mathcal{D}$ is chosen such that $\lambda_k^{\mathcal{D}} = 2$. For liquids and gas phases the bulk modulus defines the speed of sound by the relation

$$c_s^2 = \frac{K}{\rho}. \tag{3.48}$$

$\nu$ and $\zeta$ are the kinematic viscosity and bulk viscosity of the mixture respectively. The factor $1/3$ in $\lambda_k^{\nu}$ and $\lambda_k^{\zeta}$ corresponds to the lattice speed of sound. Note that the relaxation parameters $\lambda_k^{\mathcal{D}}$ and $\lambda_k^{\nu}$ are species independent, but this restriction does not lead to a single independent diffusion parameter. Instead, the momentum exchange among the species is modeled by $\mathbf{v}_k^{eq}$ in $f_k^{eq,m}$ in Eq. 3.42. The relaxation matrix for $D3Q19$ is given in App. A.2.1. Finally, the equilibrium moments for species $k$, the $\boldsymbol{m}_k^{eq}$ vector can be obtained by multiplying the transformation matrix $\boldsymbol{M}$ (Eq. 3.17) with the equilibrium distribution function $f_k^{eq,m}$ Eq. 3.42 results in

$$\begin{aligned}\boldsymbol{m}_k^{eq} = \Big( & \rho_k \quad \rho_k v_{x,k}^{eq} \quad \rho_k v_{y,k}^{eq} \quad P_k + \rho_k v_x^2 \quad P_k + \rho_k v_y^2 \\ & \rho_k v_x v_y \quad c_s^2 \rho_k v_{y,k}^{eq} \quad c_s^2 \rho_k v_{x,k}^{eq} \quad c_s^2 \left( P_k + \rho_k v_x^2 + \rho_k v_y^2 \right) \Big)^T. \end{aligned} \tag{3.49}$$

The forcing term $d_k^m$ in Eq. 3.39 for nonideal liquid mixture is directly

related to the diffusive driving forces and mixture forces as

$$d_k^m = \underbrace{min_\alpha(M_\alpha)\omega^m \boldsymbol{u}^m \cdot \left(\sum_l \Gamma_{k,l}^{-1} c_t^0 \boldsymbol{F}_l\right)}_{=:d_k^{m,(1)}\text{diffusive driving force on species } k} + \underbrace{\frac{\omega^m}{c_s^2}\boldsymbol{u}^m \cdot \left(y_k \rho^0 \boldsymbol{F}\right)}_{=:d_k^{m,(3)}\text{fraction of mixture force on species } k} . \tag{3.50}$$

Here, $d_k^{m,(1)}$ is the diffusive driving force on species $k$ as described by the Maxwell-Stefan diffusion equation. $d_k^{m,(3)}$ is the fraction of mixture force on species $k$ as described by the momentum equation of the incompressible Navier-Stokes equations. The subscripts (1) and (3) defies the order of magnitude of the force term in their corresponding equations. They come from the asymptotic analysis, see App. A.1.2. $\boldsymbol{F}_k$ is the external diffusive drive force acting on species $k$ and it discriminates the nature of the component due to its charge as given by Eq. 2.20. $\boldsymbol{F}$ is the external body force of mixture acts on all the components in a same way as given in Eq. 2.56 i.e. $\boldsymbol{F} = \boldsymbol{g}$. The relation of the external forcing term with the incompressible Navier-Stokes equations is given in App. A.1.2 by matching terms of the right order in the asymptotic analysis. In case of ideal mixture, the forcing $d_k^m$ can be written as

$$d_k^m = \omega^m \boldsymbol{u}^m \cdot \left(c_t^0 min_\alpha(M_\alpha)\boldsymbol{F}_k + \frac{y_k}{c_s^2}\rho^0 \boldsymbol{F}\right). \tag{3.51}$$

Here, the macroscopic density and momentum of the species $k$ are obtained by

$$\rho_k = \sum_m f_k^m, \tag{3.52a}$$

$$\mathbf{p}_k = \rho_k \boldsymbol{v}_k = \sum_m \boldsymbol{u}^m f_k^m. \tag{3.52b}$$

Direct calculations show that the density moments of $f_k^m$ and $f_k^{eq,m}$ are equal

$$\rho_k = \sum_m f_k^m = \sum f_k^{eq,m}. \tag{3.53a}$$

This shows that the species density is conserved in collision i.e. is collision invariant. However, the species velocities $\mathbf{v}_k$ are not conserved in collision, i.e. the first order moments of $f_k^m$ and $f_k^{eq,m}$ do not equal necessarily. In general, the following relation holds

$$\begin{aligned}\mathbf{p}_k = \rho_k \mathbf{v}_k &= \sum_m \boldsymbol{u}^m f_k^m \\ &\neq \sum_m \boldsymbol{u}^m f_k^{eq,m} = \rho_k \mathbf{v}_k^{eq} \\ &= \rho_k \mathbf{v}_k + \sum_l \Gamma_{k,l}^{-1} \rho_l \phi_l \sum_\zeta \frac{\mathcal{D}}{\mathcal{D}_{l,\zeta}} \chi_\zeta \left(\boldsymbol{v}_\zeta - \boldsymbol{v}_l\right). \end{aligned} \tag{3.53b}$$

Furthermore, we notice that for $\boldsymbol{\Gamma} = \boldsymbol{I}$ and summing the above equation for all the components gives

$$\sum_k \rho_k \mathbf{v}_k^{eq} = \sum_k \rho_k \mathbf{v}_k = \mathbf{p}. \tag{3.54}$$

Therefore the total momentum is a collision invariant of our model. The effect of concentration dependent diffusion coefficients in nonideal Maxwell-Stefan formulation where $\boldsymbol{\Gamma} \neq \boldsymbol{I}$ is discussed in [38, 103].

The fully discrete LBE for species $k$ is obtained by integrating the continuous Boltzmann equation with MRT collision model (Eq. 3.39) along its characteristics $\boldsymbol{u}^m$. Similar to the single component case, the integrand of the collision term is approximated with trapezoidal rule and the diffusive forcing term is approximated by forward Euler, App. A.1.2.4. Dissimilar to the single component case, where relaxation parameters in the relaxation matrix $\boldsymbol{\Lambda}$ were transformed according to Eq. A.55, here in the multicomponent model, the relaxation parameters in $\boldsymbol{\Lambda}_k$ are unchanged. Thus, the fully discrete MRT-LBE for each species $k$ with first-order force term is written as

$$\begin{aligned}\bar{f}_k^m(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t + \delta t) - \bar{f}_k^m(\boldsymbol{x}, t) = \delta t \tilde{\boldsymbol{A}}_k \left(f_k^{eq,m}(\boldsymbol{x}, t) - \bar{f}_k^m(\boldsymbol{x}, t)\right) \\ + \delta t d_k^m(\boldsymbol{x}, t). \end{aligned} \tag{3.55}$$

Notice that the fully discrete relaxation matrix $\tilde{\boldsymbol{A}}_k = \boldsymbol{A}_k \left(\boldsymbol{I} + \frac{\delta t \boldsymbol{A}_k}{2}\right)^{-1}$ can be pre-computed at initialization. With this pre-computation, the cost for applying the relaxation matrix for one species is very much the same as for single component MRT-LBM. However, due to the introduction of the transformed PDF $\bar{f}_k$ and the collision invariance of species momentum, Eq. 3.53b, the linear equation system has to be solved to compute

the untransformed species momentum $\mathbf{p}_k$ from the transformed species momentum $\bar{\mathbf{p}}_k$ for every lattice cell

$$\bar{\mathbf{p}}_k = \mathbf{p}_k + \frac{\delta t \lambda_k^{\mathcal{D}}}{2} \sum_l \mathbf{p}_l \phi_l \sum_\zeta \Gamma^{-1}_{k,l} \frac{\mathcal{D}}{\mathcal{D}_{l,\zeta}} \chi_\zeta - \frac{\delta t \lambda_k^{\mathcal{D}}}{2} \sum_\zeta \mathbf{p}_\zeta \phi_\zeta \sum_l \Gamma^{-1}_{k,l} \frac{\mathcal{D}}{\mathcal{D}_{l,\zeta}} \chi_l. \tag{3.56}$$

For ideal mixtures, the above equation can be written as

$$\bar{\mathbf{p}}_k = \left(1 + \frac{\delta t \lambda_k^{\mathcal{D}}}{2} \phi_k \sum_l \frac{\mathcal{D}}{\mathcal{D}_{k,l}} \chi_l\right) \mathbf{p}_k - \frac{\delta t \lambda_k^{\mathcal{D}}}{2} \chi_k \sum_l \frac{\mathcal{D}}{\mathcal{D}_{k,l}} \phi_l \mathbf{p}_l. \tag{3.57}$$

### 3.2.2 Boundary conditions

In this section, boundary conditions for the multicomponent lattice Boltzmann method are presented. Similar to the single component case discussed in Section 3.1.2, BC for the multicomponent model are also given in terms of macroscopic moments of the target equation, e.g. $\rho_k, \boldsymbol{v}, \mathbf{v}_k, \cdots$. Thus, the number of macroscopic BCs are insufficient to determine the state in mesoscopic space, i.e $f_k^m$. In contrast to the single component case, there are a few difficulties in treating BC for multicomponent such as:

- Large number of different BC due to the complex nature of the macroscopic multicomponent equations.
- Each species requires different boundary condition along the same boundary.
- Non-standard diffusion-flux boundary for species.
- The multicomponent LBM Eq. 3.55 is formulated in transformed variable, therefore the moments which are defined for the boundary must be converted to transformed variable using Eq. A.123.

In addition to these difficulties, there are only a few proposed boundary conditions for multicomponent flows and most of them are analyzed only on binary mixtures. The only relevant BC proposed for multicomponent flows are moments based BC for binary mixtures proposed in [10]. This type of boundary conditions is robust, can be used for any specific macroscopic boundary condition, and is purely local. It can be applied even in presence of the variable transformation Eq. A.130. The key idea is to reduce the

PDF at the boundary to the linearly independent information and impose the remaining information via macroscopic moments. The main drawback of this approach is the handling of arbitrary geometries because it is not straight forward to compute moments for arbitrary unknown PDFs. An alternative to the moment based approach is the standard bounce back BC which works well for complex geometries. In contrast to the moment based boundary conditions, bounce back boundaries are implemented without additional boundary elements. This makes them more efficient with regards to memory and computational cost. Both these approaches are implemented in *Musubi* and in addition the most simple equilibrium boundary condition is also implemented for Dirichlet mole fraction and mole flux.

A single component bounce back BC like simple wall, higher order $q$-values and velocity bounce back presented in Section 3.1.2 can be defined directly on the transformed variable $\bar{f}_m^k$ in the multicomponent setting [103]. However, to simulate the ions transport in dilute and concentrate channels BC for certain other variables are required such as mole fraction and mole flux BC. The mole fraction BC is required to specify the species concentration at the inlet. The specified mole fraction is converted into species mass density using the relation

$$\rho_{k,in} = c_t \chi_{k,in} M_k \tag{3.58}$$

and the equilibrium function is used to update the incoming direction $\bar{m}$ of PDFs as

$$\bar{f}_k^{\bar{m}}(\boldsymbol{x}_b, t) = f_k^{eq,\bar{m}}(\rho_{k,in}, \mathbf{v}_k, \boldsymbol{v}). \tag{3.59}$$

Here, the species and mixture velocities ($\mathbf{v}_k$ and $\boldsymbol{v}$) are extrapolated from the neighbor lattice along the normal boundary.

In ED application, species concentrations and velocity are measured at both inlet and outlet. However, the channel length in the numerical simulation setup is smaller than the actual channel length so the species concentrations and velocity are defined at the inlet and they are extrapolated at the outlet. At the inlet, concentration and velocity of the species are defined as the mole flux of the species $\boldsymbol{N}_{k,in} = c_{k,in}\boldsymbol{v}_{in}$. The species mole flux $\boldsymbol{N}_{k,in}$ is converted into the species mass flux $\boldsymbol{n}_{k,in}$ using the relation $\boldsymbol{n}_{k,in} = \boldsymbol{N}_{k,in} M_k$ and the incoming direction $m^*$ of PDFs are updated using

$$\bar{f}_k^{c,\bar{m}}(\boldsymbol{x}_b, t) = \bar{f}_k^{m,c}(\boldsymbol{x}, t) + \frac{2}{c_s^2}\omega^m \left(\boldsymbol{u}^{\bar{m}} \cdot \boldsymbol{n}_{k,in}(\boldsymbol{x}_w, t)\right). \tag{3.60}$$

This BC is also used for source and sink BC to inject and remove ionic species respectively from spacer-filled flow channels. The source and sink boundaries are used to couple the flow channel with the membrane. In this thesis, the mole flux for source or sink boundary is computed from the membrane black-box model presented in Section 3.3.

## 3.3 Membrane black-box model

In ED stack, the membranes play a vital role due to theirs elective ions transport. As previously presented in Section 2.2.2, the Nernst-Planck equations are used to describe the ions transport through the membranes. Assuming electroneutrality and neglecting concentration gradient in membranes, selective ions are transported through the membranes only due to the applied external electrical field. In Section 2.2.3, the migrative flux $\boldsymbol{N}_k^{mig,mem}$ of species $k$ through membranes is given by Eq. 2.53 with $T_k^{mem}$ being the transport number of species $k$ in the membrane and $\boldsymbol{i}^{d,mem}$ being the current density through the membrane. Thus, if the transport number $T_k$ and the current density $\boldsymbol{i}^d$ are known then it's possible to define the migrative flux $\boldsymbol{N}_k$ as BC to the multicomponent LBM model. The positive and negative direction of the flux defines the source and sink boundaries respectively. Therefore, instead of solving the Nernst-Planck equations explicitly in the membranes, it can be treated as black-box by defining the transport number and flux direction as the boundary conditions.

In this section, the black-box model for the transport processes in the ion-exchange membranes is presented. It is based on a pure empirical description of the current-voltage characteristics of the membrane. Moreover, the current through the ion-exchange membranes is carried mostly by the counter ions or mobile ions. Therefore, the charge selective transport of multiple-ionic species is defined by concentration dependent transport numbers $T_k$. The transport processes are not spatially resolved with respect to the membrane thickness, instead they are modeled by the black-box model. If a dependence of the transport properties on the concentrations of the external electrolyte solutions is assumed, this leaves a spatial distribution only along the length and width of the membrane with the coordinates $\boldsymbol{x}^{mem} \in [\boldsymbol{x}_L^{mem}, \boldsymbol{x}_R^{mem}], l = 1, 2$, where $x_L^{mem}$ and $x_U^{mem}$ are fixed lower and upper bounds.

### 3.3.1 Assumptions and modeling equations

Lets recall the mole flux density $\boldsymbol{N}_k^{mem}$ of the ionic species through the membrane given in Eq. 2.53 as

$$\boldsymbol{N}_k^{mem}\left(\boldsymbol{x}^{mem},t\right)=\frac{T_k^{mem}\left(\boldsymbol{x}^{mem},t\right)}{z_k\mathcal{F}}\boldsymbol{i}^{d,mem}\left(\boldsymbol{x}^{mem},t\right),\qquad k=1,...,n_i, \tag{3.61}$$

where $n_i$ is the number of ionic species considered in the electrolyte solution. $T_k^{mem}$ is the transport number describing the fraction of electric current that is transported by a specific ionic species [72, 91–93] and $\boldsymbol{i}^{d,mem}$ is the current density transported through the membranes. The relation between current density and electric potential is given by the Ohm's law expression which is the most basic description of the current-voltage characteristics as

$$\psi^{l,mem}\left(\boldsymbol{x}^{mem},t\right)-\psi^{r,mem}\left(\boldsymbol{x}^{mem},t\right)=-R^{mem}\left(\boldsymbol{x}^{mem},t\right)\,\bar{i}^{d,mem}\left(\boldsymbol{x}^{mem},t\right). \tag{3.62}$$

Here, $\psi^{l,mem}$ and $\psi^{r,mem}$ are the electric potential at the left and right side of the membrane respectively, $R^{mem}$ is the concentration dependent specific surface membrane resistance and $\bar{i}^{d,mem}$ is the current density transported through the membrane along the normal direction of the membranes and it is given as

$$\bar{i}^{d,mem}\left(\boldsymbol{x}^{mem},t\right)=\boldsymbol{n}^{l}\cdot\boldsymbol{i}^{d,l}\left(\boldsymbol{x},t\right)=-\boldsymbol{n}^{r}\cdot\boldsymbol{i}^{d,r}\left(\boldsymbol{x},t\right). \tag{3.63}$$

In other words, $\bar{i}^{d,mem}$ corresponds to the orthogonal projection of the current density at the left $\boldsymbol{i}^{d,l}$ and right $\boldsymbol{i}^{d,r}$ side of the membrane. The normal vectors $\boldsymbol{n}^{l}$ and $\boldsymbol{n}^{r}$ point into the membrane surface. With this, the scalar molar fluxes of the ionic species through the membranes along its normal direction $\boldsymbol{n}$ can be expressed as

$$\bar{N}_k^{mem}\left(\boldsymbol{x}^{mem},t\right)=\frac{T_k^{mem}\left(\boldsymbol{x}^{mem},t\right)}{z_k\,\mathcal{F}}\,\bar{i}^{d,mem}\left(\boldsymbol{x}^{mem},t\right). \tag{3.64}$$

Furthermore, the empirically determined transport numbers have to satisfy the constraint

$$1=\sum_{j=1}^{n_i}T_k^{mem}\left(\boldsymbol{x}^{mem},t\right). \tag{3.65}$$

In case of multicomponent solutions, membrane specific correlations have to be derived, which shows a strong influence of the concentrations in the

external electrolyte solutions. The correlations for the transport numbers and the effective surface membrane resistance are derived from the Nernst-Planck equations in Section 2.2.3. Recall Eq. 2.54, the transport number of species $k$ through the membrane is given as [93]. This results in

$$T_k^{mem}\left(\boldsymbol{x}^{mem}, t\right) = \frac{z_k\, \mathcal{D}_k\left(\boldsymbol{x}^{mem}, t\right)\, \bar{c}_k^{mem}\left(\boldsymbol{x}^{mem}, t\right)}{\sum_{j=1}^{n_i} z_j^2\, \mathcal{D}_j\left(\boldsymbol{x}^{mem}, t\right)\, \bar{c}_j^{mem}\left(\boldsymbol{x}^{mem}, t\right)} \tag{3.66}$$

where $\mathcal{D}_k$ is the Nernst-Planck diffusion coefficient and $\bar{c}_k^{mem}$ are the averaged concentrations of the ionic species in the membrane phase computed as

$$\bar{c}_k^{mem} = \frac{1}{2}\left(c_k^{e,l} + c_k^{e,r}\right). \tag{3.67}$$

Here, $c_k^{e,l}$ and $c_k^{e,r}$ are concentration of ionic species in the electrolyte at left and right side of membrane respectively. The surface membrane resistance is given as

$$R^{mem}\left(\boldsymbol{x}^{mem}, t\right) = \frac{R\,T}{\delta^{mem} \sum_{j=1}^{n_i} z_j\, \mathcal{D}_j\left(\boldsymbol{x}^{mem}, t\right)\, \bar{c}_j^{mem}\left(\boldsymbol{x}^{mem}, t\right)}. \tag{3.68}$$

Here, $\delta^{mem}$ is the thickness of the membrane and $R$ is the universal gas constant.

Additionally, an empirical correlation for the water transport in the membrane is given by Fidaleo et al. [24] as

$$\begin{aligned}\bar{N}_{\mathsf{w}}^{mem}\left(\boldsymbol{x}^{mem}, t\right) = {} & \frac{T_{\mathsf{w}}}{\mathcal{F}}\, \bar{i}^{d,mem}\left(\boldsymbol{x}^{mem}, t\right) \\ & + LW \cdot \left(IS^{l}\left(\boldsymbol{x}^{mem}, t\right) - IS^{r}\left(\boldsymbol{x}^{mem}, t\right)\right).\end{aligned} \tag{3.69}$$

Here, $\bar{N}_{\mathsf{w}}^{mem}$ is the water flux density through the membrane, $T_{\mathsf{w}}$ is the water transport number, $LW$ is the osmotic water transport coefficient. $IS^l$ and $IS^r$ is the ionic strength of the electrolyte solution at the left and the right membrane side, respectively which is defined as

$$IS^{l}\left(\boldsymbol{x}^{mem}, t\right) = \frac{1}{2}\sum_{k=1}^{n_i} z_k^2\, c_k^{mem}\left(\boldsymbol{x}^{mem}, t\right). \tag{3.70}$$

**Model parameters** Lee et al. [57] and Fidaleo et al. [24] use a constant surface membrane resistance. The experimental values of the surface membrane resistance that have been determined for different membranes vary in the range of 0.3 to 5 $\Omega\mathrm{cm}^2$. In [24] experimentally determined values for the parameters characterizing the water transport through the

membrane are given as $LW = \frac{0.032}{3600}$ m s$^{-1}$ and $T_{\mathrm{w}} = 9.31$. Transport numbers $T_k^{mem}$ are usually measured in a single salt electrolyte. Fidaleo et al. [24] report estimated values for an sodium chloride solution as $T_{Na^+}^{cem} = 0.971$ and $T_{Cl^-}^{aem} = 0.998$.

## 3.4 Lattice Boltzmann method for the electric potential

The Poisson's equation Eq. 2.64 is an elliptical equation (i.e. time invariant) but the LBM intrinsically deals with parabolic (time dependent) equations. Therefore, to solve the Poisson's equation with LBM, it is necessary to introduce an artificial time-dependent term into Eq. 2.64 [62]

$$\partial_t \psi(\boldsymbol{x}, t) = \gamma \nabla^2 \psi(\boldsymbol{x}, t) + \gamma \frac{\rho^e(\boldsymbol{x}, t)}{\varepsilon}. \tag{3.71}$$

Here, $\rho^e$ is the charge density and $\varepsilon$ is the electric permittivity. $\gamma$ is a non-zero coefficient ($\gamma > 0$) represents the potential diffusivity and it's used to control the evolution speed of the transient electric potential equation Eq. 3.71. Note that the Poisson's equation Eq. 2.64 is the steady state solution of Eq. 3.71 for any $\gamma$. The value of $\gamma$ affects the numerical stability and also the computational cost. From some preliminary tests, the optimal value is found to be 0.167 and for this value a numerically accurate electric field can be obtained from Eq. 3.79. Since the transient electric potential equation Eq. 3.71 needs to be solved for steady state solution for every physical time step of the multicomponent flow equations. The simple and computationally cheaper, the BGK collision model with the single relaxation parameter $\bar{\lambda}^\gamma$ is used. The BGK Boltzmann equation for the electric potential discretized with a finite set of discrete velocities $\boldsymbol{u}^m$ with source term for Eq. 3.71 can be written as

$$\partial_t f^m(\boldsymbol{x}, t) + \boldsymbol{u}^m \cdot \nabla f^m(\boldsymbol{x}, t) = \lambda^\gamma (f^{eq,m}(\boldsymbol{x}, t) - f^m(\boldsymbol{x}, t)) + \mathcal{S}^m. \tag{3.72}$$

The Poisson's Eq. 2.64 is recovered from the above equation under asymptotic analysis (App. A.1.3). Here $f^m$ is the PDF of the electric potential and its equilibrium distribution function $f^{eq,m}$ is given as

$$f^{eq,m}(\boldsymbol{x}, t) = \omega^m \psi. \tag{3.73}$$

The source term $\mathcal{S}^m$ is related to the right hand side of the Poisson's Eq. 2.64

$$\mathcal{S}^m = \omega^m \gamma \frac{\rho^e(\boldsymbol{x}, t)}{\varepsilon_r \varepsilon_0}. \tag{3.74}$$

The fully discrete BGK LBM for Eq. 3.71 can be written as

$$\bar{f}^m(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t + \delta t) + \bar{f}^m(\boldsymbol{x}, t) = \bar{\lambda}^\gamma (f^{eq,m}(\boldsymbol{x}, t) - \bar{f}^m(\boldsymbol{x}, t)) + \delta t \mathcal{S}^m. \tag{3.75}$$

The relaxation time $\bar{\lambda}^\gamma$ is related to the potential diffusivity $\gamma$ by

$$\gamma = c_s^2 \delta t \left( \frac{1}{\bar{\lambda}^\gamma} - \frac{1}{2} \right). \tag{3.76}$$

With lattice speed of sound $c_s = 1/\sqrt{3}$ and lattice time step $\delta t = 1$, the above equation can be rewritten as

$$\gamma = \frac{1}{3}\left(\frac{1}{\bar{\lambda}^{\gamma}} - \frac{1}{2}\right). \tag{3.77}$$

The macroscopic electric potential $\psi$ is computed from the distribution function by

$$\psi = \sum_{m} \bar{f}^{m}. \tag{3.78}$$

The macroscopic electric field $\boldsymbol{E}$, which is the driving force of ions towards the membranes, can be computed locally using (App. A.1.4)

$$\begin{aligned} \boldsymbol{E} = -\nabla\psi &= \frac{\bar{\lambda}^{\gamma}}{c_s^2 \delta x \delta t} \sum_{m} \boldsymbol{u}^{m} f^{m} \\ &= 3\lambda^{\gamma} \sum_{m} \boldsymbol{u}^{m} f^{m}. \end{aligned} \tag{3.79}$$

### 3.4.1 Boundary conditions

Here, the unknown post-collision PDF $f^{c,\bar{m}}(\boldsymbol{x}_b, t)$ at a virtual boundary node $\boldsymbol{x}_b$ in incoming direction $\bar{m}$ is updated with the non-equilibrium extrapolation method [27, 61]. This method works for curved boundaries with q-values, which is essential to solve the electrical potential on the spacer geometry. Notations used here are represented in Figure 3.3. The unknown post-collision PDF at a virtual boundary node $\boldsymbol{x}_b$ can be expressed as the sum of equilibrium and non-equilibrium PDF at the virtual boundary node as

$$\bar{f}^{c,\bar{m}}(\boldsymbol{x}_b, t) = f^{eq,\bar{m}}(\boldsymbol{x}_b, t) + f^{neq,\bar{m}}(\boldsymbol{x}_b, t). \tag{3.80}$$

The equilibrium function $f^{eq,\bar{m}}(\boldsymbol{x}_b, t)$ can be calculated as

$$f^{eq,\bar{m}}(\boldsymbol{x}_b, t) = \omega^{\bar{m}} \psi_b \tag{3.81}$$

where $\psi_b$ is the electric potential at the virtual boundary node. For the Dirichlet condition, the electric potential $\psi_w$ is given at the exact physical boundary position $\boldsymbol{x}_w$. Thus, the electric potential at the virtual boundary node is obtained as [61]

$$\psi_b = \begin{cases} \frac{1}{q}[\psi_w + (q-1)\psi_f] & \text{if } q \geq 0.75 \\ [\psi_w + (q-1)\psi_f] + \frac{1-q}{1+q}[2\psi_w + (q-1)\psi_{ff}] & \text{if } q < 0.75 \end{cases}. \tag{3.82}$$

For the Neumann condition, which is required at the spacer geometry, the electric potential at virtual boundary node $\psi_b$ is obtained by simple second order extrapolation

$$\psi_b = (4\psi_f - \psi_{ff})/3. \tag{3.83}$$

Finally, the non-equilibrium PDF at virtual boundary node $f^{neq,\bar{m}}(\boldsymbol{x}_b, t)$ is obtained by the following second-order approximation [61]

$$f^{neq,\bar{m}}(\boldsymbol{x}_b, t) = \begin{cases} f^{\bar{m}}(\boldsymbol{x}_f, t) - f^{eq,\bar{m}}(\boldsymbol{x}_f, t) & \text{if } q \geq 0.75 \\ q[f^{\bar{m}}(\boldsymbol{x}_f, t) - f^{eq,\bar{m}}(\boldsymbol{x}_f, t)] & \\ +(1-q)[f^{\bar{m}}(\boldsymbol{x}_f f, t) - f^{eq,\bar{m}}(\boldsymbol{x}_{ff}, t)] & \text{if } q < 0.75 \end{cases}. \tag{3.84}$$

Note that the calculation of the non-equilibrium PDF at $\boldsymbol{x}_b$ is independent of the Dirichlet and Neumann conditions.

## 3.5 Parameterization

All the variables used in the LBM are in lattice units, they are parameterized using the following fundamental physical quantities in SI unit:

- Size of a fluid cell $\delta x$ [m]
- Time step $\delta t$ [s]
- Mass density $\rho$ [$\mathrm{kg\,m^{-3}}$]
- Mole density $c_t$ [$\mathrm{mol\,m^{-3}}$]
- Fundamental electric charge $q$ [$1.602\,176\,57 \times 10^{-19}$ C]
- Temperature $T$ [K].

To distinguish between the physical and the lattice variables asterisk $*$ has been added to the lattice variables in this section. In previous sections 3.1, 3.2, 3.4, the lattice velocity $c^* = \frac{\delta x^*}{\delta t^*} = 1$. i.e., the fluid particle can travel only one lattice cell per time step. So, the lattice cell size $\delta x^*$ and the lattice time step $\delta t^*$ are kept constant during the whole simulation.

$$\delta x^* = \frac{\delta x}{\delta x} = 1, \tag{3.85}$$

$$\delta t^* = \frac{\delta t}{\delta t} = 1. \tag{3.86}$$

Additionally, the mass density $\rho$, the mole density $c_t$, the charge $q$ and the temperature $T$ in lattice units are scaled to unity and kept constant during the whole simulation

$$\rho^* = \frac{\rho}{\rho}, \quad c_t^* = \frac{c_t}{c_t}, \quad q^* = \frac{q}{q}, \quad T^* = \frac{T}{T}. \tag{3.87}$$

Using the fundamental physical SI units other lattice variables can be calculated. The lattice viscosity $\nu^*$ and the lattice velocity $\boldsymbol{v}^*$ can be calculated from the physical kinematic viscosity $\nu$ and the physical velocity $\boldsymbol{v}$ using

$$\nu^* = \nu \frac{\delta t}{\delta x^2}, \tag{3.88}$$

$$\boldsymbol{v}^* = \boldsymbol{v} \frac{\delta t}{\delta x}. \tag{3.89}$$

The pressure $p$, force ($\boldsymbol{F}$), potential $\psi$, electric field ($\boldsymbol{E}$), mole flux ($\boldsymbol{J}$), diffusivity coefficient ($D_i$), charge density $\rho^e$, current $\delta A$, current density $\boldsymbol{i}^d$ and dielectric constant $\epsilon_0$ in lattice can be obtained as follows:

$$\begin{aligned} &p^* = p\frac{\delta t^2}{\rho \delta x^2}, \quad \boldsymbol{F}^* = \boldsymbol{F}\frac{\delta t^2}{\rho \delta x^4}, \quad \phi^* = \phi\frac{\delta t^2 C}{\rho \delta x^5}, \quad \boldsymbol{E}^* = \boldsymbol{E}\frac{\delta t^2 q}{\rho \delta x^4}, \\ &\boldsymbol{J}^* = \boldsymbol{J}\frac{\delta t}{c_t \delta x}, \quad D_i^* = D_i\frac{\delta t}{\delta x^2}, \quad \rho^{e,*} = \rho^e\frac{\delta x^3}{q}, \quad \delta A^* = \delta A\frac{\delta t}{q}, \\ &\boldsymbol{i}^{d,*} = \boldsymbol{i}^d\frac{\delta t \delta x^2}{q}, \quad \epsilon_0^* = \epsilon_0\frac{\rho \delta x^6}{q^2 \delta t^2}. \end{aligned} \tag{3.90}$$

The physical constants like Faraday $\mathcal{F} = 96\,485.3365\ \mathrm{C\,mol^{-1}}$and gas constant $\mathcal{R} = 8.314\,462\,1\ \mathrm{kg\,m^2\,s^{-2}\,mol^{-1}\,K^{-1}}$in lattice are given as

$$\mathcal{F}^* = \mathcal{F}\frac{c_t \delta x^3}{q} \tag{3.91}$$

$$\mathcal{R}^* = \mathcal{R}\frac{\delta t^2 c_t T}{\rho \delta x^2}. \tag{3.92}$$

## 3.6 Conclusion

In this chapter, the numerical scheme LBM to solve three physical equations was presented. The LBM for fluid flow to solve the incompressible Navier-Stokes equations in Section 3.1, LBM for multicomponent flow to solve the multicomponent transport equations in Section 3.2 and LBM for electric potential to solve the steady state potential equation in Section 3.4.

The fully discrete LBE equation with force/source term of each of those LBMs are only first-order in time because the second-order discretization of force/source term results in the modification of macroscopic velocity/potential with force/source term. The current implementation of the LBM solver *Musubi*(see Section 5.4.3) is not designed for such modification of macroscopic velocity/potential since the force/source is added to the LBE after the compute (stream-collision) kernel. Therefore, the implemented LBM scheme is only first-order when an external force is considered. However, it can be easily tackled by choosing small time steps. To increase numerical stability, the MRT collision model was introduced for both LBM for fluid flow and multicomponent flow. For the electric potential, the BGK collision model is sufficient. For all these LBMs, the PDFs are initialized with their equilibrium functions. Initializing with equilibrium functions causes numerical fluctuations and they remain in the domain for several time steps until they get washed out. This initial fluctuation can be minimized by slowly ramping the inflow velocity from rest. The asymptotic analysis of the discrete Boltzmann equation recovering each of the LBMs macroscopic physical equations under diffusive scaling is detailed in App. A.1.1, App. A.1.2, App. A.1.3.

Several ways to apply BCs in LBM were presented: bounce-back BC for fluid flow, equilibrium, bounce-back and moments based BC for multicomponent flow and non-equilibrium extrapolation BC for electric potential. Every treatment has its advantage and disadvantage. The advantages of bounce-back BCs are that they are simple, straightforward and easy to implement especially for no-slip wall boundaries. However, the numerical accuracy might be reduced for curved boundaries and this is resolved by using q-values to approximate the curved boundaries. The moments based BC is robust and accurate but difficult to define moments equation of unknown PDF for curved boundaries. The equilibrium BC are easy to implement by setting the unknown PDF to equilibrium function. It's stable but less accurate than bounce-back and moments BC. Finally, the non-equilibrium extrapolation BC is presented for LBM for the electric potential which is stable and accurate for curved boundaries using q-values.

The multicomponent LBM introduced in Section 3.2 was developed as part of this thesis to simulate the transport of ionic species in nonideal liquid mixtures. Different multicomponent models were investigated and the model proposed by Asinari was found to be accurate and more flexible to implement than others. Also, it can be used for mixtures with large molecular weights ratios between the species. In this model, the velocity in the equilibrium function was modified such that the Maxwell-Stefan equations are recovered in the asymptotic limit. Though this model has

its advantages, it was proposed for gas mixtures only so beforehand one of the objectives of this thesis is to extend it for nonideal liquid mixtures. During this extension, it was found that the Navier-Stokes equations for mixtures could be recovered only when the modified velocity is used in the linear part (second term) of the equilibrium function and the mixture velocity in the quadratic part (third and forth term) of the equilibrium function. The investigation on the effect of the choice of velocity in the equilibrium function is discussed in App. A.1.2.5. The first-order time discretization of the multicomponent Boltzmann equation suffers from lack of mass conservation of species and stability so the Boltzmann equation is discretized by second-order Crank-Nicolson scheme. The Crank-Nicolson discretization results in an implicit equation but can be turned into an explicit scheme by variable transformation. The density moments of the transformed equations are conserved. However, the velocity moments are not conserved due to the change in the velocity term in the equilibrium function. Thus, to obtain the actual species velocities, the linear equation system of size of number of species needs to be solved for every element. Even though this results in additional computations, it can be optimized for certain number of species to obtain good sustained performance in supercomputers through efficient utilization of processors.

In this work, the membranes are not explicitly solved by a numerical scheme. Instead the black-box model was proposed to model the membrane characteristics through pure empirical relations. The selective transport characteristic of the membrane is defined by the transport number $T_k$ of species through the selective ion exchange membrane. Thus, this model computes the mole flux $N_k$ of the species from the current density $i^d$ and the transport number $T_k$ of species. The mole flux $N_k$ of the species is treated by the multicomponent LBM as source/sink boundary condition. The direction of the flux defines source or sink. The incoming direction (pointing away from the membrane) is the source and outgoing direction (pointing towards the membrane) is the sink. In addition to ionic species transport through the membrane, the empirical relation for the water transport through the membrane was also introduced.

The LBE in general is an evolution equation but the Poisson's equation for the electric potential is a time-independent steady state equation. So, to solve this equation using the LBM scheme, an artificial time was introduced in the Poisson's equation. Therefore in a coupled simulation setup, the LBM for the electric potential must be solved until a steady state solution is reached at every time step of multicomponent flow. The steady state solution can be obtained faster by either choosing the spatial resolution lower than for the multicomponent flow or by decreasing the number

of lattice directions. Since it's better to have a high spatial resolution to maintain accuracy of the coupling simulation, the smaller stencil is preferred. Additionally, the artificial potential diffusivity, which is used to modify the relaxation parameter can also be tuned to reduce the number of iterations to reach steady state. From initial simulations, the potential diffusivity of 0.167 is found to be optimal so if not mentioned this is the default value used for artificial potential diffusivity.

At last, this chapter was concluded with the parameterization of physical variables, i.e. conversion of physical units to lattice units. Here, the basic SI units like meter (m), second (s), Kelvin (K), Coulomb (C), mole (mol) and kilogram (kg) are used to parameterize the element size ($\delta x$), time step size ($\delta t$), temperature, fundamental electric charge, mole density and mass density respectively.

# 4 Coupling of Multi-Physics Equations

In this chapter, the theoretical concept for coupling the numerical approaches which were presented in the last chapter to simulate different physical subsystems (Figure 2.1) is introduced. Furthermore, the physical conditions to be satisfied on the coupling interface and the variables to be exchanged between the subsystems are presented. Here, the term "domain" represents the individual physical subsystems and the term "interface" refers to the surface or the volume through which the individual physical subsystems interact with each other. The ED process contains three domains: 1. Flow channel with spacer, 2. Membrane and 3. Electrodynamics. The coupling strategy is derived from considering the physical interaction of these subsystems or domains. From ED layout in Figure 1.4, it is obvious that the flow channel with spacer interact with the membranes only via the surface area resulting in a surface coupling, where as both the flow channel and the membrane interact with electrodynamics on the entire volume resulting in a volume coupling. For both surface and volume coupling, the number of points on which variables need to be exchanged depends on the mesh resolution on either side of surface and volume. However, the volume coupling is more involved since more points in space are coupled.

At first, the general set-up of an ED process and its modules are introduced in Section 4.1. Then, the couplings of different numerical approaches are presented in detail. As mentioned in the previous chapter, the multicomponent LBM (Section 3.2) is deployed to simulate the transport of ionic species and the mixture. The species transport through membranes is modeled by the black-box model (Section 3.3) and the electrodynamics in the entire stack is solved by LBM for the electric potential (Section 3.4). In Section 4.2, the coupling of multicomponent LBM in spacer-filled flow channel with membrane black box model is presented. Additionally, the physical conditions to be satisfied on the membrane and electrolyte interface are discussed in detail. Finally, the coupling of the multicomponent LBM with the LBM for electric potential is presented in Section 4.3. It is worth mentioning that there is no need to couple the membrane black-box model with electrodynamics. However, if in the future the Nernst-Planck equations are to be considered for membranes then it can also be numerically solved using LBM [101]. In this case, the membrane

must be coupled with electrodynamics in a way similar to the coupling of the membrane with the flow channel.

## 4.1 General set-up for a model of an ED process and its modules

The structure of an ED batch process in laboratory is sketched in Figure 4.1. The processed electrolyte solution is initially fed to the respective storage tanks of the concentrate and dilute channels. Then, the electrolyte solution is pumped from the storage tanks to the electrodialysis module where the salt concentration in the concentrate channels increases while it decreases in the dilute channels. After the module, the solutions are recycled to the corresponding storage tanks until the desired product quality (e.g. an appropriate dilute concentration) is reached.

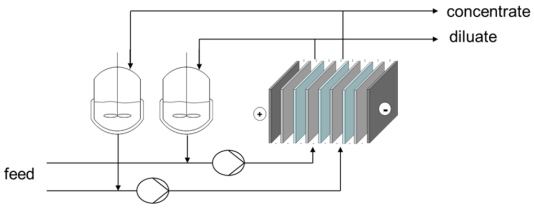
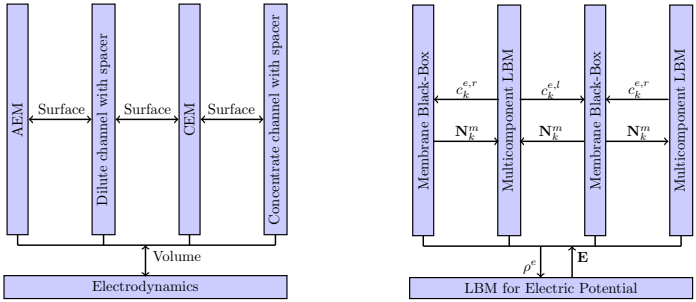



**Figure 4.1** Scheme of a typical ED batch process.

As seen in Figure 1.4, Anion- (AEM) and Cation-Exchange-Membranes (CEM) are arranged in an alternating manner in between a pair of electrodes. The spacer-filled flow channels between the membranes are rinsed with solution from the concentrate and dilute channels respectively. Thus, the repeating unit consists of one dilute and one concentrate channel with one of AEM, CEM. Figure 4.2a illustrates the coupling interaction between different physical subdomains in a repeating unit.

## 4.2 Coupling of multicomponent LBM and membrane model

The multicomponent LBM and the membrane transport model are coupled by introducing relations for the local concentrations and the total fluxes for all species in the system. In detail, the surface balance equations Eq. 4.2 are introduced as boundary conditions for the multicomponent




(a) Illustration of coupling domains

(b) Exchange od coupling Variables

**Figure 4.2** Left: Coupling between different physical systems. Right: Coupling variables exchanged between different numerical approaches chosen for each subsystems

LBM enforcing the total flux of all species at the boundary. In turn, the multicomponent LBM provides the local species concentrations $c_k^e$ as output at the boundary. From the concentration in the electrolyte solution, average concentrations in the membrane $\bar{c}_k^{mem}$ are determined using Eq. 3.67. With $\bar{c}_k^{mem}$, the black-box membrane model introduced in Section 3.3 calculates the molar fluxes $\boldsymbol{N}_k^{mem}$ which are required in the surface balance equations Eq. 4.2. The exchange of variables between the multicomponent LBM and the membrane model is illustrated in Figure 4.2b. The physical condition to be satisfied on the membrane-electrolyte interface is presented in the following section.

### 4.2.1 Membrane-electrolyte interface

Essential for the coupling of the IEM and flow-channels is the description of the resulting interface. The interface between an IEM and the electrolyte solution is abstracted as an infinitesimally thin phase, such that no storage of extensive quantities occurs. Furthermore, it is assumed that no chemical reactions take place in the interface. With these assumptions, general surface balance equations can be used to describe the transport processes across the interface. For the coupled system of nonideal mixtures, the activity coefficients $\gamma_k$ of the ions must be defined at the interface and they are obtained assuming local electro-chemical equilibrium at the membrane-electrolyte interface.

#### 4.2.1.1 Surface balance equations

The conservation laws for the extensive[1] quantities in a general thermodynamic phase hold in a relative manner at the interfaces in between the membrane and the electrolyte. Thus, surface balance equations are to be formulated expressing the conservation of mass and momentum. As isothermal conditions are considered, energy balance equations are neglected.

**Surface balance for the species mass** start with the conservation of the molar species fluxes across the interface and it can be is expressed in form of the surface balance equation as

$$\boldsymbol{n} \cdot (\boldsymbol{N}_k^e - \boldsymbol{N}_k^{mem}) = 0, \qquad k = 1, ..., n, \tag{4.1}$$

where $\boldsymbol{n}$ is a normal vector of the membrane surface. The superscripts $m$ and $e$ refers to membrane and electrolyte phase respectively. $\boldsymbol{N}_k^e = c_k^e \, \boldsymbol{v}_k = \boldsymbol{J}_k^e + c_k^e \, \boldsymbol{v}$ is the total molar flux in the electrolyte and $\boldsymbol{N}_k^{mem} = \boldsymbol{J}_k^{mem}$ is the total molar flux in the membrane phase. In case of the black-box membrane model, $\bar{J}_k^{mem}$ is a scalar in the normal direction of the membrane surface, such that Eq. 4.1 can be simplified to

$$\mathbf{n} \cdot \boldsymbol{N}_k^e - \bar{J}_k^{mem} = 0, \qquad k = 1, ..., n. \tag{4.2}$$

The surface balance equations Eq. 4.1 and Eq. 4.2 relates the normal components of the fluxes in the membrane and electrolyte solution, respectively. Thus, for a given flux $\bar{J}_k^{mem}$ entering or leaving the electrolyte channel, the normal components of the diffusive fluxes $\boldsymbol{J}_k^e, k = 1, ..., n$ are fixed.

Assuming that the mass transport along the tangential directions are negligible compared to the normal direction, the no-flux conditions are applied for flux tangential to the membrane surface, i.e.

$$\boldsymbol{t} \cdot \boldsymbol{N}_k^e = 0, \qquad k = 1, ..., n, \tag{4.3}$$

where $\boldsymbol{t}$ is the direction tangential to the membrane surface.

**Surface balance for momentum** defines the species balance equation for the conservation of the momentum on the interface as

$$\boldsymbol{n} \cdot (\boldsymbol{\Pi}^e - \boldsymbol{\Pi}^{mem}) + \bar{F} = 0, \tag{4.4}$$

[1] depends on the size of the system, E.g: mass, volume.

where $\mathbf{\Pi}^e$ and $\mathbf{\Pi}^{mem}$ are the total momentum fluxes in the electrolyte and membrane, respectively. $\bar{F}$ is the momentum source term corresponding to external electrical force in the normal direction of the membrane. In the electrolyte the total momentum flux is defined as

$$\mathbf{\Pi}^e = \rho \boldsymbol{v}^e \, \mathbf{v}^e + \boldsymbol{\sigma'}^e + \boldsymbol{I} \, P^e, \tag{4.5}$$

where $\boldsymbol{\sigma'}^e$ is the deviatoric viscous stress tensor and $\boldsymbol{I}$ is a diagonal matrix with unity entries. In membrane phase, it is not straightforward to define the total momentum flux tensor $\mathbf{\Pi}^{mem}$. Therefore, the following assumptions are made

- The viscous stresses tensor is neglected due to the very slow flow velocity in the membranes [70] and
- The convective acceleration terms are neglected because the density $\rho$ of the transported fluid is difficult to evaluate in the membranes.

Additionally, at mechanical equilibrium i.e., no velocity gradients, the following relation holds [96]

$$\frac{1}{\rho} \nabla P = \sum_{k=1}^{n} y_k \boldsymbol{F} \tag{4.6}$$

For example, consider the scenario of an ideal selective cation membrane, where the total flux that is transported across the membrane consists of sodium ions and possibly only a small fraction of water. In the simplest scenario, neglecting the viscous stress tensor and the convective acceleration term in the membrane phase, Eq. 4.4 simplifies to one relation for the normal component

$$\boldsymbol{n} \cdot (\rho \boldsymbol{v}^e \, \boldsymbol{v}^e + \boldsymbol{\sigma'}^e + \boldsymbol{I} \, (P^e - P^{mem})) = 0. \tag{4.7}$$

$P^e - P^{mem}$ represents the osmotic pressure difference between the electrolyte and the membrane, which is also called the swelling pressure of the membrane. For $NaCl$ solution i.e. monovalent electrolyte, the osmotic effects can be neglected [92]. Similar to the no-flux condition in tangential direction for the species mole flux, the mixture velocity in tangential direction is also set to zero i.e

$$\boldsymbol{t} \cdot \boldsymbol{v}^e = 0, \tag{4.8}$$

A more sophisticated approach would be for example using Beavers-Joseph conditions [9].

#### 4.2.1.2 Electro-Chemical equilibrium

Next to the interfacial surface balance equations, additional relations are required to fully describe the coupled system of nonideal mixtures. To relate the intensive[1] quantities in the membrane and electrolyte phase, the concept of a local thermodynamic equilibrium is used. Recall the diffusive driving force $\boldsymbol{d}_k$ (Eq. 2.19) introduced in Section 2.2 to describe the transport of an ion in the electrolyte and membrane phase which is equivalent to the gradient of the electro-chemical potential $\mu_k$. With that, the electro-chemical potential $\mu_k$ of species $k$ can be written as [92, 93]

$$\mu_k = \bar{\mu}_k + \tilde{V}_k \nabla P + z_k \mathcal{F} \psi \tag{4.9}$$

where $\bar{\mu}_k$ is the molar chemical potential of an ion which at constant temperature is a function of the total pressure $P$, $\tilde{V}_k$ is the specific molar volume. Eq. 2.22. $\nabla P = P - P^0$ is the pressure gradient, where $P^0$ is the standard atmospheric pressure. $\psi$ is the electric potential, $z_k$ is the specific charge number and $\mathcal{F}$ is the Faraday constant. The molar chemical potential $\bar{\mu}_k$ of species $k$ for nonideal fluids can be expressed as

$$\bar{\mu}_k = \bar{\mu}_k^0 + RT\ ln(\gamma_k) \tag{4.10}$$

where $\bar{\mu}_k^0$ is the standard chemical potential of species $k$ and $\gamma_k$ is the activity coefficient of species $k$ [96].

For the description of the behavior of the intensive quantities, e.g. molar concentrations $c_k$ or mole fractions $\chi_k$ in the two phases (membrane and electrolyte), additional relations needs to be specified. A common approach is to specify an equilibrium relation at the interface. Using Donnan equilibrium [93] (electro-chemical equilibrium) at membrane-electrolyte interface, electro-chemical potential $\bar{\mu}_k^e$ in electrolyte phase and electro-chemical potential $\bar{\mu}_k^m$ at the membrane phase are equal

$$\mu_k^{mem} = \mu_k^e, \qquad k = 1, ..., n, \tag{4.11}$$

then it follows

$$\implies \bar{\mu}_k^{mem} + \tilde{V}_k(P^{mem} - P^0) + z_k \mathcal{F} \psi^{mem} = \bar{\mu}_k^e + \tilde{V}_k(P^e - P^0) + z_k \mathcal{F} \psi^e \tag{4.12}$$

[1] does not depend on the size of the system, E.g: temperature, density.

Substituting Eq. 4.10 and rearranging the above equation gives the Donnan potential $\psi_{Don}$ as

$$\psi_{Don} = \psi^{mem} - \psi^{e} = \frac{1}{z_k \mathcal{F}} \left( RT\ ln \frac{\gamma_k^e}{\gamma_k^{mem}} + \tilde{V}_k (P^e - P^{mem}) \right) \quad (4.13)$$

which is the potential difference between membrane and electrolyte phase. The Donnan potential $\psi_{don}$ can be calculated either from cation or anion activity coefficient as

$$\begin{aligned} \psi_{Don} &= \frac{1}{z_k \mathcal{F}} \left( RT\ ln \frac{\gamma_c^e}{\gamma_c^{mem}} + \tilde{V}_c (P^e - P^{mem}) \right) \\ &= \frac{1}{z_k \mathcal{F}} \left( RT\ ln \frac{\gamma_a^e}{\gamma_a^{mem}} + \tilde{V}_a (P^e - P^{mem}) \right). \end{aligned} \quad (4.14)$$

The subscripts $c$ and $a$ refer to cation and anion respectively. If the black box membrane model introduced in Section 3.3 is used then the osmotic pressure term is neglected and the above equation can be expressed for each species $k$ as

$$\left( \frac{\gamma_k^e}{\gamma_k^{mem}} \right)^{\frac{1}{z_k}} = e^{\frac{\mathcal{F}}{RT}(\psi^{mem} - \psi^e)} \quad k = 1, .., n_i. \quad (4.15)$$

If the electric potential inside the membrane phase is not of particular interest then it is useful to reformulate the above equation such that the first ionic species is explicitly expressed as the exponential of the potential difference. Introducing this expression into the equation for the remaining species gives

$$\left( \frac{\gamma_k^e}{\gamma_k^{mem}} \right)^{\frac{1}{z_k}} = \left( \frac{\gamma_1^e}{\gamma_1^{mem}} \right)^{\frac{1}{z_1}}, \qquad k = 2, ..., n_i. \quad (4.16)$$

The model is closed by the electro-neutrality condition for the membrane phase

$$0 = \sum_{j=1}^{n_i} z_j\, \bar{c}_k^{mem} + \rho_{fix}, \quad (4.17)$$

where $\rho_{fix}$ is the non-zero space charge[1] of the fixed charged groups and $\bar{c}_k^{mem}$ is the average concentration of species $k$ in membrane phase given by Eq. 3.67.

[1] is a concept in which excess electric charge is treated as a continuous of charge distributed over a region of space

## 4.3 Coupling of multicomponent LBM and LBM for electric potential

In dilute and concentrate flow channels, the ions of selective charges are transported in opposite direction due to the applied electrical potential in the direction perpendicular to the main flow direction. Near the membranes exists a non-electroneutral regime (EDL) where the charge density $\rho^e$ becomes non-zero which results in change in electric potential. Thus, to incorporate this change in the electric potential on the ions transport, the multicomponent LBM (Section 3.2) must be coupled with the LBM for electric potential (Section 3.4).

As shown in Figure 4.2a, these two domains are interacting with each other in volume resulting in volume coupling. At every time step, the LBM for electric potential is solved for until the solution convergence i.e steady state solution is reached since the Poisson equation is time independent. From the steady state solution of the LBM for the electric potential, an electric field $\boldsymbol{E}$ is calculated using Eq. 3.79 and fed in as input to the multicomponent LBM model. In turn, the multicomponent LBM uses the electric field $\boldsymbol{E}$ as source term to transport the ions and calculates the charge density $\rho^e$ using Eq. 2.44 from the local concentration of ions $c_k$. The charge densities $\rho^e$ are then handed back to the LBM for the electric potential to close the loop of interaction. The variables are exchanged between the domains in physical units and they are locally converted into lattice units in the individual domains. Variables that are exchanged between these domains are illustrated in Figure 4.2b. The coupling tool *APESmate* presented in the next chapter enables exchanging arbitrary variables in arbitrary resolution at every time step. This allows each domain to have different mesh resolution or stencil layouts. Thus, the LBM for electric potential can be chosen to have reduced stencil layout compared to the multicomponent LBM to reduce the computational effort at every time step.

The coupling becomes more involved and complicated, if the full Maxwell equations Eq. 2.59 are to be solved for the electric field. In this case, there are a few things to be considered. At first, the divergence correction term [71] must be introduced to the Maxwell equations to clean divergence errors. Then, the efficient numerical scheme must be chosen to solve this equation. Within the *APES* framework, *Ateles* a Discontinuous Galerkin (DG) solver is used to solve these equations with explicit time stepping. In this case, the multicomponent LBM gives the charge density $\rho^e$ and the current density $\boldsymbol{i}^d$ as input to the Maxwell solver and in turn the

Maxwell solver returns the electric field $\boldsymbol{E}$. Note that if both schemes are solved with explicit time stepping and exchange variables at each time step, both schemes should have the same time step. Since electrodynamics travels with the speed of light, the time step for the Maxwell solver is much smaller than the multicomponent LBM. Additionally, it is costly to satisfy the Gauss law at all times since ions are transported with much large time scale than the electric field. This makes this coupling complicated and expensive. Therefore, to circumvent this problem, the magnetic fields are neglected which reduces the full Maxwell equations to the Poisson equation. Thus, in this thesis, only the Poisson equation for the electric potential is considered.

## 4.4 Conclusion

In this chapter, the coupling strategy employed for surface and volume coupling involved in the coupling of multi-physics in the ED process were detailed. The transport of ions in the flow channels and the membranes are coupled via the surface and the interaction between ions with electric potential in entire domain is coupled via the volume. In terms of numerical schemes, the membrane black-box model is coupled with multicomponent LBM in flow channels through BC. The multicomponent LBM provides concentration of species $c_k$ on the elements next to the boundary to the membrane black-box model. The black-box model computes mole flux $N_k$ from the concentration $c_k$ and transport number of species $T_k$ on the specific membrane provided in the configuration file. This mole flux $N_k$ is provided back to the multicomponent model where it is applied as a source/sink BC. Additionally, the surface balance equations which need to be satisfied on the surface coupling interface between channel and membrane were introduced.

Regarding the volume coupling, only multicomponent flows in flow channels are coupled with electric potential because membranes are modeled by black-box model. In the volume coupling, the multicomponent LBM computes charge density $\rho^e$ from local concentration of species $c_k$ and provides them as a source term to the LBM for the electric potential. In turn, the LBM for the electric potential computes the potential field by solving the potential equation until the steady state is reached and computes the electric field $\boldsymbol{E}$ which is the negative gradient of the electric potential. As mentioned in the previous chapter, the one advantage of using LBM for electric potential is that the electric field $\boldsymbol{E}$ can be computed directly from the PDF. This electric field $\boldsymbol{E}$ is provided as a source

term to the multicomponent LBM. Note that for every time step of the multicomponent LBM, the LBM for the electric potential must be solved until the solution reaches steady state.

# 5 Simulation framework

In this chapter, the highly scalable in-house simulation framework named *APES* (Adaptable Poly-Engineering Simulator) [80] is introduced. A brief overview of this framework is presented in Section 5.1. The central library of the framework called *TreElM* [48], which is based on an octree data structure is discussed in Section 5.2. In Section 5.3 the octree mesh generator called *Seeder* developed in this thesis to generate the computational mesh with spacer geometry is presented in detail with its algorithm. The Lattice Boltzmann solver *Musubi*[32] which is used to simulate flow dynamics, multicomponent flows and electric potential in flow channels with spacer is briefly presented in Section 5.4. Here, the performance of single component and multicomponent LBM on large-scale supercomputers is also presented. The integrated coupling tool named *APESmate* used to couple the individual *APES* solvers is the major contribution of this thesis and it is presented in detail with its algorithm in Section 5.5. Here, the important features like space-time function and variable system, which were required to realize the coupling between *APES* solvers, are also presented.

## 5.1 Apes overview

Most large-scale industrial applications require enormous computing resources and efficient numerical algorithms to utilize those resources. Thus, *APES* [80] was developed to satisfy this requirement using octree-mesh based highly scalable parallel algorithms. The entire framework was mainly written in Fortran 2003 and some C-codes are used as external libraries. The overview of *APES* with all its sub-projects is illustrated in the schematic layout Figure 5.1.

The framework revolves around central library the *TreElM* [48], which provides a tree based data structure which is efficient on modern large scale distributed memory systems [49]. The mesh generator *Seeder* [63] generates a computational domain and writes it to disk. The numerical solver *Musubi* [32], reads the mesh data from disk and converts it into local solver specific data structures. These data structures are set upon the octree data structure and are designed such that they are efficient

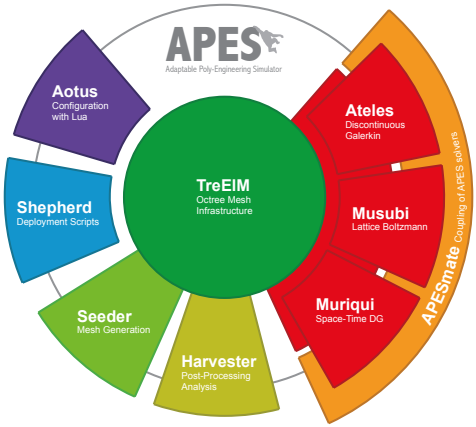



**Figure 5.1** Schematic layout of solver suite *APES*.

on their respective compute kernels per solver. The solver dumps solver state values (PDF) at specific times to disk as restart files. These files can be used to restart the simulation and also to post-process results using *Harvester*. Thus, the post-processing tool Harvester provides output in different visualization formats like *vtk* or *ascii*. The configuration files for all modules in *APES* are provided as Lua [37] scripts. *Aotus* [47] provides an interface to load variables or functions defined in Lua scripts into the Fortran application. It is worth mentioning that this framework also enables us to couple different numerical solvers easily since they have same octree data structure. In the following section, the octree data structure in *TreElM*, the mesh generator *Seeder*, the lattice Boltzmann solver *Musubi* and the integrated coupling tool *APESmate* are discussed. A python tool *Shepherd* is used to steer the *APES* workflow especially for parameter studies and to submit jobs in supercomputers. This tool is also used to run regression checks to the *APES* projects.

## 5.2 Octree data structure

In this section, the octree data structure of *TreElM* is briefly discussed to support the mesh generation *Seeder* discussion in the next section. Tree based meshes are regular structural meshes created by recursive subdivision of a $d-$dimensional cube. In general, subdivision creates $2^d$ number of children and subdivision starts with the root cube which represents the mesh universe or bounding cube. The bounding cube

represents the root node of the tree and contains all its children nodes. The term *node* refers to an element or cell. To identify the node in the tree, we utilize a global breadth-first numbering scheme for all the elements of the full tree with a geometric ordering by a space-filling curve (SFC) [69]. *TreElM* supports only octree-mesh data structure i.e $d = 3$, thus a node is subdivided into 8 children. The children local coordinate following SFC are $(0,0,0),(1,0,0),(0,1,0),(1,1,0),(0,0,1),(1,0,1),(0,1,1),(1,1,1)$. The refinement level ($\mathfrak{L}$) of a node defines the number of subdivision performed on the bounding cube to obtain a node on that level. The *TreeID* ($\mathfrak{tID}$) is an unique number to identify every node in the tree. It implicitly encodes the spatial and topology representation of an element within the bounding cube, which allows for fast loop up of certain $\mathfrak{tID}$ in the array of $\mathfrak{tID}$s. The $\mathfrak{tID}$ is defined as 64-bit signed long integer and for octree meshes this limits the maximum level ($\mathfrak{L}_{globalmax}$) a node can be refined to be 20 i.e. 1 million nodes per direction. For the root node, both *TreeID* and level are set to zero, $\mathfrak{tID} = 0$ and $\mathfrak{L} = 0$. The parent $\mathfrak{tID}$ of its children for $d - dimensional$ is given as,

$$\mathfrak{tID}_{parent} = int\left[\frac{\mathfrak{tID}_{child} - 1}{2^d}\right]. \tag{5.1}$$

Likewise, the $\mathfrak{tID}$ of the child index $k$ of its parent can be obtained from,

$$\mathfrak{tID}_{child} = 2^d * \mathfrak{tID}_{parent} + k. \tag{5.2}$$

Additionally, the first $\mathfrak{tID}_{\mathfrak{L}}$ on any level can be computed easily just by knowing the first $\mathfrak{tID}_{\mathfrak{L}-1}$ on previous level

$$\mathfrak{tID}_{\mathfrak{L}} = 2^{d(\mathfrak{L}-1)} + \mathfrak{tID}_{\mathfrak{L}-1}. \tag{5.3}$$

An example of how a tree based mesh looks like is illustrated in Figure 5.2 together with a domain decomposition of the fluid $\mathfrak{tID}$ list on multiple process. An advantage of the unique $\mathfrak{tID}$ and SFC is that each process can compute its neighbors locally with minimal data from the remote partitions i.e only the first and last *TreeID* in each process are known to all the processors [48, 80]. Thus, the computational workload of this neighbor identification grows as O($log(nprocs)$) where *nprocs* is the number of processor involved in the computation. The computational domain is equally distributed on all processors, i.e. the number of fluid elements on each processor is almost equal except when the total number of elements is not exactly divisible by *nprocs*. In this case the last processor will have less number of elements. Thus, almost perfect load balancing can be achieved. However, in reality the computational cost of some elements

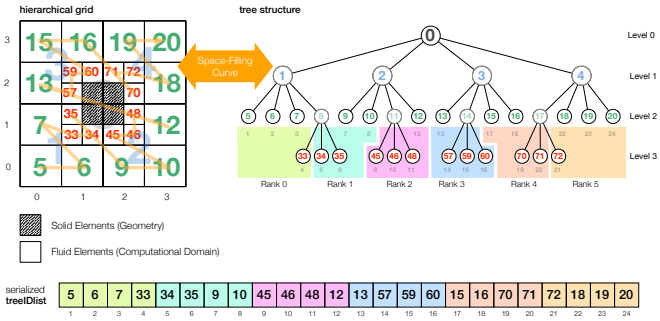



**Figure 5.2** Left: Quadtree (2D) mesh with an obstacle. Right: Level-wise tree representation with domain decomposition. Bottom: Serialized fluid treeID list dumped into disk

might differ due to boundary, source or multilevel interpolation. In such case, a dynamic load balancing algorithm is used [79] (see also Section 5.4.5).

## 5.3 Seeder - Mesh generator

The mesh generator *Seeder* generates an octree mesh from geometries provided in the form of STL files or predefined canonical shapes. The bounding cube, which defines the mesh universe is considered as periodic domain. As mentioned before, the octree can be generated by recursive subdivision of this cube. The approach used in an earlier *Seeder* version [30], recursively subdivides the bounding cube until a desired level specified for the node is reached. Then, if nodes at their maximum level ($\mathfrak{L}_{max}$) are intersected by a boundary geometry then they are marked as intersects boundary nodes else if they are intersected by a seed geometry then they are marked as fluid nodes. In this approach, all nodes are already at their user defined level ($\mathfrak{L}_{max}$) before flooding step. The term *flooding* refers to the marking of a node as fluid if that node is not intersected by a boundary geometry and one of its face neighbor nodes is fluid. In the flooding step, the fluid (computational) domain is identified by looping over all nodes and marking nodes as flooded which are non-intersected by boundary geometries and have a face neighbor that is a fluid. This step is repeated until there are no new flooded nodes. Each of this repeating flooding steps is referred to as flooding wave. The identification of the fluid domain might require several flooding wave. Thus, this step plays

an important part in mesh generation time because it needs to check all nodes in every flooding wave, even the nodes that are not part of the fluid domain. In the earlier *Seeder* version, all nodes are at $\mathfrak{L}_{max}$ before the flooding step starts, which made this step an expensive step in mesh generation. Additionally, the flooding step requires 6 direct neighbors for a node and they need to be identified for all nodes and stored in memory, resulting in unnecessary memory and time consumption. However, the idea of flooding a node only if any of its face neighbor is fluid results in a watertight mesh. The watertight is essential to create an enclosed computational domain by avoiding leakage of flooding into the non-computational domain. Hence, a new efficient mesh generation algorithm was developed with reduced memory consumption and mesh generation time but keeps the watertightness property. Thus, *Seeder* [63] was completely re-implemented from configuration file to mesh generation.

The optimized new *Seeder* [63] algorithm is given in Alg. 1. An important optimization step in this algorithm is the building of a *protoTree* where only the nodes with intersected boundary geometry are refined to the level defined for that boundary and the fluid nodes away from the boundary geometry are refined to the desired level only after the flooding. This building *protoTree* step drastically reduces the memory consumption, neighbor identification, flooding, boundary computation and in-turn the overall mesh generation time.

**Algorithm 1** Seeder - Octree mesh generation

1: Load Lua configuration file
2: Create protoTree by refining bounding cube until level of the boundary object is reached
3: Identify neighbors of each node in protoTree
4: Recursively flood nodes starting from node with seed object towards the intersected boundary node
5: Refine flooded leaf nodes to their finest level
6: Recursively traverse through the tree and identify boundaries for the node adjacent to the intersected boundary node
7: Convert protoTree To TreElm
8: dump mesh into disk in TreElm format

In the *Seeder* configuration, the geometries and the level to refine each geometry are attached to one of the following attributes:

- *seed:* Nodes intersected with seed geometry are marked as "flooded"

(fluid) nodes. They are source nodes of the flooding step and helps to decide which part of the domain to flood, i.e. identification of fluid domain.

- *boundary:* Nodes intersected with boundary geometry are marked as "intersects boundary" nodes. Only these nodes are refined to the level defined by the user while building the preliminary tree (*protoTree*). For this attribute, in addition to the geometries and the level, a label must be defined by the user to create a unique boundary ID. This label is later used by the solver to set BC.

- *refinement:* Nodes intersected with refinement geometry are refined to their level after the flooding step. Only the flooded nodes are further refined to the level defined by the user.

Regarding the geometries, most basic geometrical shapes such as point, line, plane, box, triangles, sphere and cylinder are supported in addition to *STL* files. These basic geometrical shapes are internally defined in *Seeder*. The geometry provided as STLs is converted into triangles. The intersection of these basic shapes with cubes is implemented in *Seeder* and is used to refine or attach certain property to a node. In the following, the building of the *protoTree* and the flooding are explained in detail with a 2D example. But before that some variables are introduced to ease the dicussion. $\mathfrak{L}_{min}$ is the minimum level to which all fluid nodes should be refined. $\mathfrak{L}_{bnd}$ and $\mathfrak{L}_{refine}$ are the level of s particular boundary and the refinement attribute respectively. Each node is refined to its maximum level using $\mathfrak{L}_{max} = max(\mathfrak{L}_{min}, \mathfrak{L}_{bnd}, \mathfrak{L}_{refine})$. $\mathfrak{L}_{globalmax}$ is the maximum refinement level possible and it's 20 since $\mathfrak{tID}$ is defined as 8-bytes (64-bit) signed long integer. In addition to the limit on the maximum refinement level, there is also a limit on the number of nodes ($nNodes$) as largest integer value since the loop variable which iterates over nodes from 1 to $nNodes$ is defined as 4 bytes (32-bit) signed integer. This could be overcome by changing the iteration counter to 8-bytes (64-bit) signed long integer or with parallel mesh generation. If $\mathfrak{L}_{max} > \mathfrak{L}_{min}$ then the resultant computational mesh will be a multi-level mesh with different sizes of the fluid elements. On the other hand, if $\mathfrak{L}_{max} \leq \mathfrak{L}_{min}$ then the mesh will be a single-level mesh, i.e. size of all fluid elements will be same. The multi-meshes are used in applications which involves multiple spatial scales like the propagation of sound wave from the wind turbine blades. In such applications, the ratio of the size of the flow domain and the geometry is quite high and usage of single-level mesh to resolve all scales would result in a mesh with the huge number of elements. The computations on such large

mesh will be very expensive and sometimes becomes impossible due to the memory limit. With the multi-level mesh, the number of elements can be drastically reduced. However, the data must be exchanged between the levels and it is achieved by either interpolation or polynomial evaluation depending on the solver. In this work, the single-level mesh is used because the flow channels in the ED module are narrow and the spacer filaments are closer to each other.

**Building the protoTree** A first step in *Seeder* is to load the configuration file which includes loading STL files too. Then, the *protoTree* is build level-wise by iterating over levels from 1 to $\mathfrak{L}_{globalmax}$. The parent nodes are subdivided until those parent nodes, which are intersected by boundary geometry are refined to their $\mathfrak{L}_{max}$. The algorithm used to build the *protoTree* is given in Alg. 2. In the *protoTree*, the following information are stored per node:

$\mathfrak{tID}$: A unique ID of a node.

- Virtual: A parent (intermediate) node which is refined further.

Property: Used to attach certain properties to a node. Fortran 'ibset' intrinsic function is used to set certain bit position

  - Leaf: A node has reached $\mathfrak{L}_{max}$ of the "intersects boundary" geometry or not intersected by any geometries.
  - Flooded: A node is intersected by the seed geometry or flooded during flooding algorithm. A node is marked as flooded only if it is not intersected by the boundary geometry.
  - Intersects boundary: The node intersects the boundary geometry.
  - Has boundary: One of the 26 neighbor has the "intersects boundary" property
  - Wetface(6): A node with wet face required for flooding algorithm. One bit for each face. Whenever a node is marked as fluid, the face of its neighbor nodes facing the fluid node are marked as wet using the appropriate bit.

- HasBoundaryBit: Stores which of the 26 directions of a node has intersects boundary. One bit for each direction. In 26 directions, the first 6 are direct face neighbors and the next 12 are edge neighbors and the last 8 corner neighbors.

- MinBCID: BCID is a unique ID given to each unique boundary label according to the order in which the boundaries are defined in the configuration file. As a rule of thump, the minBCID refer to the boundary that is defined first in the configuration file among the multiple boundaries which are intersected by a node. This single integer value is very useful in choosing the boundary when a node is intersected by multiple boundary geometries.
- Intersected_objects: Contains the list of geometrical objects intersected by a node.
- $\mathfrak{L}_{min}$: Minimum level to refine a node
- Linkpos(6): Stores location of 6 neighbors for leaf nodes in *TreeID* list and the first child for virtual nodes at linkpos(1).

Here, $\mathfrak{tID}$s are stored as 'dynamic array' and all others as 'growing array'. Both these arrays are a description to create a growing array of data structures. But the dynamic one creates an array of unique values by adding the same value only once. It is achieved by using a sorted index list, which is used to fast loop up a given value using a binary search. If the value already exists in the array then the position of that value in the array is returned, otherwise a new value is appended to the array and the nVals in an array is increased by 1 and the return position is assigned to nVals. The nVals is the number of values in the array. It is different from the actual array size because the array size is always doubled if the nVals is greater than the array size. Due to the additional sorted index array in the 'dynamic array', it consumes more memory than the growing array. Therefore, it is used only to store data which requires unique entries such as the array of $\mathfrak{tID}$s. The other arrays are accessed using the position of a node in the $\mathfrak{tID}$ array. Additionally, for every level, the positions of the first and the last node in the sorted list of the dynamic array of $\mathfrak{tID}$ are stored. They help to loop over the nodes per level.

**Algorithm 2** Building of *protoTree*

```
1: if boundary geometries are defined then
2:     mark root node as intersects boundary
3: else
4:     mark root node as leaf
5: end if
6: nNodes ← 1
7: Initialize firstNode and lastNode on level=0 to nNodes
8: for iLevel ← 1, 𝔏_globalmax do
9:     firstNode(iLevel:) ← nNodes + 1
10:    for iParent ← firstNode(iLevel − 1), lastNode(iLevel − 1) do
11:        if iParent intersects boundary and also leaf then
12:            for iDir ← 1, 26 do                ▷ Loop over all 26 directions
13:                Set hasBoundary for neighbor node
14:            end for
15:        end if
16:        if iParent leaf then
17:            linkpos ← identify neighbors on 6-faces
18:            if iParent flooded then
19:                Wet neighbors face of the neighbor node
20:            end if
21:        else
22:            linkPos(1) of iParent ← nNodes + 1
23:            Create 8 children
24:            nNodes ← nNodes + 8
25:            for iChild ← 1, 8 do
26:                for Each iChild loop over intersected objects of its parent
   do
27:                    if iChild intersected with boundary object then
28:                        Mark iChild as intersects boundary
29:                        Inherit intersected objects from parent to iChild
30:                        𝔏_max ← max(𝔏_min, 𝔏_bnd)
31:                        if 𝔏_max <= iLevel then
32:                            Mark iChild as leaf
33:                        end if
34:                    else if iChild intersected with seed object then
35:                        Mark iChild as flooded
36:                    else    ▷ Not intersected by any geometrical objects
37:                        Mark iChild as leaf
38:                    end if
39:                end for
40:            end for
41:        end if
42:    end for                                                          83
43:    lastNode(iLevel:) ← nNodes ▷ Set last node position in sorted list
   to total number of nodes created so far
44: end for
```

Alg. 2 starts with an initialization of the root node by marking it as "intersects boundary", if a boundary geometry is defined in the configuration file, else it is marked as a leaf node. Inside the level loop, for every parent node intersected with boundary geometry, 8 children are created and these children inherit intersected geometry information from their parent. A node which is not intersected by any boundary geometry or reached its level of the "intersects boundary" is marked as leaf. Then, the 6 face neighbors of the leaf node are identified. A leaf node which is intersected by the seed geometry is marked as fluid (flooded) and 6 of its face neighbors, face is marked as wet. Wetting neighbors faces helps to flood a non-intersected boundary leaf node in the flooding step. For a leaf node, the position of six direct neighbors which are connected to its faces in the dynamic array of *TreeID* are identified to optimize the flooding. If the neighbor is not found on the same level as the leaf node level then the parent of neighbor must exist in *protoTree* and its position is identified. Thus, neighbors are identified recursively in *protoTree* by looping from current level to 0. It's worth mentioning that the *protoTree* algorithm creates nodes level-wise, so the entries (tIDs) in the dynamic array of tID are sorted as shown in Figure 5.4. For a leaf node with intersected boundary geometry, the minimum boundary ID (bcID) is stored. If a node is intersected by multiple boundaries then the minimum bcID of those boundaries is used. This bcID of a boundary depends on the order in which boundary objects are defined in the configuration file. At the end of this step, the nodes intersected with boundary geometries are fully resolved and non-intersected geometry nodes are at their coarsest levels. Additionally, there must be at least one fluid node in *protoTree* to build the fluid tree in flooding. Therefore, a seed geometry must be defined such that it does not overlap with the boundary geometry because if a node is intersected by both seed and boundary geometry then it is not flooded. The efficiency of generating the *protoTree* depends on how many intersected objects need to be checked per node.

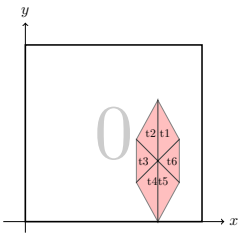



**Figure 5.3** 2D bounding cube (tID = 0) with a boundary with six triangles

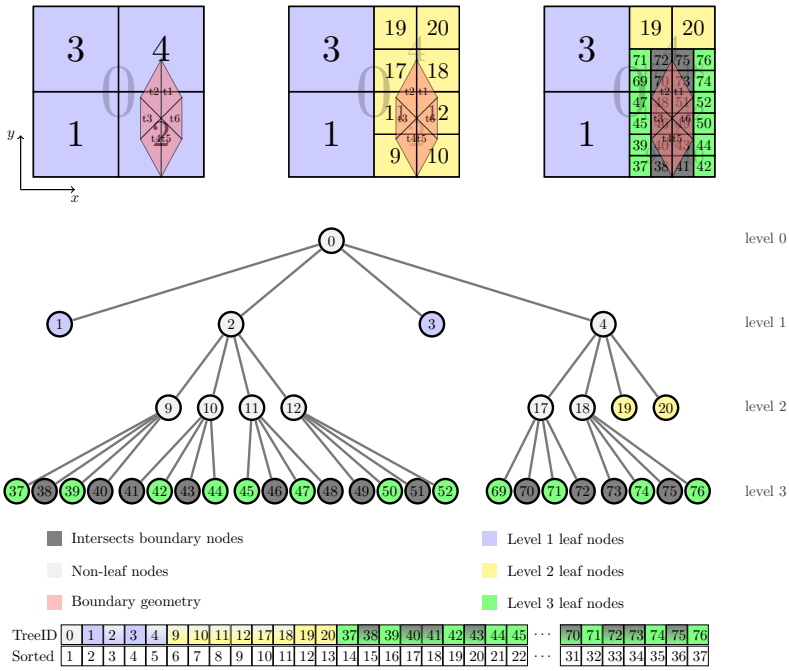



**Figure 5.4** *Prototree* generation at every refinement level with $\mathfrak{L}_{bnd} = 3$

The build-up of the *protoTree* is illustrated using a 2D (quadtree) example with a boundary of 6 triangles (geometries). The bounding cube and the triangles are shown in Figure 5.3. Let us consider that the user defines the level for a boundary of 6 triangles to be $\mathfrak{L}_{bnd} = 3$. It means that the nodes intersecting these triangles are required by the user to be refined up to level 3. The generation of the quadtree at every level with a boundary level $\mathfrak{L}_{bnd} = 3$ is illustrated in Figure 5.4. The color of the node represents their level except for nodes with intersected boundary geometries, which are represented by gray color. The $\mathfrak{tOD}$ list and its sorted index list are also shown in the figure.

**Flooding** The algorithm works by flooding the domain, starting from seed nodes up to intersected boundary nodes. This approach is quite robust even to broken STL definitions, as any crack below the resolution is automatically healed and there is no dependence on the orientation of the surfaces. Since flooding takes place only on 6 direct neighbors, which are

connected by faces, there will be no fluid nodes which are connected by edges or corners. The flooding algorithm is given in Alg. 3. The algorithm iterates over all nodes in the *protoTree* from breath-first numbering fashion which is the order in which the nodes are created and stored in $\mathfrak{tOD}$ list. If a node does not not intersect a boundary and is not flooded and is leaf node with any wet face then the node is flooded and its 6 direct neighbors faces are wetted. If a node is a not a leaf then its wet faces are inherited to its eligible children faces. This iteration is done several times until there are no new flooded nodes. Each iteration is refered to as one flooding wave. After flooding the leaf nodes, all virtual (intermediate) nodes with at least one fluid child are also flooded to speed up the extraction of the final fluid tree from the *protoTree* in *convert protoTree to TreElm* step. The efficiency of the flooding step depends on the number of flooding waves required to identify the fluid domain. The number of flooding waves can be reduced by a good placement of seed geometries covering most of the fluid domain.

**Algorithm 3** Flooding

```
1: repeat
2:     for iNode ← 1, nNodes do
3:         if leaf node then
4:             if not intersected boundary node and not flooded then
5:                 if any face is wet then
6:                     Mark this node as flooded
7:                     Wet 6 [3D] / 4 [2D] direct neighbors face connected
   to this node
8:                 end if
9:             end if
10:        else ▷ Virtual node inherit the wet face to its eligible children
11:            for iFace ← 1, 6 do
12:                if iFace is wet then
13:                    Inherit the wetness to its direct children on this iFace
14:                end if
15:            end for
16:        end if
17:    end for
18: until no new node is flooded
```

The flooding algorithm is depicted in Figure 5.5 using a simple quadTree example with a seed placed at node 1. Figure 5.5 also shows the inheritance of a wet face from virtual nodes to their eligible children faces. Wetting

of neighbors face can be understood by looking at the flooded node 1. In this work, the terms: east, west, north and south refer to $-x$, $+x$, $+y$ and $-y$ directions respectively. The node 1 wets west face of node 2, south face of node 3, east face of node 2 and north face of node 3. The east face of node 2 and north face of node 3 are wetted since the bounding cube is a periodic domain. Due to inheritance of wet faces from non-leaf nodes to children, this example is flooded in a single wave. The flood nodes are highlighted in blue color in the $\mathfrak{tOD}$ list shown in the figure.

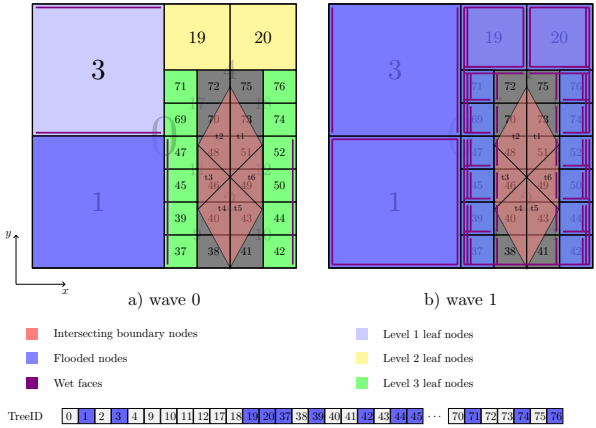



**Figure 5.5** Flooding of protoTree in 2D mesh

In case, q-values are required by the Lattice Boltzmann scheme to increase the accuracy of the simulation as explained in Section 3.1.2. Then, an additional step is performed in the flooding algorithm. After flooding the inner domain, the algorithm iterates over all nodes again but only over the nodes with "intersects boundary" property and q-values are determined by calculating the normalized distance between the "intersects boundary" node and the intersected geometries. To support this discussion, these intersected boundary nodes are referred to as current nodes. The current nodes are flooded only if there is a wet face and there are no intersections with boundary geometries between the current node and the flooded neighbor node along the wet face direction. The intersection of boundary geometries between the current node and the flooded neighbor node is done by checking the intersection of a ray from the current node barycenter along the wet face direction against the intersected boundary geometries in the current node. This additional step for q-values can be determined for any shape of the boundary. Figure 5.6 shows the *protoTree*

after flooding with and without q-value. It can be seen that in Figure 5.6b with q-value, the nodes 38, 41, 70, 72, 73 and 75 are flooded. Note that only these nodes barycenter are outside the triangles, while other "intersects boundary" nodes barycenter are inside the triangles.

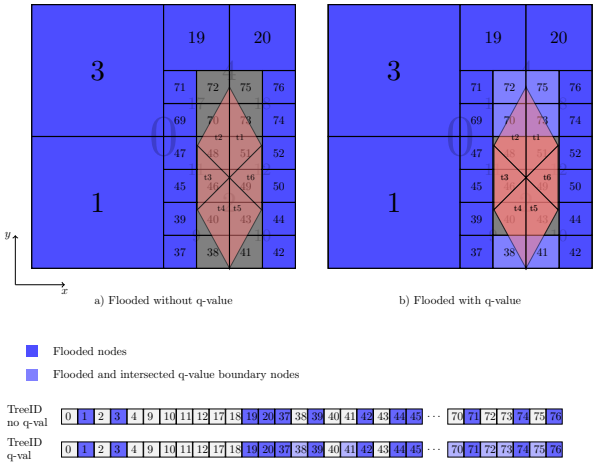



**Figure 5.6** ProtoTree after flooding with and without q-values

**Refine leaf** After flooding, the computational domain is identified but there are leaf nodes in the *protoTree* which might exist on a level lower than $\mathfrak{L}_{min}$. Thus, in the refine leaf step, all flooded leaf nodes which are not "intersects boundary" are refined further if the level of that node is lower than $\mathfrak{L}_{min}$. If refinement geometries are defined then the nodes intersected by refinement objects are refined to the level of the refinement attribute by inheriting the refinement object to the children. To avoid any level jump between a flooded node and a boundary neighbor node, a flooded node is refined if a boundary neighbor node in any of the 26 directions exist in higher level. Newly created children inherit all the properties from their parent and the leaf property in the parent node is removed. The algorithm to refine flooded leaf nodes is given in Alg. 4. This algorithm also iterates over parent nodes level-wise similar to the building of the *protoTree* step but the major difference is that in the refine leaf step, only the flooded leaf nodes with no "intersects boundary" are further refined. New children are appended to the dynamic array of $\mathfrak{tID}$s resulting in an unsorted array. This is the reason why the first node and the last node

for every level refer to the sorted index array in the dynamic $\mathfrak{tID}$ array. The *refine leaf* step is illustrated in Figure 5.7 where nodes 1 and 3 are refined further to $\mathfrak{L}_{min} = 2$. The unsorted $\mathfrak{tID}$ list and its sorted index and serialized fluid IDs ordered by SFC are also depicted in the figure.

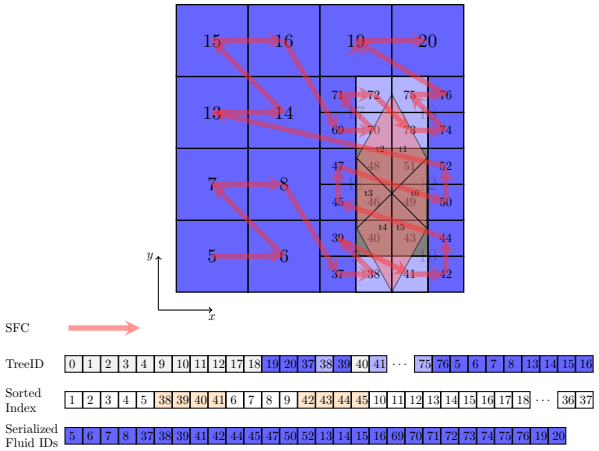



**Figure 5.7** ProtoTree after the refine leaf showing the SFC ordering of the fluid tree. Newly created 8 leaf nodes are appended to the end of the $\mathfrak{tID}$ list and its sorted index is updated accordingly. Serialized fluid IDs array shows the final fluid $\mathfrak{tID}$s ordered by the SFC.

**Convert protoTree to *TreElM*** After the refine leaf step, all the flooded leaf nodes are refined to their finest level in an unsorted fashion in the dynamic array of the $\mathfrak{tID}$ list. To minimize storage, only the flooded leaf nodes in the *protoTree* are extracted and sorted as required by *TreElM*. It is achieved by traversing down the *protoTree* in the order of depth first manner following the SFC. This results in the sparse representation of the tree instead of a full matrix representation. While traversing down the *protoTree*, only flooded nodes are considered because there is no need to traverse down non-flooded nodes, which are not part of fluid domain. This is the reason why virtual nodes with at least single flooded children are flooded at the end of flooding step. For the flooded leaf nodes with *HasBoundary* property, BCIDs are identified in all 26 directions. If a neighbor is fluid then the BCID is set to zero. Note that in flooding, only the 6 direct neighbors are flooded so there might be a neighbor node in the other 20 directions which might not be flooded and not intersected by

**Algorithm 4** Refine leaf nodes in *protoTree* to their level

```
1: nNodes ← 0
2: for iLevel ← 1, 𝔏_globalmax do
3:     for iParent ← firstNode(iLevel − 1), lastNode(iLevel − 1) do
4:         if iParent is flooded, leaf and non-intersected boundary then
5:             𝔏_max ← 𝔏_min
6:             if iParent intersected by refinement geometry then
7:                 𝔏_max ← max(𝔏_min, 𝔏_refine)
8:             end if
9:             if iParent has boundary neighbor then
10:                𝔏_max ← max(𝔏_min, 𝔏_bnd,neighbor)
11:            end if
12:            if 𝔏_max ≥ iLevel then
13:                Create 8 childrens
14:                nNodes ← nNodes + 8
15:                for iChild ← 1, 8 do
16:                    for each iChild loop over intersected objects of its
   parent do
17:                        if iChild intersected with refinement object then
18:                            Inherit that intersected refinement object from
   parent
19:                        end if
20:                    end for
21:                end for
22:            end if
23:        end if
24:    end for    ▷ Shift last node position in sorted list for iLevel with
   nNew nodes created on this level
25:    lastNode(iLevel) ← lastNode(iLevel) + nNodes ▷ Shift first node
   and last node position in sorted list for all levels higher than iLevel
26:    firstNode(iLevel+1,:) ← firstNode(iLevel+1,:) + nNodes
27:    lastNode(iLevel+1,:) ← lastNode(iLevel+1,:) + nNodes
28: end for
```

boundary, these nodes are referred to as hanging nodes. If a hanging node is encountered as neighbor then the BCID for that direction is set based on the BCID on adjacent face neighbors. As a rule of thump, the BCID in the direction of the handing node is set to minimum of BCID (minBCID) of other directions. In addition to the BCIDs in all 26 directions for flooded

leaf node with boundary neighbor, the *HasBoundary* property is also extracted because the property of a node is required by the solver to apply special treatments. Furthermore, q-values are calculated for nodes with q-value boundaries and the property for those nodes are set to *HasQval*. The serialized final fluid $\mathfrak{tJD}$ list and the SFC connection of the consecutive elements in the quadTree example are shown in Figure 5.7

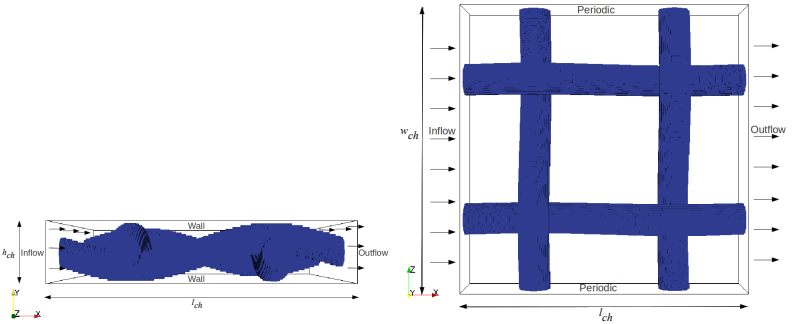
It would be efficient to combine this step with the refine leaf step by refining a leaf node if it is not at its finest level while traversing through the tree in depth-first manner. However, it leads to difficulties in parallelization, and therefore it is not applied.

**Dump to disk** Finally, the serialized fluid $\mathfrak{tJD}$ list, the boundary identifications (BCIDs) and the q-values are dumped to disk in binary format. For every node, a tuple of 2 entries is stored: $\mathfrak{tJD}$ and property like *HasBoundary* or *HasQval* of that node are written in native binary format. Additionally, for every node with *HasBoundary* and *HasQval* property, BCID and q-values respectively for all 26 directions are written in separate binary files. For every binary file, a header file in ascii format is created with basic information about the binary file like the number of fluid elements, the number of boundary elements, bounding cube description, minimum and maximum level in the fluid tree, etc. This header file is written in the form of a Lua script. The $\mathfrak{tJD}$ and the property bit are 8-bytes each and, BCID and q-values are 8-bytes per direction each. Thus, inner fluid nodes consume 16-bytes per node and fluid nodes near boundary consume additional 208-bytes per node and fluid nodes next to q-value boundaries require also additional 208-bytes per node.

### 5.3.1 Mesh generation with spacer geometry

As previously mentioned, the complex spacer geometry is used in dilute and concentrate flow channels to keep the membranes apart from each other and maintain mechanical stability. The simulation setup with woven spacer geometry is illustrated in Figure 5.8. The woven spacer with 2 filaments in x- and z- direction constitutes a single spacer element. The length and width of this single spacer element is $L = W = 0.2\,\text{cm}$ with distance between the filaments $l_m = 0.1\,\text{cm}$. The diameter of the filament, $d_f = 0.02\,\text{cm}$ and channel height, $h_{ch} = 2 \cdot d_f$.

The mesh generation of the woven spacer geometry with 6 filaments in $x$-direction ($l_{ch} = 0.6\,\text{cm}$ and 4 filaments in z- direction $w_{ch} = 0.4\,\text{cm}$ using *Seeder* is illustrated in Figure 5.9. The spacer geometry is provided as STL file generated using the 3D modeling and rendering package Blender



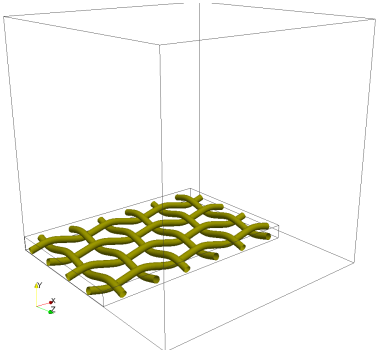

(a) Side view (b) Top view

**Figure 5.8** Simulation setup with interwoven laboratory spacer structure

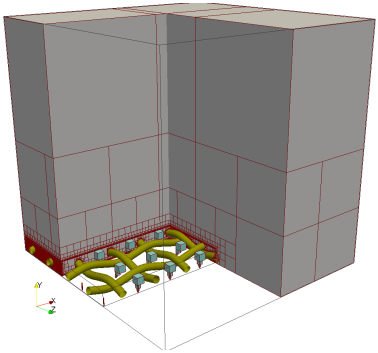
[97]. For the woven spacer, only a single spacer element is generated using Blender, which contains 33024 triangles. For longer channels, a single spacer element is duplicated using the translation feature in *Seeder*. To improve the accuracy of the numerical simulation near the spacer structure, the q-values are calculated on the spacer geometry. In tree based mesh generation, the mesh resolution or element size ($\delta x$) depends on the level to which the node is refined. The bounding cube length ($L_{BndCube}$) depends on element size $\delta x$ and level $\mathfrak{L}$ as

$$L_{BndCube} = (2^{\mathfrak{L}}) \cdot \delta x. \tag{5.4}$$

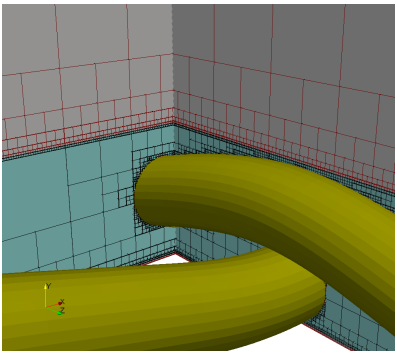
Usually in numerical simulations, it is preferred to define either the mesh resolution $\delta x$ or the required number of elements in a characteristics length. Here, the mesh resolution $\delta x$ is defined by fixing the number of elements in the height of channel $nH_{ch}$ as $\delta x = h_{ch}/nH_{ch}$. In order to create the mesh with a certain length, the bounding cube length should be at least 2 elements sizes larger than the maximum of length, width and height of the fluid domain to include one element for boundary in each direction of the enclosed computational domain. In spacer flow channels, the width of the channel is assumed to be periodic and only the length of the channel is larger or equal to the width of the channel. Thus, the bounding cube length is set to be $L_{BndCube} = l_{ch} + 2 \cdot \delta x$ or the number of elements in the bounding cube as $nL_{BndCube} = nL_{ch} + 2$. Here, $nL_{ch}$ is the number of elements in the length of the channel. With this, the level



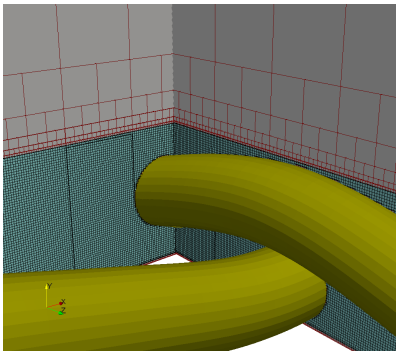
(a) Outline of bounding cube and computational domain around spacer



(b) After build protoTree, nodes are refined towards spacer and boundaries enclosing computational domain. Cyan color nodes represents seeds



(c) After flooding, nodes inside computational domain are flooded



(d) After refine leaf, all flooded nodes are refined to their finest level

**Figure 5.9** Mesh generation with woven spacer geometry.

to obtain the given $\delta x$ can be computed using

$$\mathfrak{L} = math.ceil(math.log(nL_{BndCube}, 2)). \tag{5.5}$$

Figure 5.9a shows the outline of the bounding cube and the computational domain around woven spacer (colored olive). In build protoTree, the nodes are refined towards the spacer geometry and boundaries enclosing the computational domain such as inlet at $x = 0$, outlet at $x = l_{ch}$, bottom wall at $y = 0$, top wall at $y = h_{ch}$, back $z = 0$ and front $z = w_{ch}$. Here,

the front and back are defined as periodic boundaries. To decrease the number of flooding waves, the seeds are defined as a line along the height between the filaments for each single spacer element, which are colored in cyan. The *protoTree* after flooding and refine leaf is shown in Figure 5.9c and Figure 5.9d respectively. Since the channels are so narrow and filaments are closer to each other, all fluid elements are refined to same level. The only disadvantage of this single level mesh is that it will be too expensive when it comes to resolving the electrical double layer.

The efficiency of the *Seeder* mesh generation with spacer is analyzed using two cases. In the first case, the element size $\delta x$ is varied on single spacer element and in the second case, the channel length $l_{ch}$ is varied by duplicating the single spacer element along the length for fixed $\delta x$. This analysis was performed on Intel(R) Xeon(R) CPU E5-2650 2.00GHz processor with 20MB Cache and 380 GB of main memory. The mesh element size $\delta x_{nH_{ch}}$ is varied by varying number of elements in the height i.e. the mesh resolution $nH_{ch}$ from 32 to 256. The total runtime to generate a mesh for each of these resolutions are tabled in Table 5.1 along with the number of fluid elements (nFluids) and the number of fluid elements with boundary neighbor (nBnds). Additionally, the virtual main high water mark memory (VmHWM) consumption and the size of mesh files dumped to disk are also tabled.

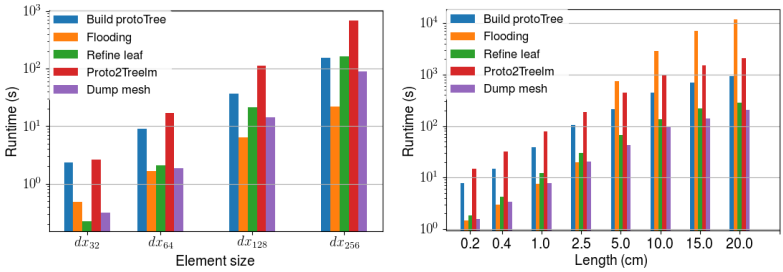
| $nH_{ch}$ | $\delta x$ ($\mu$m) | nFluids | nBnds | Runtime (s) | VmHWM (GB) | Mesh Files (GB) |
|---|---|---|---|---|---|---|
| 32 | 12.5 | 686 844 | 111 756 | 6.0 | 0.12 | 0.0092 |
| 64 | 6.25 | 5 503 785 | 455 712 | 31.4 | 0.67 | 0.043 |
| 128 | 3.125 | 44 048 647 | 1 837 692 | 191 | 4.48 | 1.21 |
| 256 | 1.5625 | 352 456 277 | 7 377 172 | 1113 | 32.63 | 7.6 |

**Table 5.1** Mesh generation of single spacer element for various resolution

The resolution $\delta x_{256}$ results in roughly 350 million fluid elements and it was generated in less than 20 min. Doubling it i.e. $\delta x_{512}$ failed during the build *protoTree* step due to the 2 billion limit of the largest 4-byte (32-bit) signed integer which was used as loop variable $iNode$ and total nodes counter $nNodes$. Note that in a octree mesh, nFluids increases roughly 8 times with doubling the resolution. The detailed split up of mesh generation run time by every step in *Seeder* is tabled in Table 5.2 and plotted in Figure 5.10a. It can be seen that the runtime of the refine leaf steps gets more expensive with increasing resolution because all the flooded leaf nodes at coarser level are refined at this step. The *proto2Treelm* step also gets expensive with increasing resolution because of boundary identification on 26 directions for fluid nodes next to the boundary node.

| $nH_{ch}$ | Build protoTree (s) | Flooding (s) | Refine leaf (s) | Proto2-treelm (s) | Dump mesh (s) |
|---|---|---|---|---|---|
| 32 | 2.35 | 0.48 | 0.23 | 2.61 | 0.32 |
| 64 | 8.95 | 1.67 | 2.07 | 16.8 | 1.87 |
| 128 | 36.9 | 6.46 | 21.5 | 112 | 14.2 |
| 256 | 154 | 22.2 | 161 | 687 | 88.8 |

**Table 5.2** Break down of time taken by every mesh generation step for single spacer element for various resolution




(a) For various mesh resolution

(b) For various spacer length

**Figure 5.10** Comparison of time taken by steps involved in mesh generation with spacer geometry.

In addition, the q-values are calculated on 26 directions for the nodes next to boundary nodes intersected by spacer geometry. In the spacer-filled flow channel, there are more nBnds but with increase in mesh resolution the ratio of nFluids to nBnds decreases. In other words, the number of inner fluid elements in the space between the filaments increases. Both the build protoTree and the flooding step costs roughly 4 times more at every increment in mesh resolution. This increase in cost is due to the creation of more nodes nearing spacer geometry.

From the convergence analysis (see Section 7.1.2.1), it was found that the difference in pressure drop across the channel between resolution $\delta x_{64}$ and $\delta x_{128}$ was below 3%. Thus, the resolution $\delta x_{64}$ is enough to approximate the spacer geometry and also reduce the numerical computation time. Therefore, for the second analysis of the mesh generation with spacer, the resolution is fixed to $\delta x_{64}$ and the channel length $l_{ch}$ is increased from a single spacer element length of 0.2 cm to the length of 20 cm. The limit of maximum length 20 cm is due to the fact that the woven spacer used in the laboratory of AVT, RWTH Aachen has the dimension $L \times W \times H$: 20 cm × 10 cm × 0.04 cm. For lengths larger than a single spacer element, the STL of a single spacer element was duplicated using the *Seeder* translation

feature instead of creating a large STL by Blender. The width of the spacer is kept to 0.2 cm and it is assumed to be periodic along the z-direction. The mesh generation time, nFluids, nBnds, VmHWM and mesh file size for various spacer lengths for resolution $\delta x_{64}$ is tabled in Table 5.3. The full spacer of length 20 cm has roughly 618 million fluid elements and 14% of them are fluid elements with boundary neighbor. This largest mesh was generated in around 4.5 Hrs with a virtual main memory consumption of 68 GB.

| $l_{ch}$ (cm) | nFluids | nBnds | Runtime (s) | VmHWM (GB) | Mesh files (GB) |
|---|---|---|---|---|---|
| 0.2 | 5 503 785 | 455 712 | 27.3 | 0.67 | 0.21 |
| 0.4 | 11 006 960 | 876 020 | 57.1 | 1.32 | 0.42 |
| 1.0 | 27 516 471 | 2 136 908 | 145 | 3.52 | 1.05 |
| 2.5 | 68 789 975 | 5 289 186 | 364 | 8.37 | 2.62 |
| 5.0 | 150 678 658 | 10 686 695 | $1.51 \times 10^3$ | 17.1 | 5.48 |
| 10 | 301 356 308 | 21 337 720 | $4.58 \times 10^3$ | 33.85 | 10.96 |
| 15.0 | 457 801 588 | 32 020 733 | $9.64 \times 10^3$ | 51.76 | 16.53 |
| 20.0 | 618 441 508 | 42 727 010 | $1.52 \times 10^4$ | 68 | 22.17 |

**Table 5.3** Mesh generation for various spacer length with fixed $nH_{ch} = 64$

As expecteds nFluids, nBnds, VmHWM and mesh file size increase roughly by a same factor with increase in length i.e, linearly. However, the runtime for the mesh generation increases linearly only up to the length of 2.5 cm and from 2.5 cm to 5 cm it suddenly increases by a factor of 4 and from 5 cm to 20 cm it is increased by a factor of 10. Looking into the runtime of each mesh generation step tabled in Table 5.4 reveals that the non-linear increase in mesh generation time for large channel length is caused by the flooding step. This can be seen in Figure 5.10b. Since seeds are defined between every filament, the flooding should scale linearly but this is not the case. A probable cause might be the increase in bounding cube length which also increase the channel length, and this results in different resolution near the boundaries which might have lead to more flooding waves. Consider an example: if there is only one finer element next to the boundary then it gets flooded because it inherits the wet face from its parent. However, if there are two fine elements along the normal direction of the boundary, then the fine element next to the boundary gets flooded only when the neighbor fine element gets flooded first and this might result in more flooding waves.

| Length (cm) | Build protoTree (s) | Flooding (s) | Refine leaf (s) | Proto2-treelm (s) | Dump mesh (s) |
|---|---|---|---|---|---|
| 0.2 | 7.76 | 1.44 | 1.81 | 14.7 | 1.56 |
| 0.4 | 15.0 | 2.93 | 4.17 | 31.6 | 33.7 |
| 1.0 | 3.91 | 7.44 | 12.1 | 77.9 | 7.88 |
| 2.5 | $1.04 \times 10^2$ | 19.7 | 30.1 | $1.89 \times 10^2$ | 20.7 |
| 5.0 | $2.14 \times 10^2$ | $7.37 \times 10^2$ | 66.1 | $4.53 \times 10^2$ | 42.5 |
| 10 | $4.5 \times 10^2$ | $2.92 \times 10^3$ | $1.34 \times 10^2$ | $9.75 \times 10^2$ | 97.7 |
| 15.0 | $7.03 \times 10^2$ | $7.05 \times 10^3$ | $2.17 \times 10^2$ | $1.53 \times 10^3$ | $1.39 \times 10^2$ |
| 20.0 | $9.43 \times 10^2$ | $1.17 \times 10^4$ | $2.84 \times 10^2$ | $2.11 \times 10^3$ | $2.06 \times 10^2$ |

**Table 5.4** Break down of time taken by every mesh generation step for various spacer length with fixed $nH_{ch} = 64$

## 5.4 Musubi - lattice Boltzmann solver

*Musubi* [32] is the highly scalable lattice Boltzmann solver in the *APES* framework. The lattice Boltzmann equation (LBE) for single component flow, multicomponent flow and electric potential presented in Section 3.1, Section 3.2 and Section 3.4 are implemented in *Musubi*. It also offers numerical solution of other physical systems like weakly compressible flows, non-Newtonian flows, clotting process, and few more. Each of these physical systems is implemented as a dedicated compute kernel performing streaming and collision. For each physical system, a dedicated compute kernel is implemented for various stencils like $D2Q9$, $D3Q19$, $D3Q27$, etc. and for various collision operators like $BGK$, $MRT$, etc. *Musubi* offer full flexibility to chose a specific compute kernel by configuring LBE, stencil layout and collision operator in the configuration file. In addition to dedicated compute kernels, the various boundary conditions and source terms for each physical system are also configurable. The variables required at boundaries and source terms are defined as space-time function, explained in Section 5.5.2. *Musubi* also offers a multi-level solution with different interpolation techniques [32]. The detail description and implementation details of this software can be found on [31, 32, 79]. Here, in this thesis the LBM stream-collide algorithm used in the compute kernel for the single component flow, the multicomponent flow and the electric potential are briefly discussed. Furthermore, the algorithm of the *Musubi* main program is also presented to explain the coupling algorithm in a later section.

### 5.4.1 Domain decomposition

The octree mesh generated by *Seeder* is provided to the solver as the level independent list of fluid elements serialized by depth-first order following the SFC. In the solver, this level independent list of elements is splitted and distributed to each processor such that each processor gets almost the same number of elements. The advantage of the SFC is that it maintains a fair degree of locality and reduces the communication surface between the processors and achieves almost perfect load balancing. However, in real applications with boundaries and sources terms, some fluid elements cost more than others which results in load imbalance. This imbalance is resolved by deploying the load balancing algorithm in *TreElM* which uses SPartA [29] to redistribute the elements using weights calculated from the computational cost of each element. Thus, with SFC for reducing the communication surface and the load balancing algorithm balancing computational cost, the good scalability is achieved on large parallel systems.

### 5.4.2 Topology unaware solver data structure

Before getting into the LBM algorithm, the computation on elements and the data layout of the state (PDF) values in the solver are briefly discussed. To achieve good peak performance the computational cost per processor must be optimized. It is achieved by vectorization, minimizing the number of floating point operations and by efficient usage of cache memory. To vectorize the computations, the level independent list of fluid elements provided by the *Seeder* is converted into a level-wise list of elements. Additionally, to apply the stream-collide algorithm, the solver only requires the relationship between the elements (direct neighbors) depending on the stencil definition. A stencil is defined as set of offset directions describing the relative position of an element. Therefore, for each element in the level-wise list, the direct neighbors according to the stencil are identified by searching the neighbor $\mathfrak{tID}$ in the level independent serialized list of elements. If $\mathfrak{tID}$ is not found in local processor then the processor which contains the $\mathfrak{tID}$ is identified locally since each processor knows the first and last $\mathfrak{tID}$ of all processors. The element that is found on the remote processor is added as virtual halo element in the level-wise list of elements. This allows the solver to apply vectorial operations on the list of elements and to be unaware of the complex geometries and the arbitrary nature of the tree. In addition to the compute kernels, the initial conditions, boundary conditions, source terms, multi-level interpolation

and communication between processors are applied to the level-wise list of elements. Thus, *TreElM* manages the topology of the tree and the solver operates only on level-wise list of elements with information about direct neighbors for each element on their level. For more details on neighbor search, stencil construction and tree topology refer to [48].

For each level, the PDF values are stored in a double buffer array, one for reading and one for writing. The index is swapped before every time step. The size of each PDF buffer is $N \times N_s \times Q$ where $N$ is the number of elements, $N_s$ is the number of species and $Q$ is the number of directions per element. For single component and electric potential LBM, $N_s = 1$ and for multicomponent LBM, it depends on number of species defined in configuration file. *Musubi* offers two different data layouts: Structure of Arrays (SoA) and Array of Structures (AoS). In this work, the AoS data layout is used to store values in the PDF buffer, i.e. the PDF values of $Q$ directions for a single element are grouped together. In case of multicomponent LBM, the PDF values of $Q$ directions for all species for a single element are grouped together as shown in Figure 5.11 because the PDF of all species are required to compute the mixture velocity.

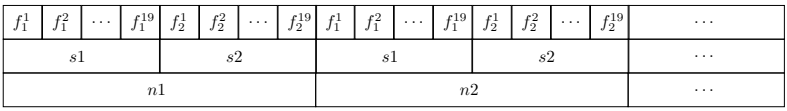



**Figure 5.11** Array of Structures data layout of state buffer for $D3Q19$ stencil and $N_s = 2$. $n$ represents element number and $s$ represents species number.

In the stream-colliide algorithm, the collision operation is local but the streaming operation requires direct neighbors in $Q$ directions to propagate the PDF from neighbor to local or vice versa. The 1D connectivity array is used to indirectly address the direct neighbors for each direction in the state array and size of this array is $N \times Q$. At boundaries where neighbor elements are missing, the link is reflected resulting in implicit bounce-back operation. Thus, explicit bounce-back treatment is not required for no-slip wall boundaries. Since streaming is just coping PDF values from local to neighbor or vice versa, the computational cost of compute kernel depends only on the collision operator.

### 5.4.3 Stream-collide algorithm

The LBE of any physical system Eq. 3.2, Eq. 3.55 and Eq. 3.75 can be generalized as

$$\hat{f}_k^m(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t + \delta t) = \hat{f}_k^m(\boldsymbol{x}, t) + \hat{\boldsymbol{A}}_{\boldsymbol{k}}(\hat{f}_k^{eq,m}(\boldsymbol{x}, t) - \hat{f}_k^m(\boldsymbol{x}, t)) + \hat{S}_k^m(\boldsymbol{x}, t) \tag{5.6}$$

where $\hat{f}_k^m$, $\hat{f}^{eq,m}$ and $\hat{\boldsymbol{A}}_{\boldsymbol{k}}$ are particle distribution function, equilibrium function and collision matrix respectively and $\hat{S}^m$ is the source or force term depending on the LBE. The subscript $k$ is the number of components/species and $k = 1$ for single component and electric potential LBM. The superscript $m = 1 \ldots Q$ represents the discrete velocity directions and $Q$ is the number of discrete directions according to the stencil. The evolution of this LBE is solved by two step procedure i.e streaming and collision. The collision step

$$\hat{f}_k^{m,c}(\boldsymbol{x}, t) = \hat{f}_k^m(\boldsymbol{x}, t) + \hat{\boldsymbol{A}}_{\boldsymbol{k}}(f_k^{eq,m}(\boldsymbol{x}, t) - f_k^m(\boldsymbol{x}, t)) + \hat{S}_k^m(\boldsymbol{x}, t) \tag{5.7}$$

describes the collision or interaction of the fluid particle by relaxing particle distribution function $\hat{f}_k^m$ towards the equilibrium distribution function $\hat{f}_k^{eq,m}$ with the relaxation collision matrix $\hat{\boldsymbol{A}}_{\boldsymbol{k}}$ as shown in Figure 5.12. $\hat{f}_k^{m,c}$ represents the post-collision PDF after the collide step. After the

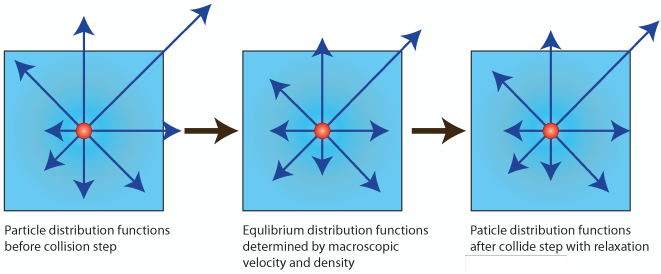



**Figure 5.12** Collide step of the LBM in $D2Q9$ stencil

collide step, the streaming step is performed by pulling or pushing the PDF from or to their neighboring cells respectively according to their velocity directions. The stream step can be written as

$$\hat{f}_k^m(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t + \delta t) = \hat{f}_k^{m,c}(\boldsymbol{x}, t). \tag{5.8}$$

Figure 5.13 shows the pulling of the PDF from neighboring cells.

The stream and collision for each LBE are implemented as dedicated highly optimized compute kernel by collapsing the stream and collide operations, minimizing the number of floating point operations and optimizing

(a) Before streaming

(b) After streaming

**Figure 5.13** Streaming step of the LBM in 2D with pulling $\hat{f}_k^m$ from neighbors

cache and memory access [102]. The kernel is implemented for different stencil layouts like $D2Q9$, $D3Q19$, $D3Q27$, etc and different collision operations like BGK, MRT, etc. The algorithm used within the compute kernel for the single component LBM, the multicomponent LBM and the LBM for the electric potential are given Alg. 5, Alg. 6 and Alg. 7 respectively.

**Algorithm 5** Compute kernel for the single component LBM

1: **for** each element in level-wise list of elements excluding halo elements **do**
2: Pull InPDF links from corresponding neighbor Eq. 5.13
3: Compute macroscopic density from PDF Eq. 3.4
4: Compute macroscopic velocity from PDF Eq. 3.5
5: Compute equilibrium function Eq. 3.3
6: Update outPDF with collision step Eq. 5.7
7: **end for**

The multicomponent LBM kernel is implemented for both ideal and nonideal liquid mixture. The kernel for nonideal liquid mixture is more involved and expensive since it requires thermodynamic factor and concentration dependent diffusivity coefficients. They are obtained by functions provided by AVT, RWTH Aachen. These functions are written in C++ so they are coupled with *Musubi* via Fortran-C interface. In addition, the linear equation system of size $N_s \times N_s$ needs to be solved for every element to obtain the species velocity of the original PDF. In *Musubi*, optimized kernel is implemented for $D3Q19$ stencil with three components for both ideal and nonideal mixtures. In the next section, the performance of the multicomponent LBM kernel for ideal mixture with three components is given. The number of floating point operations per element in the compute kernel of $D3Q19$ with three components is 850 i.e. 280 floating point

operations per element per component.

**Algorithm 6** Compute kernel for the multicomponent LBM for nonideal liquid mixture

1: **for** each element in level-wise list of elements excluding halo elements **do**
2: Pull inPDF links from corresponding neighbor Eq. 5.13
3: Compute macroscopic species density from PDF using Eq. 3.52a
4: Compute thermodynamic factor and diffusivity coefficient from local species concentration using C++ code from AVT, RWTH Aachen
5: Compute species macroscopic velocity of transformed PDF using Eq. 3.52b
6: Compute species velocity of untransformed PDF by solving linear equation system Eq. 3.55
7: Compute equilibrium velocity for nonideal mixture using Eq. 3.40
8: Compute equilibrium function using Eq. 3.42
9: Update outPDF with collision step Eq. 5.7
10: **end for**

The compute kernel of the LBM for the electric potential is very cheap compared to other the LBMs. In the $D3Q19$ compute kernel, it requires only 79 floating point operations per element. However, the solution of the LBM for the electric potential is expensive since it needs to be solved until the solution converges to steady state.

**Algorithm 7** Compute kernel for the LBM for the electric potential

1: **for** each element in level-wise list of elements excluding halo elements **do**
2: Pull inPDF links from corresponding neighbor Eq. 5.13
3: Compute macroscopic potential from PDF using Eq. 3.78
4: Compute equilibrium function using Eq. 3.73
5: Update outPDF with collision step Eq. 5.7
6: **end for**

In general, the stream and the collide step can be performed in any order. Nevertheless, depending on the order, the PDF at the end of the compute kernel is either pre-collision or post-collision PDF. The conserved moments can be derived from both PDFs since they are collision invariants. However, the non-conserved moment like the deviatoric stress tensor in the

single component LBM must be computed differently depending on the pre-collision or the post-collision PDF. As mentioned in Section 3.1, the deviatoric stress tensor can be computed from the non-equilibrium PDF, $f^{neq,m} = f^m - f^{eq,m}$. This relation is valid only with the pre-collision PDF, so it can be rewritten as

$$f^{neq,m} = f^{pre,m} - f^{eq,m}. \tag{5.9}$$

Thus, the above equation is used to compute the non-equilibrium PDF when the stream step is performed at the end of the compute kernel. On the other hand, if the collide step is performed at the end of the compute kernel then the non-equilibrium function must be computed from the post-collision PDF $f^{m,c}(\boldsymbol{x}, t)$ using

$$f^{neq,m}(\boldsymbol{x}, t) = \frac{f^{m,c}(\boldsymbol{x}, t) - f^{eq,m}(\boldsymbol{x}, t)}{1 - \lambda^{\nu}}. \tag{5.10}$$

This modification in the non-equilibrium PDF can be applied to other LBM schemes. E.g. the electric field in the LBM for the electric potential can be computed using

$$f^{neq,m}(\boldsymbol{x}, t) = \frac{f^{m,c}(\boldsymbol{x}, t) - f^{eq,m}(\boldsymbol{x}, t)}{1 - \bar{\lambda}^{\gamma}}. \tag{5.11}$$

### 5.4.4 Main program

The *APES* solvers are build on top of *Aotus* and *TreElM* libraries. *Aotus* provides functionalities to load configuration files and *TreElM* manages the topology of the octree data structure. In addition, *TreElM* also provides

- Efficient reading and writing of serialized tree mesh files in parallel,
- Routines to initialize and finalize the MPI environment and supports several MPI communication patterns to exchange data between processors,
- Logging, tracking and restart infrastructure,
- Variable system to define arbitrary variables and solver independent procedural interfaces to obtain a variable,
- Space-time functions to define boundary and source variables and etc.

The space-time function and variable system are key features to couple *APES* solvers so they are explained in detail in Section 5.5.1 and Section 5.5.2 respectively.

The algorithm used in the main program of *APES* solvers is given in Alg. 8. Like every distributed MPI parallel algorithm, the MPI is initialized and finalized at start and end of the program respectively. In the tem_start routine, the MPI communicator and processor ranks are also created to communicate between processors. Additionally, the logging is also initialized. Each MPI program loads the configuration file and mesh files are loaded in chunks i.e. each processor loads only a chunk of elements distributed by splitting the serialized fluid elements list equally in each processor.

**Algorithm 8** *APES* solvers main program

▷ Initialize MPI environment and logging infrastructure. Also prints solver revision and compiler version numbers
1: tem_start()
▷ Load configuration file
2: load_configuration()
▷ Load mesh files from *Seeder*
3: load_mesh()
▷ Build solver specific data structures and initialize state arrays
4: initialize_solver()
▷ Do main time loop computation: Set boundary, compute kernel, apply source and write outputs
5: do_time_loop()
▷ Finalize outputs and dump runtime measurements of each step
6: finalize_solver()
▷ Finalize MPI
7: tem_finalize()

In the initialize_solver, step Alg. 8.4, the loaded level independent serialized list of elements is converted into a solver specific level-wise list of elements. Each solver creates this level-wise list according to its requirements. In *Musubi*, the direct neighbors are identified according to the stencil definition and the connectivity array is created. These direct neighbors define the horizontal relationship between elements and for multi-level meshes, at the level interface the vertical relationship between them are also created. Here, the virtual ghost elements are introduced to exchange values at interfaces. Two types of ghost elements are used:

ghostFromFiner and ghostFromCoarser. ghostFromFiner is used to fill coarser ghost element by taking average of values on finer elements. However for ghostFromCoarser, the interpolation order depends on the number of coarser elements being used. The availability of the number of coarser elements depends on the location of the interface. For example: at a level interface away from boundaries, coarser elements in all neighbor direction can be found and its possible that some of those neighbors might be ghostFromFiner. However, for the level interface at the boundary some coarser elements will be missing which will result in a lower interpolation order. In addition to the relationship between elements, the communication buffer to communicate between partitions is also created. The variable system that provides the method to obtain a variable is created depending on the chosen physical scheme. The variables in the variable system are classified into four types: state, derived, operation and space-time function variables. The state and derived are solver specific variables. In LBM, the PDF are defined as state variables and all macroscopic quantities that are derived from PDF are defined as derived variables. The boundaries and source variables are defined in configuration file as space-time function (Section 5.5.2). The operation variables are usually the user-defined variables in the configuration file. The coupling between *APES* solvers is realized via the variable system Section 5.5.1 and the space-time functions Section 5.5.2. To write simulation output, tracking and restart features are also initialized in this step. Finally, the flow is initialized with equilibrium function with given initial condition according to the chosen physical system.

At every time step, the following steps are performed in do_time_loop step Alg. 8.5:

**Swap state array index:** of the double buffer used for state array to read and write

**Update relaxation parameters:** when relaxation parameter is depending on time

**Set boundary:** update post-collision PDF according to boundary condition

**Compute kernel:** perform streaming and collision

**Apply source:** Add source term to LBE

**Communicate:** Exchange state values between partitions

**Output:** Write tracking and restart output and also check status of simulation.

*Musubi* has two different control routines called for each time step: one for single-level mesh and one for multi-level mesh. The steps presented above is for a single-level mesh simulation and for multi-level mesh simulation, a recursive routine is used since it requires interpolation and communication between partitions and levels. The recursive algorithm used for multi-level meshes was presented in [32]. The time loop is terminated if the simulation reached maximum time or if the simulation reaches steady state or if the simulation gets non-physical values. The results of the simulation terminated due to non-physical values are discarded. To decide whether a steady state has been reached, a time average is computed for a sliding window[1] of past iterations. The size of the sliding window can be configured by the user. A steady state is assumed to be reached when the difference between the current value and the time averaged value is less than a given threshold. This threshold is another configurable parameter to be set by the user. If the deviation is larger than the threshold, then the current value replaces the oldest value in the sliding window of past iterations and it is assumed that no steady state is reached yet. If more than one variable is to be checked for the steady state then the steady state is assumed to have reached only if all the variables defined by the user have reached the steady state. After the time loop, the solver finalizes the outputs and also dumps the runtime measurement of each routine to a file which is used to study the performance of the code.

Alg. 8 is very general and it can be applied to any numerical solver. The main reason for this generalization is due to the requirements of the coupling tool *APESmate* that is build on top of the *APES* solvers. It utilizes routines from solvers to load solver configuration file, mesh files, build solve data structure, initialize flow, compute time steps and write outputs. In the next section, the performance of both single component and multicomponent LBM-BGK on periodic domain and with spacer geometry are presented. The performance results of *Musubi* are published in the yearly HLRS review proceedings [64, 65].

### 5.4.5 Performance

In this section, the performance of *Musubi* on the cray XE6 system Hermit at HLRS is described. Hermit system has 3552 computing nodes. Each node with two AMD Interlagos sockets with 16 cores each resulting 32 cores per node. Up to 1024 computing nodes i.e. 32768 cores were used for performance analysis. For this analysis, only MPI parallelism is considered.

[1] an array of given size whose values are updated using sliding window algorithm, i.e. modulo operator is used to get the position to store the current value.

Two different test cases were chosen: the analytical setup with a fully periodic domain and the channel with complex spacer structure. A periodic domain is well suited for scaling analysis since each processor ends up in similar domain with the same communication to computation ratio. The channel with spacer structure resembles the actual production run for dilute and concentrate channels in the electrodialysis stack.

Generally, the performance of a numerical code on the machine is measured in terms of floating point operations per second i.e GFLOPS (Giga floating point operation per second). However, Lattice Boltzmann codes are measured in terms of number of lattice updates per second. Since the number of floating point operations per lattice is fixed, it is straightforward to convert it into FLOPS. In the following, Lattice Update Per second (LUPS) is used as a measurement unit as it directly provides information about how long a simulation with a certain number of lattices takes with a certain number of cores.

To get overall information, the behavior of the code is presented in terms of LUPS per node as shown in Figure 5.14 and Figure 5.16. An ideal parallel execution is expected to have replica of serial execution. However, the variation of computation to communication ratio and the cache-usage affects the performance per core. In addition to a performance map, an exclusive plot for strong and weak scaling are also used to present the scalability of the code up to 1024 nodes on the Hermit system. In strong scaling, the total problem size is fixed and distributed on different numbers of cores i.e. with an increase in the number of cores, the problem size per core is reduced. Alternately, weak scaling is measured by fixing the problem size per core resulting in the same computational load per core. Both strong and weak scaling can also be explained using the performance map Figure 5.14 and Figure 5.16 i.e. strong scaling by connecting values along a horizontal axis and weak scaling along vertical axis. Thus, the closer the measurements along a vertical axis in the performance map, the better is the weak scaling.

The performance of *Musubi* is measured for both single component and multicomponent (with three components) LBM. In both schemes, the BGK-collision operator with $D3Q19$ stencil layout is used. Stream and collide steps are solved at each time step for both models. However, the multicomponent model requires additional operations due to variable transformation (App. A.1.2.4) which results in a linear system of equations of size $N$-species. The single component model requires 169 floating point operations per lattice while the multicomponent model for three components requires 850 floating point operations per lattice i.e 280 floating point operations per lattice per component. Note that the number of

floating point operations per component for multicomponent does not increase linearly with the number of components. Nevertheless, the memory consumption and MPI communication data increases lineally with N-components. In *Musubi*, the amount of communication data is reduced through communicating only the required links of the PDF $f^m$. This reduces the communication time driven by MPI buffer size and bandwidth.

**Periodic domain** A fully periodic cubic simulation is chosen as a test case because when distributed each processor results in exactly the same problem size. This test case is well suited for weak scaling analysis. Every refinement results in an increase in the number of elements by a factor of 8 due to doubling of elements in each direction. Here, the problem size is varied from 64 elements up to 1 billion elements. Figure 5.14 shows the internode performance of single component LBM and multicomponent LBM with three components. For comparison of both models, the measurement is shown in GFLOPS per node. The performance of multicomponent is higher than the single component due to additional floating point operation per lattice in the multicomponent model which results in efficient usage of machines operation per cycle. A sustained performance of 7.2% is achieved with multicomponent LBM and 4.2% with single component LBM on the Hermit system, which has a theoretical peak performance of 294.4 GLOPS per node. This reveals that additional floating point operations per lattice in the multicomponent model results in better utilization of the Interlagos processors.

In Figure 5.14, the performance is lower for the small problem size due to increase in communication to computational ratio. As the problem size increases i.e. the ratio of communication to computation decreases, the performance increases. A peak in performance is observed when the problem size fits into cache. Beyond this cache region, the performance flattens out for the single component LBM, irrespective of increase in problem size per node resulting in better weak scaling. But for the multicomponent LBM, it does not flatten, it still increases.

Dedicated strong and weak scaling plots for both models are depicted in Figure 5.15. In strong scaling Figure 5.15a, the overall problem size is fixed to 16777216 elements. This problem is distributed on 2, 16, 128 and 1024 nodes. The parallel efficiency decreases for both models due to the increase in communication to computational over load. For single component, the efficiency is slightly increased from node 16 to 128 as the problem size fits into the cache and it decreases again for 1024 nodes. For multi component model to fit into cache for this problem size, it requires

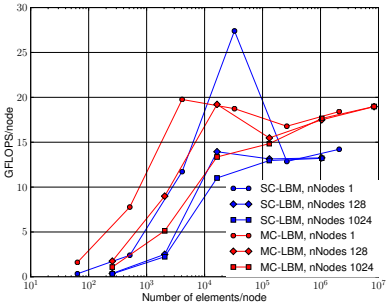
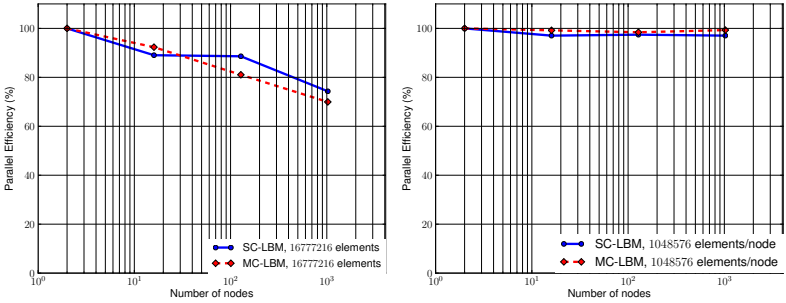



**Figure 5.14** Performance map of the single component and multicomponent LBM (three components) on periodic domain

more than 1024 nodes. Regardless, the parallel efficiency of both models is above 70%. The weak scaling Figure 5.15b is plotted for 1048576 elements per node with data points corresponding to 2, 16, 128 and 1024 nodes. Both models exhibit perfect weak scaling and parallel efficiency is above 98%.




(a) Strong scaling (b) Weak scaling

**Figure 5.15** Strong and weak scaling of single component and multicomponent LBM (three components) on periodic domain

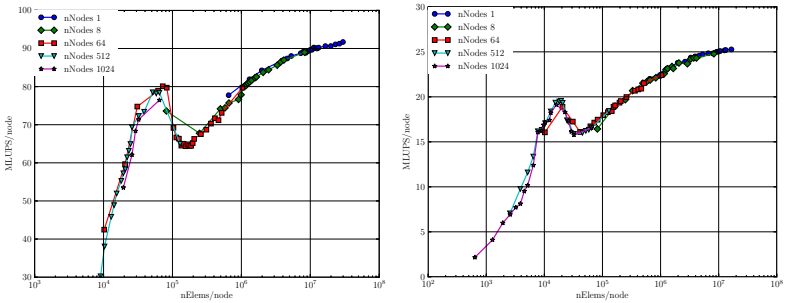
**Complex spacer structure** Now the actual production run test case i.e flow channel with spacer structure is considered for performance analysis. The purpose of this analysis is to determine the optimal point of operation for production runs i.e. to estimate the computation time required for a fixed problem size using a certain number of nodes. The same simulation

setup illustrated in Figure 5.8 with interwoven spacer structure for *Seeder* performance is also used here. The fluid flows horizontally along the spacer filaments between the two membranes on top and bottom. The spacer structure and the two membranes are treated with the simple bounce back no-slip BC. In the following, a single spacer element $L = W = 0.2$cm is discretized with $nH_{ch} = 32$ and no q-values resulting in 660 thousand elements. For scaling analysis, this single spacer element is replicated up to the total length of the actual laboratory spacer of 20cm used by AVT, RWTH Aachen. The full length of the spacer has a problem size of 66 million elements. The periodic boundary is assumed along the width of the channel. This setup covers all almost all relevant production settings; hence most effects that influence the production performance are included in this analysis.

Due to the SFC, solvers in the *APES* framework achieve an almost perfect distribution of the computation of load with at most a difference of one in the element count between two partitions. In the previous test case of fully periodic domain, the problem size was distributed perfectly. However, with the spacer structure which results in a large number of boundary elements, the communication surface between two partitions might vary drastically, resulting in large imbalances of communication costs. Additionally, computational imbalances are caused by the treatment of inlet and outlet boundary elements. It is not easy to resolve imbalances due to communication costs but computational load imbalance can be overcome by deploying a dynamic load balancing algorithm. The effect of dynamic load balancing on the performance of this test case is presented in the later section.

Figure 5.16 shows the performance of single component LBM on left and multicomponent LBM with three components on the right for the channel with spacer structure. Both models exhibit similar profiles except for the absolute value of performance. It can be seen that for the single component LBM roughly 30 million elements fit on a single node where as for multi component LBM with three components this factor is reduced to 15 million elements. Similar to the periodic test case, a cache region is observed at $8 \times 10^4$ and $2 \times 10^4$ elements per none for the single component and the multicomponent models respectively. A steep slope that is observed at out of cache region in both models might be due to load imbalance in communication which arises from the complex spacer structure. This slope affects the strong scaling of the code while the weak scaling is almost perfect.

Once again, a dedicated strong and weak scaling plot of both models with spacer structure are shown in Figure 5.17. Strong scaling Figure



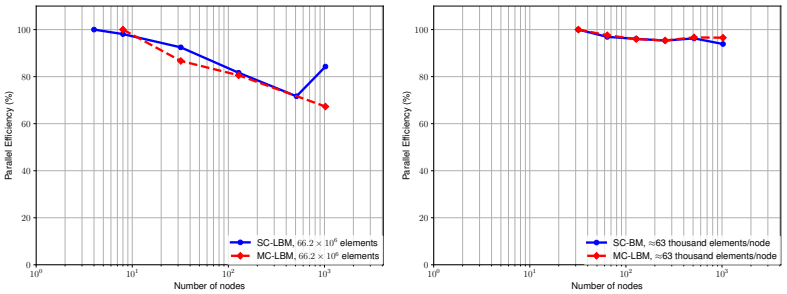

(a) Single component LBM

(b) Multicomponent LBM with three species

**Figure 5.16** Performance map of single component and multicomponent LBM (three components) with laboratory spacer structure

5.17a is shown for a total problem size of 66.2 million elements that covers the full length of the laboratory spacer. The minimum number of nodes required to fit this problem size of single component and multicomponent are 4 and 8 nodes respectively. This shows that the multicomponent model requires roughly twice the number of nodes than the single component model to fit the same problem size. The performance increase in the single component case for 1024 nodes due to cache effects as the problem size is getting small enough to fit into cache. For both models, the performance drops due to communication overload. Overall, the code shows good strong scaling with parallel efficiency above 70% on 1024 nodes, i.e. 32768 cores. Weak scaling for a problem size of approximately 63 thousand elements per node is shown in Figure 5.17b. As can be seen, the weak scaling is almost perfect for both models with only a small drop in the parallel efficiency for larger counts of compute nodes.

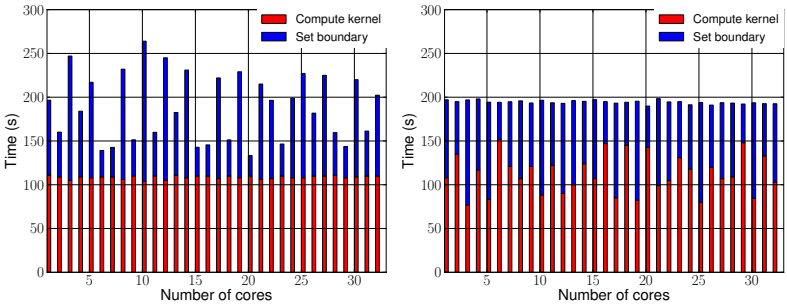
**Dynamic Load Balancing** In multicomponent flow simulation with spacer structure, the computational load imbalance arises from fluid element that require boundary treatment like inlet, outlet, spacer wall and membrane. These imbalances are resolved by load balancing algorithm. In *Musubi*, the space filling curve partitioning algorithm (SPartA) [29] is deployed to handle the computational cost associated with boundary treatment. For load balancing, the weights are provided for each element. In our application, the weights per element are computed from the computational time spent on compute kernel and boundary treatment. SPartA uses these




(a) Strong scaling (b) Weak scaling

**Figure 5.17** Strong and weak scaling of single component and multicomponent LBM (three components) with laboratory spacer structure

weights to compute the optimal split position such that weights are equally distributed among the processors. To ensure optimal load balancing, this algorithm is applied dynamically.

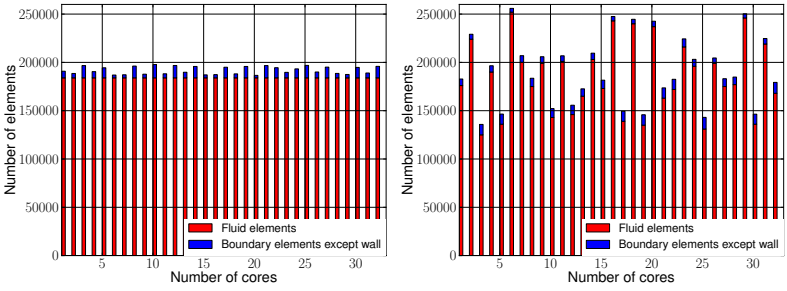



(a) Before load balancing (b) After load balancing

**Figure 5.18** Time spent on compute kernel (red) and set boundary (blue) routine for 1000 time steps with a total problem size of 5 million elements, distributed on 32 cores

Figure 5.18 shows the run time of "compute kernel" and "set boundary" routines for 1000 time steps with a total problem size of 5 million elements with spacer structure, distributed on 32 cores. Due to initial distribution, almost perfect computational load is obtained for compute kernel but the boundary elements cause load imbalances that can be seen in Figure 5.18a. After load balancing, Figure 5.18b, the elements are redistributed and a

perfect balancing is achieved. The element distribution before and after load balancing is shown in Figure 5.19.




(a) Before load balance

(b) After load balance

**Figure 5.19** Elements distribution before and after load balancing. Red represents fluid elements and blue presents fluid elements with boundary treatment except wall. Here, q-Values are not considered for spacer.

## 5.5 APESmate - Integrated coupling tool

As previously mentioned, the ED process is the multiphysics heterogeneous system, and to simulate this system the involved different physics must be coupled. The theoretical concept to couple this multi-physics system was introduced in Chapter 4. As shown in Figure 4.2, this system involves both surface and volume coupling. The spacer-filled dilute and concentrate flow channels are coupled with the membrane black-box model via surface coupling. Both dilute and concentrate flow channels are coupled with electric potential via volume coupling. In the previous Section 5.4, the highly scalable lattice Boltzmann solver *Musubi* to simulate single physics was introduced. In this section, the coupling tool *APESmate* developed to couple solvers in the *APES* framework is introduced. At first, the coupling challenges and certain requirements of the coupling tool are briefly discussed. Then, the implementation details of this tool are presented in detail.

Just like the ED process, most real world applications involve multi-physics or multi-scales or both which are not feasible to compute using traditional single physics solvers. One way to efficiently simulate these applications are to split the domain into several sub-domains according to their physics or length scales and couple them with each other via

surface or volume. The coupling between sub-domains can be achieved by exchanging variables at the coupling interfaces (surface or volume) as point values. These spatial points on the coupling interface are called coupling points and the variables evaluated on those points in the remote domain are referred to as coupling variables. In this work, the term 'local domain' and 'remote domain' refer to the domain that requires and provides coupling variable values respectively. The *Musubi* solver presented in the previous section is highly scalable on its own so the coupling tool must be scalable as well to simulate large coupled simulations. Therefore, the challenge here is the efficient coupling of the scalable solvers such that scalability and numerical accuracy of the solvers are maintained in the coupled simulation. In addition to these coupling challenges, there are certain requirements which need to be satisfied by the efficient coupling tool such as

- Mesh independent data (coupling variable) exchange in space and time so each domain can have different resolution, time scale and scheme order i.e.
  - Interpolation in space, at coinciding or non-coinciding points.
  - Interpolation in time, for non-coinciding time intervals i.e. when each domain have different time steps.
- Coordinated execution of involved solvers.
- Minimal access to solver specific data structures and its routines.
- Exchange coupling variables between domains in parallel.
- Define the distribution of domains on certain number of processes as configurable parameter.

The sole purpose of the coupling tool is to exchange values between the domains at the interface. Most coupling tools do that through treating solvers as black box. The solver of the local domain provides the coupling variables on a certain list of points to the coupling tool and the coupling tool interpolates the variable values provided by the remote domain on the points requested by the local domain. The advantage of such a tool is that any solver can be coupled but the disadvantage is that numerical accuracy of the interpolated values depend on the interpolation order and number of points available for interpolation. Thus, it reduces numerical accuracy and scalability of the code. *APESmate* is an integrated coupling tool developed to satisfy these challenges and most of the fore-mentioned requirements. It is implemented as part of *APES* framework to couple

*APES* solvers like *Ateles* and *Musubi*. The current limitation of *APESmate* is that it can be used to couple only solvers in the *APES* framework as it relies on the *TreElM* data structures. On the other hand, the integrated coupling approach serves its purpose by improving the numerical accuracy since the variables are evaluated directly by the solver for a given set of points resulting in an accuracy order which is the same as the order of the solver.

As shown in the *APES* schematic layout, *TreElM* is the central library providing the octree data structure. The mesh generator, *Seeder* generates the octree mesh and the SFC linearizes elements in the mesh. The linearization of fluid elements in the mesh results in a level independent list of fluid elements. As previously stated in Section 5.4.2 The generated linearized mesh is dumped to disk which the solver reads and create a level-wise list of elements to vectorize the computations. The data structure used by the solver to store the level-wise list of elements is referred to as the solver specific data structure. Then the state array to store the state variables of the equation system is constructed from this solver specific data structure. The lattice Boltzmann solver *Musubi* is stencil based so the state values per element are stored according to the stencil definition. These state values are cell-centered values so the values at the off-centered points are obtained by interpolation. On the other hand, the Discontinuous Galerkin (DG) solver *Ateles* is the high-order scheme with polynomial representation for the state values. Here, the state values stored per element are polynomial coefficients. Therefore, the values on points are obtained by polynomial evaluation. Thus, to couple *APES* solvers with different solver internal data structures, the concept of variable system (*varSys*) was introduced (see Section 5.5.1). The *varSys* provides generic interfaces to extract variables for the list of elements or points. The space-time function (*STfun*) is a type of variable in the varSys that is used to define BC and source term in solvers. In *APES*, the surface and volume coupling are treated as BC and source term respectively. Since BC and source term are defined as *STfun* variable, the coupling data (points, variable names, variable values, etc,) to exchange between domains are stored explicitly in *STfun*. *APESmate* can access the data in the *STfun* through the *varSys* in the local domain. Using these data, *APESmate* can evaluate the variables on the coupling points using the *varSys* in the remote domain. The flow of data between domains through *APESmate* is illustrated in Figure 5.20. In the figure, *APESmate* takes the coupling points and the variable names from *STfun* in 'domain 1', evaluates the variables on the coupling points using *varSys* in 'domain 2' and then it stores the evaluated variable values back to the *STfun* in 'domain 1'.

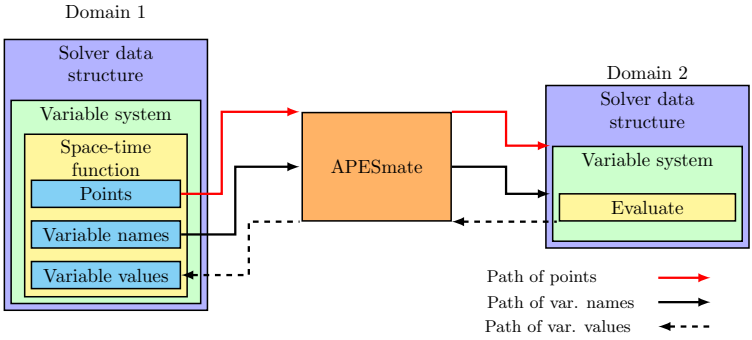



**Figure 5.20** Illustration of exchange of points, variable names and variable values between domain 1 and 2 through *APESmate*

In short, *APESmate* only needs to access the *varSys* and *STfun* inside the *varSys* to exchange data between domains. Therefore, these two core features: *varSys* and *STfun* are implemented in the central library *TreElM*. In the next sections, the implementation details of the two features are explained in detail.

### 5.5.1 Variable system

The *varSys* is a feature in *TreElM* that provides different methods to evaluate variables in the solver. This feature allows the user to define arbitrary variables and their relations in the configuration file. Its main purpose is to provide an easy access to evaluate variables for post-processing. Since the interface of the functional procedure used to evaluate variables is independent of the solver data structure, this feature is also best suited to couple different solvers.

In *varSys*, the variables are classified into: *state*, *derived*, *operation*, *STfun* and *anonymous*. In the following, the definition of these variables and how they are added to the *varSys* are explained.

***State***: These variables are variables in the equation system on which the computations are carried out in the numerical solvers. Therefore, the *state* variables are allocated and stored in memory. Other variables of interest are usually derived from the *state* variables. The name and the number of components of the *state* variable depend on the type of the equation system which is being solved. So, these variable

are predefined in the solver for each equation system. E.g. in LBM, the *state* variable is the PDF $f^m$ with $Q$ components.

***Derived*:** These variables are derived from another variable. In solver, most *derived* variables are derived from the *state* variables. The list of *derived* variables depends on the type of the equation system. Therefore, these variables are also predefined in the solver for each equation system. E.g. in LBM for fluid flow, the macroscopic quantities: *density* ($\rho$), *pressure* ($P$), *velocity* ($\boldsymbol{v}$) are derived from the PDF but the*shear stress* ($\boldsymbol{\sigma}'$) is derived from the $\boldsymbol{f}^{neq}$. Likewise, in multi-component LBM, the species macroscopic quantities: *spc_density* ($\rho_k$), *spc_velocity* ($\mathbf{v}_k$) are derived from species PDF $f_k^m$ and mixture variables are then derived from their respective species variables. In LBM for electric potential, the macroscopic quantity: *potential* ($\psi$) and *electric-field* ($\boldsymbol{E}$) are derived directly from the PDF $f^m$.

***Operation*:** These variables are derived by applying a certain mathematical operation on another variable. They are referred to as *operation* variables. E.g. *momentum* ($\boldsymbol{p}$) can be computed by multiplying *density* and *velocity* as $\boldsymbol{p} = \rho\boldsymbol{v}$ and *vorticity* ($\boldsymbol{\omega}$) can be computed from curl of *velocity* as ($\boldsymbol{\omega} = \nabla \times \boldsymbol{v}$). Almost all basic mathematical operations are supported: basic arithmetic operations ($+, -, *, /$), logical operations ($>, <, \geq, \leq, =, /=, .and., .or.$) and vector operations (magnitude, gradient). These variables are usually defined in the configuration file to obtain arbitrary variable for post-processing/visualization. Example for *operation* variable definition in configuration file is given in App. Lst: A.1.

***STfun*:** These variables are defined as space-time function in the configuration file. They are especially used to define boundary and source variables. Since this variable is used for coupling solvers, they are explained in detail in the next section. An example configuration of the *STfun* variable in the variable table is given in Lst: A.2.

***Anonymous*:** This variable is a specific form of the *STfun* variable. It refers to a boundary/source variable which is directly defined as the *STfun* instead of referring to a variable name defined in the variable table. In the example configuration given in App. Lst: A.3, the boundary and the source variables are defined as the *anonymous* variable.

In summary, there are five variable types: *state*, *derived*, *operation*, *STfun* and *anonymous* variables. They can be grouped into two category

as: dependent and independent variables. The *state*, the *STfun* and the *anonymous* variables are independent variables because they are self-defined whereas the *derived* and the *operation* variables are dependent variables because they are derived either from another variable.

In addition to the aforementioned mathematical operations, there are two additional operations introduced especially for coupling: "combine" and "extract". The "combine" operation gathers multiple variables into a single variable and the "extract" operation extracts a certain number of components from a vector variable. Often in surface/volume coupling, more than one variable is required to be exchanged between domains and they have to be evaluated on the same set of points so it's efficient to combine all those variables into one and exchange just one variable with a single MPI communication. Thus, if the local domain requests multiple variables then *APESmate* combines these variables into a single variable and append it to the *varSys* in the remote domain. This variable in the remote domain is then evaluated at every synchronization time step by *APESmate* and the results are stored in the local domain where individual variables are extracted. An example is given in App. Lst: A.4, where 'pressure' and 'velocity' in *STfun* variable 'surface_coupling' is combined into a single variable by *APESmate* and evaluated in the remote domain 'dom_1' and the results are stored in the 'surface_coupling' variable in the local domain. These results in the 'surface_coupling' variable are then extracted using 'vel_d1' and 'press_d1' variable and assigned to the boundary variable 'velocity' and 'pressure' respectively.

In *tem_varSys_type* (Lst: 5.1), *varname* is declared as 'dynamic array'[1] of label (character(len=80)) to avoid duplication of variables in the *varSys* and *method* is declared as 'growing array'[2] of *tem_varSys_op_type* (Lst: 5.2) which contains information on how to obtain a variable. Since *varname* is a dynamic array, it is possible to find the index position of a requested variable name using binary search and this index position is then used to access the *method* of that requested variable. In both Lst: 5.1 and Lst: 5.2, only the declarations that are relevant for this discussion are given.

```
1 type tem_varSys_type
2 ...
3     !> List of variables in the system.
4     type(dyn_labelArray_type) :: varname
5
6     !> Definition of how to obtain a variable.
7     type(grw_varOpArray_type) :: method
8 end type tem_varSys_type
```

[1] contains an array with unique entries which grows over time and a sorted index array for fast lookups of a given value using a binary search, see Section 5.3.

[2] contains an array which grows over time

**Listing 5.1** Description of the variable system

```
type tem_varSys_op_type
    !> Number of components for this variable.
    integer :: nComponents

    !> Number of variables, that are needed as input to obtain this variable.
    integer :: nInputs

    !> Position of the input variables in the variable system.
    !! There are as many entries as nInputs.
    integer, allocatable :: input_varPos(:)

    !> Data that is required by the get method.
    type(c_ptr) :: method_data

    !> Function to actually obtain the variable at a given point.
    !!
    !! This is either a function accessing a state variable directly,
    !! a function returning a space time function evaluation or a
    !! derived quantity, that computes a new variable out of others.
    procedure(tem_varSys_proc_point), pointer :: get_point => null()

    !> Function to setup points set for boundaries and sources.
    !! Pointe set are stored in method_data level wise 1D growing array for
    !! dimension X,Y and Z.
    !!  * For solver variables, points are stored in solver container.
    !!  * For space time variables, points are stored in space time function.
    !!  * For operation variables, points are passed down to its input variable.
    procedure(tem_varSys_proc_setupIndices), pointer :: setup_indices => null()

    !> Function to get value for point set stored in method_data for requested
    !! index in point set. This function either returns a pre stored value or
    !! compute value depends on variable type and space time function.
    !! For time independent space time function, values are computed
    !! in setupIndices and growing array of points are deleted
    procedure(tem_varSys_proc_getValOfIndex), pointer :: get_valOfIndex => null()

end type tem_varSys_op_type
```

**Listing 5.2** Description of the method on how to obtain a variable

In order to keep the interface of functional procedures in *tem_varSys_op_type* (Lst: 5.2) independent of the solvers, a *method_data* c_ptr is used. For *state* and *derived* variables in the solvers, the *method_data* stores the address of the solver container. The solver container is the data type with pointers to all required data types in the solver to obtain a variable. The *method_data* of the *STfun* variable refers to an element in the linked list data structure (*tem_st_fun_listElem_type*) and the *operation* variable refers to the operation data structure. With the *method_data*, each variable can be implicitly obtained.

The *varSys* provides two ways to obtain a variable value: Either by single-stage direct evaluation using *get_point* or by a two-stage approach

using *setup_indices* and *get_valOfIndex*. *get_point* is the function pointer of the *tem_varSys_proc_point* procedure and it returns variable values at a given set of points $\boldsymbol{x}_i$, where $i$ is the number of points requested. For *state* variable, this function directly access the state array. In the *Musubi* solver, the state values are cell averaged values and if the requested point is not a barycenter of the element then a value is computed by weighted average of neighbors using

$$\varphi_i(\boldsymbol{x}_i, t) = \sum_m w_i^m \varphi(\boldsymbol{x} + \boldsymbol{u}^m, t), \tag{5.12}$$

where $w_i^m \in [0, 1]$ are the weights computed by

$$w_i^m = \frac{d^m}{\sum_m d^m}, \quad \text{where } d^m = |\boldsymbol{x}_b^m - \boldsymbol{x}_i|, \tag{5.13}$$

$d^m$ is the distance between the requested point $\boldsymbol{x}_i$ and the barycenter of an element $\boldsymbol{x}_b^m$ along the direction vector $\boldsymbol{u}^m$. On the other hand, in the *Ateles* solver, the state values are polynomial coefficients of the given order. Therefore, a value is obtained by the evaluation of polynomial on the requested point. *get_point* of a *STfun* variable evaluates the *STfun* at the given set of points $\boldsymbol{x}_i$. The *derived* or *operation* variable calls their dependent variable *get_point* using *input_varPos* and then derives or applies the operation on the dependent variable output respectively. Thus, *get_point* is a direct approach to obtain the variable at a given point.

The mesh independent data exchange between domains is achieved by exchanging coupling variables at points in space. The variable at a certain point can be directly obtained by *get_point* routine. However, in the solver the state variables are stored element wise and it is expensive to compute the element location in the $\mathfrak{tJD}$ list, which contains the requested point at every time step. To resolve this, a two-stage approach was introduced. In this approach, the points and the element location in the state array are stored in the variable *method_data* during initialization. At each time step, variable values are obtained using the index of points stored in the *method_data*. This approach was mainly introduced to allow coupling between domains via variables in the *varSys*. As mentioned earlier, the surface and the volume coupling are carried out through boundary condition and source terms respectively. The coupling points required to evaluate the coupling variable in the remote domain are generated from the boundary or source in the local domain and stored in their respective boundary or source variable *method_data*. As stated before, the boundary and source variables are defined as *STfun*, so the coupling tool *APESmate*

needs to access only the *STfun* variable *method_data* to get the coupling information. One of the goals for the design of the coupling tool *APESmate* (Section 5.5) is to have minimum access necessity to each solver and it is achieved using *varSys* and *STfun* in the solver.

In the following, the two-stage evaluation that is used for coupling in explained. At first, the points are generated in boundary/source and they are send to a variable via the *setup_indices* routine and it appends the points to a growing array of points within in a variable *method_data*. It then returns the index positions of the stored points, which are then stored in the corresponding boundary/source data structure. Finally, the indices are stored in boundaries/sources and they are used to obtain a variable at every time step using the *get_valOfIndex* routine. Here, to avoid appending a same point twice in the growing array of points, a point is converted into a $\mathfrak{tID}$ on the maximum refinement level ($\mathfrak{L}_{max} = 20$) using

```
1  ! location of point in bounding cube
2  locInCube = point  bounding_cube%origin
3  ! Number of elements in maximum level
4  dimLen = 2**level
5  meshDensity = real(dimLen) / bounding_cube%length
6  ! coordinate of point in given level
7  coord(1:3) = max( min( int(locInCube*meshDensity), dimLen1), 0)
8  ! Start with first treeid on given level
9  treeID = firstIDAtlevel(level)
10 fak8 = 1
11 fak2 = 1
12 ! Find the treeID by summing treeID of its parent
13 do iLevel = 0, level-1
14   treeID = treeID + fak8 * ( mod( coord(1)/fak2, 2 ) + 2 * mod( coord(2)/fak2, 2 ) &
15     &    + 4* mod( coord(3)/fak2, 2 ) )
16   fak2 = fak2 * 2
17   fak8 = fak8 * 2
18 end do
```

and this $\mathfrak{tID}$ is added to a dynamic array of $\mathfrak{tID}$s to maintain a unique growing array of points. Thus, if the same point is sent to *state/STfun* variable by another boundary/source then *setup_indices* does not append a point instead it will return the index of provided point in a growing array of points. The points are stored only in the *method_data* of *state* and *STfun* variables and the index are stored in corresponding boundary/source. For the dependent variables: *derived* and *operation*, the points are passed down to the *state/STfun* variables from which they are derived and store only their required index.

Figure 5.21 illustrates the exchange of points and index between the variable and the boundary for the example configuration in Lst: A.4. The points from boundary are sent to a *STfun* variable 'surface_coupling' through variables 'vel_d1' and 'press_d1'. Let us assume that the points

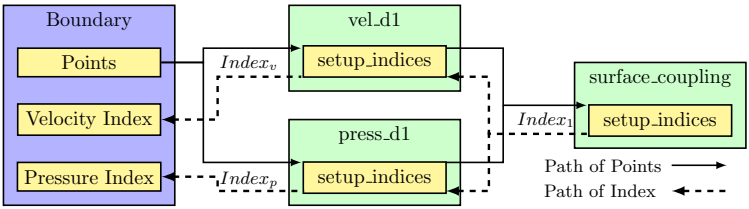



**Figure 5.21** Illustration of exchange of points and index between variables and boundary/source

are sent to 'surface_coupling' variable through 'vel_d1' variable. The points are then stored in the *method_data* of the 'surface_coupling' variable and returns the indices of the stored points, i.e. $Index_1$ in Figure 5.21. When the same points are sent to the 'surface_coupling' variable by the 'press_d1' variable, it returns the same index $Index_1$ without appending the points. This $Index_1$ is then stored in *method_data* of 'vel_d1' and 'press_d1' and they both create their own new index called $Index_v$ and $Index_p$ respectively. These indices are then stored in the corresponding boundary variables: *velocity* and *pressure*. The exchange of coupling points between domains is elaborated in Section 5.5.3.3.

### 5.5.2 Space-time function

As mentioned earlier, the *STfun* is required to define macroscopic variables at boundaries and sources in the configuration file. For example: In the fluid flow of the LBM solver, *velocity* ($\boldsymbol{v}$) and *pressure* ($P$) are defined for inflow and outflow boundaries respectively, and *force* ($\boldsymbol{F}$) is defined as source term. In general, a *STfun* variable $\boldsymbol{\varphi}$ at boundary/source is defined as function of space and time $\boldsymbol{\varphi}(\boldsymbol{x}, t)$, where $\boldsymbol{x}$ is the physical co-ordinate point or position vector and $t$ is the physical simulation time. The variable $\boldsymbol{\varphi}$ can be a scalar or vector depending on the requested physical variable at the boundary/source. The *STfun* data type definition in *TreElM* allows to define a variable to be an arbitrary definition of STfun i.e. a variable definition in configuration file can be a constant or a space-time Lua function or a predefined Fortran function. There are several predefined Fortran functions implemented in *TreElM*. One of them is predefined='combined' that splits a *STfun* into spatial $\varphi_s$ and temporal $\varphi_t$ function as $\varphi(\boldsymbol{x}, t) = \varphi_s(\boldsymbol{x}) \cdot \varphi_t(t)$. An example configuration is given in Lst: A.2, in which the pressure at the outflow boundary is defined as constant, velocity at inflow boundary is defined as predefined='combined'

with temporal and spatial definitions, and force at source is defined as a Lua function.

It's worth mentioning again that the *STfun* is also a type of variable in the *varSys*. So, it is possible to define *STfun* as a variable in the configuration file and then refer to that variable name in the boundary and source variables. Example of *STfun* as a variable is given in Lst: A.2 and Lst: A.6. One advantage of defining *STfun* as a variable in the variable table is that then it can be tracked by the tracking infrastructure to debug the *STfun*. If *STfun* is defined directly as *anonymous* variable, it cannot be tracked because the name for this variable is created within the code.

In the solver, the boundary variables must be evaluated on the boundary (surface) of the computational domain and the points (spatial position vector) $\boldsymbol{x}$ on the surface are generated using the boundary information provided by *Seeder*. The source terms are usually applied on the entire volume so the source variables must be evaluated on the entire computational domain. However, in some cases source terms need to be applied only to a limited area of the computational domain and for these cases, the *shape* object is defined. An example of such a *STfun* variable with shape defining a box of length 1 in x- and y- direction is given in Lst: A.5.

As mentioned before, the coupling variables are evaluated in the remote domain on the spatial points from the local domain. Therefore, the coupling points generated by every boundary and source term in the local domain are stored in their respective *STfun* point data type *tem_pointData_list_type* object in *tem_st_fun_listElem_type* (Lst. 5.3).

```
1  !> An element for a spacetime function within a linked list.
2  !!
3  !! Besides the actual list of space time definitions that are provided, there
4  !! is a pointer to the next element in the list.
5  !! Method data of a space time function variable refers to this type.
6  type tem_st_fun_listElem_type
7
8      !> Number of space time functions
9      integer :: nVals
10
11     !> List of space time function since single variable can have multiple
12     !! space time function
13     type(tem_spacetime_fun_type), dimension(:), pointer :: val => NULL()
14
15     !> Points data containing space coordinates or evaluated
16     !! values for time indepentent functions
17     type(tem_pointData_list_type) :: pntData
18
19     !> Pointer to next space time function
20     type(tem_st_fun_listElem_type), pointer :: next => NULL()
21 end type tem_st_fun_listElem_type
```

**Listing 5.3** Description of the space-time function element data type

*APESmate* gathers the points stored in the coupling *STfun* point data type and evaluates the coupling variables at the points in the remote domain. In order to evaluate the variable in the remote domain, the points must exist in the remote domain. Unlike volume coupling where sub-domains overlap, the surface coupling has non-overlapping sub-domains so the coupling points on the surface of the local domain cannot be found on the remote domain. Therefore, to identify and evaluate a variable at the coupling points in the remote domain, the points need to be shifted by a small offset towards the remote domain. The additional offset directions are provided by the boundary routines in the local domain and stored in *tem_pointData_list_type* for every point. The calculation of the offset direction depends on the solver. For the link-based solver like *Musubi* the points must be shifted along the outgoing direction pointing towards the boundary(*offset_dir*) and for the DG solver *Ateles* the points must be shifted along the boundary face normal direction. To maintain numerical accuracy at the surface, the points are shifted only by a smallest double precision real number using the Fortran intrinsic *spacing* function. The *offset_dir* is three integers per point which is quite expensive to communicate. Therefore, to reduce the data size (memory consumption) of the MPI communication, the *offset_dir* is converted into *offset_bit* for every point. The *offset_bit* stores the *offset_dir* as character using

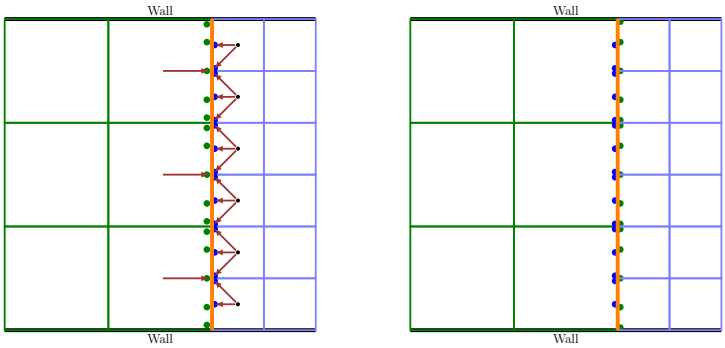
```
1 offset_bit = achar((offset_dir(1)+1) + (offset_dir(2)+1)*4 + (offset_dir(3)+1)*16)
```

The *offset_bit* can be converted back into the offset direction vector using

```
1 offset_dir(1) = mod(ichar(offset_bit),4) 1
2 offset_dir(2) = mod(ichar(offset_bit),16)/4 1
3 offset_dir(3) = ichar(offset_bit)/16 1
```

These *offset_bits* are sent to a variable through *setup_indices* by the BC and source along with the points. The offset of points on the coupling interface is illustrated using an example of a non-overlapping surface coupling interface of two domains shown in Figure 5.22. The green domain is the DG solver with spatial polynomial order 4 and the blue domain is the LBM solver. Since the DG solver requires points exactly on the face, its offset directions (red arrows) are defined along the face whereas for the LBM solver, which is link-based, the points shift along each link. Figure 5.22b shows the points after shifting along the offset direction into the remote domains. The top and the bottom are defined as wall boundaries so the LBM solver does not require points on these walls

With the *offset_bit*, the *APESmate* computes the offset direction and shift the points in the remote domain. This offset must to be done only for




(a) Points on coupling interfaces with arrows representing offset directions

(b) Points shifted along offset directions

**Figure 5.22** Example of non-overlapping coupling interface of two domain green and blue. Both domains have different resolutions and different solver. The green domain is the 4th order DG solver and Blue domain is the LBM solver.

surface coupling, so to distinguish surface coupling from volume coupling, the local domain sets an additional integer variable 'isSurface' to 0 for surface coupling or 1 for volume coupling. Since the solver treats BC and source term on a level-wise (the elements are sorted per level for multi-level meshes) list of elements, the generated points are also stored level-wise in the point data type. Therefore, the evaluated variables on the remote domain are also stored level-wise in the *cpl_value_type* object in the *tem_aps_coupling_type* (Lst: 5.4). The coupling points on the single process of the local domain might be distributed on multiple processes on the remote domain. So, during the initialization of *APESmate*, the process rank ID for each point is identified and stored in 'pntRanks' in *cpl_value_type*. This type also stores the values of the coupling variables evaluated on the remote domain and stored in the 'evalVal' array in the local domain. The 'pntRanks' helps to create the receive communication buffer 'recvBuffer' where the values are received from the remote processes and then copied to 'evalVal'. The communication buffer is the data type in *TreElM* that provides process-wise buffers to exchange data with a specific process. The communication between domains is explained in the next section.

```
1  ! type which include all exchange points information
2  type cpl_value_type
3    !> number of points per level
```

```
4       integer :: nPnts = 0
5
6       !> Global process ids to evaluate the points
7       !! It is deallocated after recvBuffer is filled
8       integer, allocatable :: pntRanks(:)
9
10      !> Evaluated variable value on each point.
11      !! If variable is time independent then values are evaluated and stored
12      !! at initialization stage, in this case point arrays are not stored.
13      !! nComp = nScalars in the tem_aps_coupling_type%varnames
14      !! Access: (iVal 1)* nComp + iComp
15      real(kind=rk), allocatable :: evalVal(:)
16
17      !> Receive communication buffer to fill evalVal
18      type(tem_communication_type) :: recvBuffer
19    end type cpl_value_type
20
21      !> Coupling description defined in config file from load space time function
22    !! which is called from load boundary condition or load sources
23    type tem_aps_coupling_type
24
25      !> Remote domain label to get data from
26      character(len=labelLen) :: rem_domLabel
27      !> Domain ID of remote domain label
28      integer :: rem_domID
29
30      !> Number of variables to get from remote domain
31      integer :: nVars
32      !> List of variables to get from domain
33      character(len=labelLen), allocatable :: varNames(:)
34
35      !> nScalars of varNames
36      !! Must be same as nComps in stFun
37      integer :: nScalars
38
39      !> Used to decided whether this space time functions are used
40      !! for surface or volume i.e boundary or source.
41      !! For boundary. isSurface = 0
42      !! For volume, isSurface = 1
43      integer :: isSurface = 1
44
45      !> store value on each level
46      type(cpl_value_type) :: valOnLvl(globalMaxLevels)
47    end type tem_aps_coupling_type
```

**Listing 5.4** Description of the coupling data type in *TreElM* to store evaluated variable

The *tem_aps_coupling_type* (Lst. 5.4) is an object in *tem_spacetime_fun_type* (Lst. 5.5), *APESmate* can access them through *method_data* of the *STfun* variable.

```
1   type tem_spacetime_fun_type
2       !> The function kind
3       !! Should be either:
4       !!    'const': Constant for all (x,y,z,t)
5       !!    'combined': This returns spatial(x,y,z)*temporal(t)
6       !!    'lua_fun': Function defined in the Lua script
7       !!    'apesmate': Get variables from remote domain
8       character(len=labelLen) :: fun_kind
9
```

```
10      !> spatial restrictions
11      type(tem_shape_type), allocatable :: geom(:)
12      \dots
13
14      !> Apesmate coupling description
15      type(tem_aps_coupling_type) :: aps_coupling
16  end type tem_spacetime_fun_type
```

**Listing 5.5** Description of the space-time function data type

In the *tem_aps_coupling_type*, 'varNames' and 'rem_domLabel' are coupling variables names to be evaluated in the remote domain and the name of remote domain respectively. These two information are provided in the configuration file as predefined Fortran function in *STfun* with predefined ='apesmate'. An example configuration of the *STfun* variable is given in Lst: A.6, where 'electric_force' is provided to source term 'force' and it needs to be computed from the remote domain name: *'potential_dom'* and the variable name in the remote domain: *'electric_field'*. The *APESmate* evaluates the coupling variables in the remote domain and stores evaluated variables in the 'evalVal' in *cpl_value_type* which is defined level-wise in *tem_aps_coupling_type* which is an object in *STfun* data type in the local domain. Then when the solver request a variable at boundary/source using *get_valOfIndex* in *varSys*, *STfun* returns the pre-stored values.

Its worth mentioning that it is possible to define multiple *STfun* for a variable in the variable table. In such cases, *evaltype* is defined in the configuration file to determine how to merge the values of multiple *STfun*. The list of spatial points from the solver and multiple *STfun* are stored as an element in the linked list type *tem_st_fun_listElem_type*. The *method_data c_ptr* (explained in the previous section) of the *STfun* variable refers to an element of this linked list. All *STfun* defined in the configuration file are added to this linked list and address the first entry in the linked list which is stored in the solver. This linked list helps *APESmate* to gather all the coupling *STfun* by looping over each entry in the list. With this design, the *APESmate* couples sub-domains independent of the solver data structure.

### 5.5.3 Coupling algorithm

*APESmate* is a single executable, integrated coupling tool build as part of *APES* framework. It utilizes existing solver routines to load solver configuration file, load mesh files, initialize solver specific data structures, initialize flow, do computation and write outputs. *APESmate* distributes the sub-domains across processes by creating individual MPI sub-communicators

for each domain. As mentioned before, it evaluates the coupling variable in the remote domain using the remote domain's *varSys* and stores the evaluated variables in the *STfun* in the local domain along with the other coupling information to exchange with the remote domain. The *APESmate* uses a global MPI communicator to exchange variables between the domains and the MPI sub-communicator to communicate within the domain. Thus, it is not altering the underlying communication patterns used in the solvers. Here, the domain refers to the physical system and solver is the numerical scheme chosen to simulate that physical system.

The algorithm used in *APESmate* is given below Alg. 9. The basic structure of this algorithm is similar to the solver Alg. 8 with few additional steps like loading *APESmate* configuration file, *domain_distribution*, *initialize_coupling*, *synchronize_domains*, *update_nextSyncTime* and *finalize_coupling*. Some of these steps are explained in detail below. Similar to the solver, the MPI communicator for *APESmate* is also initialized in *tem_start* and finalized in *tem_finalize*. In the *load_configuration* step, the *APESmate* configuration file is loaded with information about each domain and the simulation termination conditions. After that in the *domain_distribution* step, the domains are distributed across the processes and MPI sub-communicators are created for each domain. Next, the configuration file for each domain is loaded and the domain initialized to build solver specific data structures. In the *initialize_domain* step, the spatial points are generated by boundaries and source terms and stored in their respective *STfun* variables and the *state* variables are initialized with the initial conditions. In the *initialize_coupling* step, the *APESmate* gathers the coupling data (points, variable names, remote domain label) from the *STfun* variables. In this step, the rank ID of the remote domain for each point are identified using the round-robin fashion and then the points are sent to the correct process of the remote domain. Additionally, the *APESmate* checks for the availability of the variable names requested by the local domain in the remote domain *varSys* and then combines those variables into a single variable and appends it to the remote domain *varSys* as a 'combine' variable. Furthermore, the communication buffers to exchange evaluated variables between the domains are also build in the *initialize_coupling* step. Before the time loop, the domains are synchronized with initial values from the domains in the *synchronize_domains* and the next synchronization time is initialized in the *initialize_nextSyncTime*. In the synchronization step, coupling variables are evaluated on the remote domain and exchanged to the local domain. Next is the time loop in which *APESmate* steers each domain by solving each sub-domain until it reaches the next synchronization time. At every synchronization time, the coupling

variables are evaluated and exchanged and the next synchronization time is computed. After the time loop, each domain is finalized with writing the simulation outputs and runtime measurements to disk. At last, before finalizing the MPI, the runtime measurements of *APESmate* are written to the disk. In the following, each step in the coupling Alg. 9 is explained.

**Algorithm 9** *APESmate* Algorithm

▷ Initialize MPI environment and logging
tem_start()
▷ Load *APESmate* configuration file
load_configuration()
▷ Distribute domains according to procWeight per domain and create MPI sub-communicator for each domain
domain_distribution()
**for** each domain in local process **do**
▷ Load each domain configuration file and mesh files
    load_domain_config()
▷ Build solver specific data structures and initialize state arrays for each domain
    initialize_domain()
**end for**
▷ Build coupling data structure
initialize_coupling()
▷ Synchronize domains by evaluating coupling variable in remote domain and communicate to local domain
synchronize_domains()
▷ Compute first synchronization time by computing maximum of domain time steps
Initialize_nextSyncTime()
▷ Do main loop computation: solve each domain and synchronize domains
**for** $iTime \leftarrow 0, tMax$ **do**
▷ Solve each domain till next synchronization time or until steady state
    **for** each domain in local process **do**
        do_time_loop_domain()
    **end for**
▷ Synchronize domains by evaluating coupling variable in remote domain and communicate to local domain
    synchronize_domains()
▷ Update next synchronization time by computing maximum of domain time steps
    update_nextSyncTime()
**end for**
**for** each domain in local process **do**
▷ Finalize outputs and dump runtime measurements of each step of each domain
    finalize_domain()
**end for**
▷ Dump runtime measurements of *APESmate*
finalize_coupling()
▷ Finalize MPI
tem_finalize()

#### 5.5.3.1 Configuration

The configuration of the *APESmate* input file is pretty simple; the user just needs to provide the following for each domain:

**Filename:** is the configuration file for the domain and it should match the chosen solver configuration.

**Solver:** is the name of the solver to use for the domain and supported solvers are: 'musubi' and 'ateles'.

**Label:** is used in the *STfun* definition to identify the remote domain from which the coupling variables need to be obtained.

***nProc_dom*:** is the number of processors to distribute the domain to.

An example configuration is given in Lst. A.7 that defines four domains. Since *APESmate* steers the solvers, the termination condition like the maximum simulation time for the coupled simulation is defined in the *APESmate* configuration file. It means that the *sim_control* which is the solver configuration file is discarded and *APESmate* provides the solver with a maximum simulation time to terminate the solver time loop. In the solvers, the time loop is terminated when either one of this conditions are satisfied: reached maximum simulation time, reached steady state or non-physical state. These termination conditions are also applicable to *APESmate*.

#### 5.5.3.2 Domain distribution

*APESmate* provides three ways to distribute the domain:

1. Distribute the domains in a way that each process has only one domain. It is achieved by defining *nProc_dom* as integer such that the total number of global MPI ranks (*nProc_total*) is equal to the sum of *nProc_dom*.

2. Distribute the domains to the total number of processors (nProc_total) by defining *nProc_dom* as fraction (*nProc_dom_frac*). Then, the *nProc_dom* is obtained by *nProc_total* $\times$ *nProc_dom_frac*. However, if summation of the *nProc_dom* is not equal to the *nProc_total* then one process will run more than one domain.

3. Distribute all domains on all processes i.e. *nProc_dom*=*nProc_total*. In this distribution, in each process, the domains are executed in

sequence in an order according to their definitions in the configuration file. This approach was introduced especially for volume coupling like coupling of the multicomponent flow and the electric potential. The electric-potential equation is solved for steady state solution at each time step, which might result in load imbalance if the electric potential domain and the multi-component domain are distributed on separate processes.

To achieve good scaling of the coupled simulation in large parallel systems, the computational load between domains must be balanced. Hence, the distribution of the domains across processes has a direct impact on the performance of the coupled simulation. Therefore, *nProc_dom* is chosen carefully using prior knowledge on the performance of the solver and with some preliminary test runs of the coupled simulation. The current *APESmate* allows only for static load balancing between domains through configurable *nProc_dom*. It is possible to deploy the dynamic load balancing algorithm SParta [29] in *TreElM* but with some effort and it is planned for future work. However, the individual dynamic load balancing in the solvers can be used at the moment to reduce load imbalances within the domain.

While distributing the domains, *APESmate* creates MPI sub-communicators for each domain and passes it to the corresponding solver chosen for the domain. The solver uses this sub-communicator to communicate within the domain and *APESmate* uses the global MPI-communicator to communicate between the domains. With the distribution of domains across processes, the domain in a local process is unaware of from which process of the remote domain it should request coupling variables. Therefore, for initial communication between domains, the concept of round-robin communication is deployed. To establish this communication, *APESmate* stores the global ranks of each domain in each process. It is explained in Section 5.5.3.3.

Figure 5.23 shows the distribution of 4 domains given in the example configuration in Lst. A.7. The domains are distributed among 6 processes with 2 processes for each multi-component flow domain (the one in the back) and one process for each electric potential domain (the one in the front). The blue and green color represents the dilute channel and the concentrate channel respectively. The red line between the channels represents the membrane surface coupling interface which is treated using the membrane black-box BC. In the figure, the element size of multicomponent and electric potential domains is different to illustrate that the *APESmate* can handle non-matching meshes.

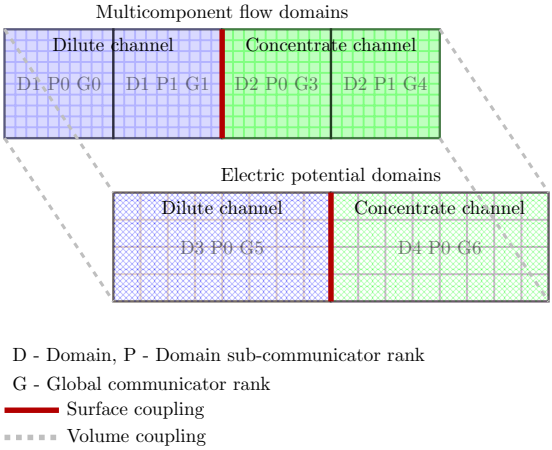



**Figure 5.23** Distribution of 4 domains (multicomponent flow and electric potential in dilute and concentrate channels) on 6 processes with four process for multicomponent flow and two process for electric potential.

#### 5.5.3.3 Initialize coupling

After the domain distribution, each domain initializes their MPI environment with its MPI sub-communicator provided by *APESmate* and loads its respective configuration and mesh files. Then, in the *initialize_domain* step, each domain builds its respective solver specific data structures, initializes *state* variables with the initial conditions, *varSys*, output infrastructure, etc. During this step, the boundary and source terms of the solver send the coupling data (points and offset_bit) to the *STfun* via *setup_indices* routines and these data are stored in the *STfun* data structure. Thus, at the end of this step, the *varSys* and the linked list of *STfun*, which are required to initialize coupling data structure are available. The linked list of *STfun* in the solver contains all *STfun* defined in the configuration file but the *APESmate* requires only the *STfun* with predefined='apesmate'. Therefore, *APESmate* gathers the coupling *STfun* of all domains in the local process to the linked list of *aps_STfun_coupling_type* data type (*cplSTfun*). As mentioned in Section 5.5.2, the points in the *STfun* are stored level-wise. Therefore, the points from each level are added as a new entry to the list of *cplSTfun*. In this type, only the pointer to the actual *STfun* data structure is stored for each level along with the local domain ID, remote domain ID and the global rank of the remote domain

(*target*) to send the coupling data to. Here, the term *target* and *source* refers to the global rank of the destination process (rank to send data to) and source process (rank to receive data from) respectively. The domain ID (domID) is the unique ID created for each domain according to the order in which they are defined in the configuration file.

A variable on a certain spatial point can be evaluated only on the process that contains the element of that point. So, in order to evaluate the coupling variables on the coupling points in the remote domain, the *target* for each point in the remote domain must be known first. Since the local domain is unaware of the distribution of serialized treeIDs on the remote domain, it does not know which *target* to request for each coupling point. Therefore, the *target* for each *cplSTfun* is created by round robin fashion through mapping the sub-communicator rank of local and remote domain.

```
!! myRank_loc is rank of local domain sub communicator
!! nProc_rem is the total of process of remote domain
!! rank_rem is the rank of remote domain sub communicator
rank_rem = mod(myRank_loc, nProc_rem)
!! globalRank_rem is the list containing global ranks of the remote domain
!! +1 is because Fortran array index starts from 1 and MPI ranks start from 0
target = globalRank_rem(rank_rem+1)
```

Now, the coupling points and the variable names are sent to the target of the remote domain. For surface coupling, to identify the coupling points on the remote domain, the *offset_bit* introduced in Section 5.5.2 is used to shift the points before sending them to the target process. Each set of coupling points and variable names received from the *source* are called coupling request (*cplRequest*). *APESmate* is designed such that each domain can send and receive arbitrary number of *cplRequest* to and from the other domain. Since *APESmate* allows sharing multiple domains in one process, it is possible that *target* is the local rank (i.e. *source*=*target*) but MPI can handle such a situation by just copying data within a process. The coupling points and variable names are communicated between domains using standard non-blocking point-to-point communication using MPI_Isend, MPI_Irecv and MPI_Waitall routines. These communications are done through the global MPI communicator. Using the *target* in *cplSTfun*, the number of coupling data to send to each process (*nCplSend_proc*) are known. However to establish point-to-point communication, the number of cplRequest to receive from each process (*nCplRecv_proc*) must be known beforehand. There are two ways to obtain *nCplRecv_proc*. Easiest way is to use MPI_Alltoall to send nCplSend_proc from all processes to all processes.

```
1 | call MPI_Alltoall( nCplSend_proc, 1, MPI_Integer, nCplRecv_proc, 1, MPI_Integer, &
2 |   &                MPI_COMM_WORLD, iError)
```

but this way is very inefficient in case of surface coupling where only a small fraction of all processors are involved.

An alternative way is to do sparse communication i.e. exchange data only between processes which are involved in coupling. This is achieved by creating two arrays: An array of *targets* and an array of *nCplSend_proc* for each *target*. Likewise, the ranks from which the cplRequest are received, are stored in an array of *sources* and another array to store *nCplRecv_proc* for each *source*. Both *nCplSend_proc* and *nCplRecv_proc* arrays have the same size and ordering as *targets* and *sources* arrays respectively. The sparse communication algorithm given in Alg. 10 is used to receive *nCplRecv_proc* from each *target*. After knowing the *nCplRecv_proc* for each *source*, *target* needs to know how many number of points (nPnts) and number of variables (nVars) to receive from each *source* so the *target* can allocate arrays to receive data accordingly. Therefore, at first the basic informations: nPnts, nVars, remote domID, local domID, couplingID and iLevel are sent to target as 6 integers per *cplSTfun*. The couplingID is an unique ID created to tag the MPI communication message between the sender and the receiver. Now, the coupling points and the variable names are sent to *target* in point-point communication between processes. In the *target*, the global rank for each point in the remote domain is identified by binary search on the tree. Additionally, the availability of the variable names in the *varSys* of the remote domain is checked. The program is aborted if a point is not found in the tree or if a variable name is not found in the *varSys*. The global ranks for the points are referred to as *pntRanks* and they are sent back to the *sources* along with the number of scalars (nScalars) of the variable names in *varSys*. The nScalars is the sum of the number of components of the variables. It is used by the *source* to allocate 'evalVal' in *cpl_value_type* (Lst: 5.4) where evaluated variables values at each time step are stored by *APESmate*. The size of 'evalVal' is $nPnts \times nScalars$.

**Algorithm 10** Sparse communication

```
1: Inputs: targets, nCplSend_pro, comm, tag
2: Outputs: sources, recv_buffer
3: targets: List of target ranks to send an integer to
4: sources: List of source ranks an integer received from
5: nCplSend_pro: Data to send to respective target ranks. Size and
   ordering same as targets
6: recv_buffer: Data received from respective source ranks. Size and
   ordering same as sources

7: subroutine SPARSE_ALLTOALL(targets, nCplSend_pro, sources,
   recv_buffer, comm, tag)
   ▷ Send data to all targets
8:     for iTarget ∈ targets do
9:         MPI_ISSEND(nCplSend_pro(iTarget), 1, MPI_Integer, tar-
   gets(iTarget), tag, comm, send_req(iTarget), iError)
10:        SERVE_REQUEST(sources, recv_buffer)
11:    end for

   ▷ Wait on sending to complete
12:    repeat
13:        MPI_TESTALL(nTargets, send_req, allSent, send_stat, iError)
14:        SERVE_REQUEST(sources, recv_buffer)
15:    until allSent

   ▷ Signal termination of the algorithm for this process
16:    MPI_IBARRIER(comm, bar_req, iError)

   ▷ Continue serving requests until all processes signaled the completion
   of their sending
17:    repeat
18:        SERVE_REQUEST(sources, recv_buffer)
19:        MPI_TEST(bar_req, allProcsDone, bar_stat, iError)
20:    until allProcsDone

   ▷ Ensure all processes had the opportunity to shut down their pending
   receives to avoid any confusion with subsequent sparse alltoall exchange
21:    MPI_BARRIER(comm, iError)
22: end subroutine
```

```
23: procedure SERVE_REQUEST(sources, recv_buffer)
24:     MPI_IPROBE(MPI_ANY_SOURCE, tag, comm, got_request,
    recv_stat, iError)
25:     if got_request then
26:         source ← recv_stat(MPI_SOURCE)
27:         MPI_RECV(recvdat, 1, MPI_Integer, source, tag, comm,
    recv_stat, iError))
28:         APPEND(sources, source)
29:         APPEND(recv_buffer, recvdat)
30:     end if
31: end procedure
```

The pntRanks in the *source* contains the correct *target* to communicate each point. The *source* uses pntRanks to create a communication buffer [1] to receive evaluated variables from the *target*. Furthermore, this buffer is used to gather points per process to send points to the correct *target*. In addition to points, variable names are also be sent to the correct *target*.

Now, the *source* knows the correct *target* to send coupling data but to establish a point-to-point communication, the *target* needs to know from which *source* to receive data. This is the same situation as before i.e. *nCplSend_proc* is known but not the *nCplRecv_proc*. Therefore, the sparse communication Alg. 10 is deployed again to obtain *nCplRecv_proc* in each target. Now, the basic informations (6 integers) are sent to the correct *target* to allocate arrays to receive the actual data (points and variable names). In the *target*, the variable names of one coupling ID are combined into a single variable using the 'combine' operation and they are added to the remote domain *varSys* with couplingID as a variable name. The position of every coupling variable added to the *varSys* are stored in *aps_coupling_variable_type*. Since the process can have multiple domains, for every variable the local domID is stored to access the correct the *varSys*. In the *target*, the local and remote domID received from the *source* are stored as remote and local domID respectively. For each variable, the points and the evaluated variable values (evalVal) are stored level-wise in the *target*. Similar to the communication buffer 'recvBuffer' in the *cpl_value_type* in the source, the send communication buffer is created in the *target* using the *source* of the received points for each level and stored along with points and evalVal. This communication buffer is used

[1] data type in *TreElM*, contains process-wise buffers to exchange data with a specific process

to send the evaluates variable values back to the *source* at every time step. Since *APESmate* supports arbitrary number of *cplRequest*, the *target* can have an arbitrary number of coupling variables. Each variable is evaluated level-wise and sent back to the *source*. Note that after the initialization, in the synchronization step, the evaluated variables are sent from the *target* to the *source* i.e. the *source* becomes the *target* and vice vera due to the definition of the *source* and the *target*. It's worth mentioning that all information in the *target* are in the remote domain so they are stored in *APESmate* while the information in the *source* are in the local domain so they are stored in the corresponding *STfun* variable.

In short, these are steps performed in the *initialize_coupling*:

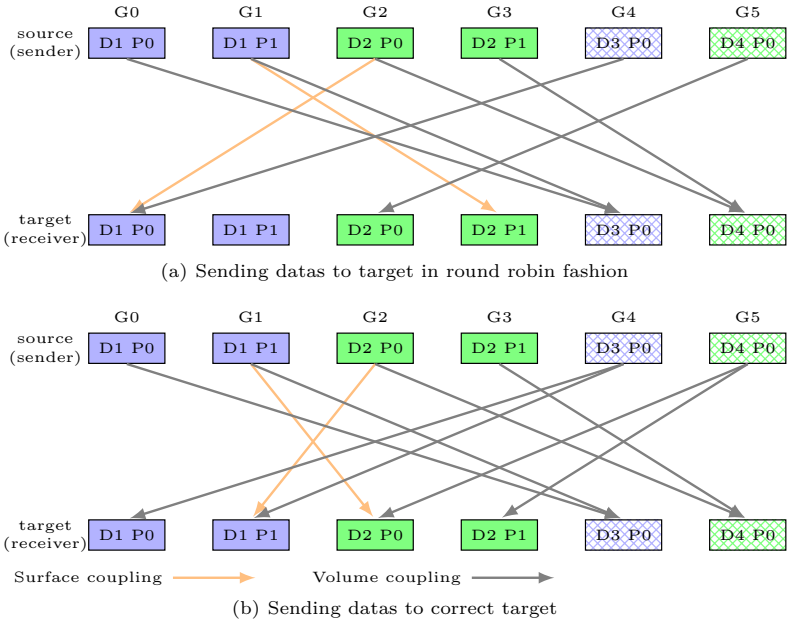
- Gather *STfun* with predefined 'apesmate' into a list of *cplSTfun*. Each level of the same *STfun* is appended as separate entry into this list.
- Send points and variable names to the *target* that contains the remote domain using round-robin fashion.
- Identify rank for each point in the remote domain and check for availability of variable names in the remote domain *varSys*.
- Send pntRanks and nScalars of variable names back to the *source*.
- Create receive communication buffer in the *source* using pntRanks and allocate evalVal. Use this communicate buffer to send points and variable names to the correct *target*.
- In the *target*, combine received variable names of one couplingID into a single variable and append it to the remote domain *varSys* with couplingID as variable name.
- Store points from every couplingID level-wise and create send communication buffer using points received from source for each coupling variable level-wise.

Figure 5.24a and Figure 5.24b illustrates the initial exchange of coupling data to the *target* in the round robin fashion and then final exchange of coupling data to the correct *target* respectively. For this illustration, the four domain coupling example presented in Figure 5.23 is used. The first domain (D1) and the second domain (D2) are coupled with each other on the surface and each domain is distributed on 2 processes. In the figure, the domain sub-communicator ranks are denoted by P and the global

communicator ranks are denoted by G. The processes which contain the surface coupling interface are P1 of D1 and P0 of D2. But due to initial round robin exchange, the coupling data in P1 of D1 are sent to P1 of D2 and the data in P0 of D2 are sent to P0 of D1. Regarding the volume coupling, the domain D1 is coupled with D3 and D2 is coupled with D4. Since D3 and D4 are distributed on a single process, the volume data from all processes of D1 and D2 are sent to P0 of D3 and D4 respectively. On the other hand, the data from D3 and D4 are sent to P0 of D1 and D2 respectively. After the data are successfully exchanged between the domains, each domain identifies the global rank that contains the requested point. As in this example, P0 of D1 identifies that the points from P0 of D2 must be evaluated in P1 of D1 and sends this information back to P0 of D2. Each process that received data from the remote domain identifies the actual process, which contains the points and sends that back to the requested process. After this, each source knows the *target* for each point. In the example, all the points from P1 of D1 are now sent to P0 of D2 and vice versa. However, P0 of D3 and D4 gather points according to their distribution in P0 and P1 of remote domains D1 and D2.

#### 5.5.3.4 Synchronize domains

In the *synchronize_domains* step, the coupling variables are evaluated level-wise in the remote domain and the values are communicated level-wise to the requested domain using the communication buffers. The values in the 'recvBuffer' are per process so they are copied to 'evalVal' in the *cpl_value_type* in the same order as the array of points in the point data type. This step is performed at every synchronization time in the time loop and also once before the time loop so the solver can start with correct BC and source terms. As mentioned earlier, the time loops in the solvers are terminated when the maximum simulation time is reached or the simulation reaches steady state or the simulation becomes non-physical. With *APESmate* steering the solvers, the maximum simulation time (tMax) for the solver is provided by *APESmate* by setting tMax of the solver to the next synchronization time i.e. $tMax = tSync = tNow + dtMax$ where $tNow$ is the current simulation time and $dtMax$ is the maximum time step of all domains. The time step $dt$ of the domain depends on the domain element size $dx$, the scheme order and the physical properties of the flow like viscosity and velocity or the flow characteristics defined by the Reynolds number or the Mach number. Thus, each domain can have different $dt$ depending on their equation system and the solver. So, *APESmate* computes the maximum time step $dtmax$ and lets each domain




(a) Sending datas to target in round robin fashion

(b) Sending datas to correct target

**Figure 5.24** Exchange of coupling datas (points and variable names) between domains. Arrows represents flow from data from source to target.

run until *tSync* is reached or till the steady state is reached. For domains with different *dt*, this results in sub-cycling in a domain with *dt* smaller than $dtMax$ and at each sub-cycling step this domain requires coupling variable values at intermediate time steps. A proper way to provide those values at intermediate time step is through the interpolation of coupling variables in time. At the moment, *APESmate* does not support the interpolation in time so the domains with different time steps cause inconsistencies in the simulations. Therefore, the domains are restricted to have either the same time step or to be solved for steady state at each time step.

To reduce the computational cost of variable evaluation in the remote domain, the two-state evaluation presented in Section 5.5.1 using *setup_indices* and *get_valOfIndex* is used. Since *APESmate* uses *varSys* of the solver for variable evaluation, the numerical accuracy of the evaluated variable depends on the order of the solver.

Regarding the performance of the coupled simulation, the variable evaluation using the *get_valOnIndex* routine is the routine which mostly

contributes to the computational cost and the communication cost depends on the number of processes which are involved in the coupling. In case of surface coupling, only the processes with the coupling interface are involved in coupling and in case of volume coupling all processes are involved in coupling. Therefore, the communication cost for the volume coupling will be higher than for the surface coupling. As long as the solvers are scalable, the coupling is also scalable in large parallel systems. However, if the domains are not properly distributed according to their computational cost of each element then load imbalances between the domains will reduce the overall performance and the parallel efficiency. E.g. for surface coupling, the higher computation cost of the few processes which are involved in coupling will lead to imbalances within the domain, in case no proper load balancing is used. So, the user must keep this in mind in defining the number of processes per domain. The performance of *APESmate* is compared to the coupling library preCICE on the SuperMUC system [53]. The test case used for this performance comparison was surface coupling of two cubes: inner and outer cube arranged in a way that inner cube is enclosed by outer cube and 3D Gaussian pulse travel from inner to outer cube. In comparison to preCICE, *APESmate* showed an advantage of 20% lower overall computational time. Furthermore, the numerical accuracy of the *APESmate* is found to be far superior than the preCICE.

## 5.6 Conclusion

In this chapter, the highly scalable simulation framework *APES* was introduced. The implementation details and algorithms of *APES* subparts: *Seeder* - octree mesh generator, *Musubi* - LB solver and *APESmate* - integrated coupling tool were discussed in detail. The development and implementation of *Seeder*, the incompressible flow, the multicomponent flow and the electric potential in *Musubi*, and *APESmate* are major contribution of this thesis.

The presented mesh generator *Seeder* is very efficient in generating octree meshes automatically. In the presented algorithm, the protoTree is created at first by refining the bounding cube towards the boundary geometries to the level defined for each boundary. The main advantage of this approach is that only nodes near the boundary are refined to their finest level so the flooding algorithm runs over fewer nodes to flood the computational domain. After flooding, the flooded nodes are refined to their finest level and those flooded nodes are dumped to disk in the *TreElM* format i.e. the nodes are serialized by depth-first ordering following the SFC. The

introduction of the protoTree reduces the overall mesh generation time and runtime memory consumption. The very high performance of *Seeder* for creating a mesh around the spacer geometry was presented and it was shown that *Seeder* can generate the mesh with several hundred millions of fluid elements in a few minutes. The presented algorithm is implemented in serial and it can be parallelized with some effort in distributing geometries and merging the trees from different partitions, etc.

The numerical scheme LBM for the incompressible fluid flow, the multicomponent flow and the electric potential are implement in the highly scalable LBM solver *Musubi*. In the solver, the serialized elements are equally distributed among partitions resulting in almost perfect load balancing. The ordering by SFC helps to maintain locality by identifying the rank of the element on a remote process locally. The serialized elements from *Seeder* are converted into level-wise list of elements in the solver to apply uniform operations to all elements in a level. The stream-collide algorithm used in the compute kernels for each of those physical systems was applied level-wise. The compute kernels were optimized for memory and the number of floating point operations. The performance of the single-component LBM and multicomponent LBM with three components in *Musubi* was analyzed on the Hermit X86 system for two test cases: the fully periodic cubic domain and the woven spacer structure. This analysis showed that *Musubi* has almost perfect weak scaling, and the strong scaling is very good down to a minimum problem size. Overall, the parallel efficiency of strong scaling for the single component and multicomponent LBM model is above 70%. In addition, the algorithm of the main program of the solver was presented to show the similarity between the solver and the coupling algorithm.

The scalable integrated coupling tool *APESmate* to couple different *APES* solvers was introduced. Two key features used by *APESmate* are the *STfun* and the *varSys* which were elaborated in detail. The *varSys*allows for several definition of variables in the configuration file and the *STfun*is one of the variable type in the *varSys*. The surface and volume coupling were realized through variables defined as BC and source term respectively. These variables are defined as the *STfun* in the configuration file. The coupling algorithm was described in detail with explaining its three important steps: domain distribution, initialize coupling and synchronize domains. *APESmate* uses the solver data structure to evaluate the coupling variables on the remote domain so the numerical accuracy of coupling is in the same order as the order of the solver. The scalability of this tool depends on the computational load balance between the domains. To handle the imbalances in the domain introduced by the coupling,

*APESmate* allows users to specify the number of processes per domain so the coupled simulation can be scalable as long as the solver is scalable. The *APESmate* is developed to handle arbitrary numbers of domains and arbitrary numbers of coupling partners per domain. Therefore, it can be used to simulate any coupled multiphysics problems. However, it is limited to physical systems implemented in *APES* solvers using the *TreElM* data structure.

# 6 Numerical validation and verification

This chapter presents various numerical experiments to validate and verify the presented numerical methods and the framework. Validation and verification are an essential part of numerical simulations to ensure the accuracy and correct implementation of the numerical schemes. The implementation of the presented numerical schemes in *Musubi* is validated by comparing the numerical results with the analytical solution but the analytical solutions exists only under certain assumptions on the physical system. Therefore, the simple two-dimensional (2D) test cases with known analytical solution are used to validate the individual numerical schemes presented in Chapter 3 and also the coupled setup presented in Chapter 4. The coupling of the multicomponent flow and the electric potential is validated using the Boltzmann approximation of ion distribution in dilute aqueous $NaCl$ solutions. However, there is no analytical solution to validate the coupling of the membrane black-box model and the multi-component flow. So in this case, the numerical results are just verified qualitatively. For all these numerical validations, the $D2Q9$ stencil with MRT collision operator is used except for the LBM for thw electric potential, where the BGK collision operator is used. The more complex three dimensional flow simulation of *Musubi* with spacer geometry using the $D3Q19$ stencil is validated against experiment and it is presented in a later chapter.

## 6.1 Poiseuille flow

The LBM for incompressible Navier-Stokes equations is validated using the well-known Poiseuille flow [54] and it is created by the steady flow between two parallel plates in the presence of a constant pressure gradient. Consider a 2D channel of length $L$ and height $H$ as shown in Figure 6.1. In this case, a constant pressure gradient in x-direction i.e. $-\partial P/\partial x = constant$ drives the flow along the length of the channel. Neglecting other external forces, the incompressible Navier-Stokes Eq. 2.56 can be reduced to the one dimensional equation for velocity in x-direction

$$\frac{\partial v_x}{\partial t} - v_x \frac{\partial v_x}{\partial x} = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \nu \frac{\partial^2 v_x}{\partial y^2}. \tag{6.1}$$

Assuming that the flow reaches steady state i.e. $\partial v_x/\partial t = 0$ and is fully developed, then $\partial v_x/\partial x = 0$ from continuity Eq. 2.55. Furthermore, setting $\partial P/\partial x = \Delta P/L$ in the above equation results in

$$\frac{\partial^2 v_x}{\partial y^2} = \frac{\Delta P}{\rho \nu L}. \tag{6.2}$$

Solving this equation with no-slip BCs on top and bottom walls i.e. $v_x(y = 0) = v_x(y = H) = 0$ results in the analytical solution of the velocity profile across the height

$$v_x(y) = \frac{\Delta P}{2\rho \nu L} y(y - H). \tag{6.3}$$

Thus, the velocity profile varies parabolically across the height, reaching its maximum $v_m$ in the center axis at $y = H/2$

$$v_m = \frac{\Delta P H^2}{8\rho \nu L}. \tag{6.4}$$

Since the pressure gradient is constant, the analytical solution of the pressure profile along the channel length can be written as

$$P(x) = P_o + \Delta P(x - L) \tag{6.5}$$

where $P_o$ is the reference pressure defined at the outlet (x=L). The pressure drop across the channel $\Delta P$ is $P_o - P_{in}$ and $P_{in}$ is the pressure at the inlet at (x=0).

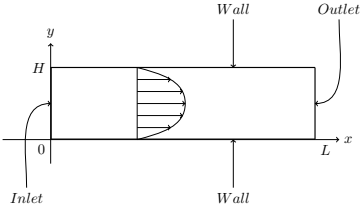
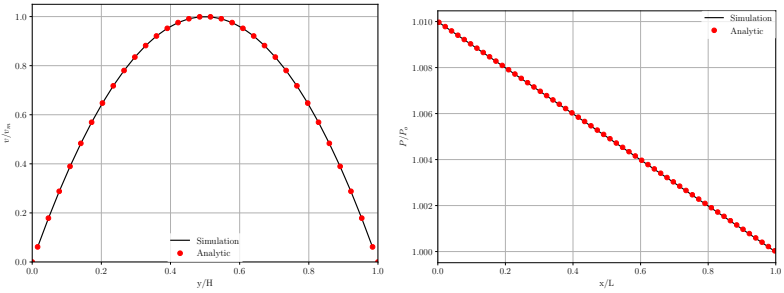



**Figure 6.1** Setup for Poiseuille flow.

In the numerical setup, there are three ways to create the flow in the channel: defining a pressure gradient as an external force; defining the pressures $P_in$ and $P_o$ as pressure BC at the inlet and the outlet respectively; defining velocity profile Eq. 6.3 as velocity BC at the inlet and $P_o$ as pressure BC at the outlet respectively. Here, the last approach is used so the results can be used to analysis the accuracy of velocity

and pressure BCs presented in Section 3.1.2. The no-slip walls at y=0 and y=H are treated with the simple bounce back BC. The channel height and the length are $H = 0.41\,\mathrm{m}$ and $L = 5H$ respectively. The flow parameters used for this validation are the pressure drop across the channel $\Delta P = 1 \times 10^{-2}\,\mathrm{N\,m^{-2}}$, the fluid density $\rho = 1.0\,\mathrm{kg\,m^{-3}}$ and the kinematic viscosity $\nu = 1 \times 10^{-3}\,\mathrm{m^2\,s^{-1}}$. The comparison of the simulated velocity profile and pressure profile with the analytical solution are plotted in Figure 6.2. The domain is resolved with 32 elements in the height and 160 elements in the length. With the relaxation parameter $\lambda_\nu = 1.8$, the maximum lattice velocity at the middle is $v^* = 0.0243$. The flow is initialized with velocity $\boldsymbol{v} = 0$ and pressure $P = P_o = 1.0\ \mathrm{N\,m^{-2}}$. The simulation reached steady state after 61000 time steps. As can be seen in Figure 6.2, the simulation results are in good agreement with the analytical solution and the relative $L^2$ error norm for the velocity and the pressure are of $5.9101 \times 10^{-4}$ and $1.360\,349 \times 10^{-5}$ respectively.




(a) Velocity profile

(b) Pressure profile

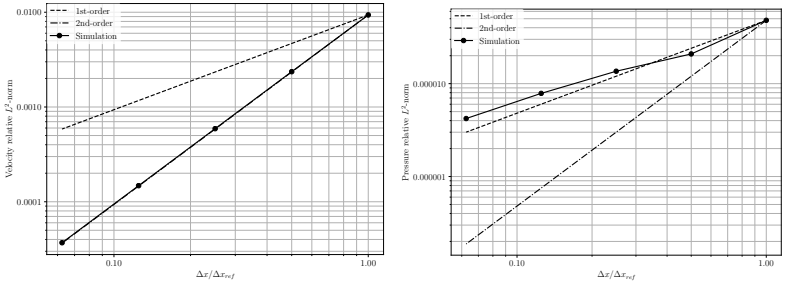
**Figure 6.2** Comparison of the simulated velocity profile across the height (left) and pressure profile across the length (right) with the analytical solution for the Poiseuille test case

In addition to the validation, the mesh convergence analysis was performed to show the accuracy of the individual numerical schemes and their BCs. The error between analytical solution ($u_a$) and numerically simulation ($u_s$) is measured in relative $L^2$ error norm $\|e\|_{L^2}$ given by

$$\|e\|_{L^2} = \sqrt{\frac{\int \|u_a - u_s\|_{L^2}^2 dV}{\int \|u_a\|_{L^2}^2 dV}}. \tag{6.6}$$

For this analysis, the diffusive scaling $\delta t \propto \delta x^2$ is used i.e. the time step size decreases with square of the element size. The number of elements

in the height (nH) is varied from 8 to 128. At every refinement, nH was increased by a factor of 2. Due to the diffusive scaling the relaxation parameter $\lambda_\nu$ is fixed to 1.8 and with every refinement the number of time steps to reach the steady state increases by a factor of 4. The relative $L^2$ error norm of velocity and pressure is plotted in Figure 6.3. It shows that the velocity error is of $2^{nd}$ order and the pressure error is of $1^{st}$ order.




(a) Velocity profile (b) Pressure profile

**Figure 6.3** Relative $L^2$ error norm of velocity and pressure on various resolution for the Poiseullie flow test case

The Poiseullie flow test case is also used to validate the multicomponent LBM for ideal mixture. For this experiment, the channel is considered to be homogeneously filled with two distinct species. At the inlet, the same mole flux is imposed for both species as a parabolic profile given in Eq. 6.3 and at the outlet, the concentration and the velocity of the species are extrapolated. Similar to single component LBM, no-flux is imposed at the walls. The mesh convergence analysis shown in Figure 6.4 depicts that the velocity convergence in second-order for both moments based and bounce back BCs, which is in prefect agreement with the theoretical predictions.

## 6.2 Stefan tube

The diffusion dominated Stefan tube experiment is used to validate the multi-component LBM for ideal mixture presented in Section 3.2.1. The Stefan tube is a device used for measuring diffusion coefficients in binary vapor mixtures. The setup of the Stefan tube is depicted in Figure 6.5. The tube is aligned in $y$-direction (vertical), open at the top to the carrier gas and at the bottom of the tube is filled with a quiescent liquid mixture. The liquid evaporates and diffuses to the top of the tube. The carrier gas

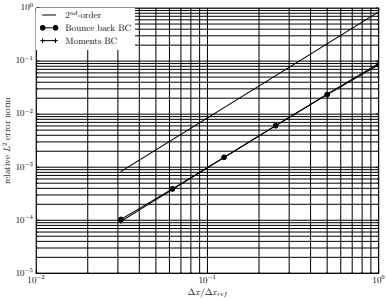



**Figure 6.4** Relative $L^2$ error norm of velocity for moments and bounce back based BC on various resolution.

carries away the liquid vapor mixture and keeps the mole fraction of the evaporative vapor there essentially to nothing. The mole fraction of the vapor at the liquid-vapor interface is its equilibrium value.

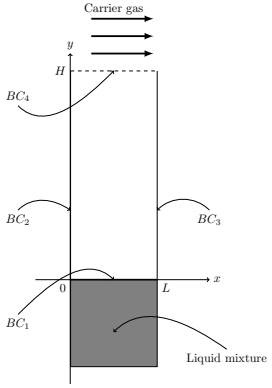



**Figure 6.5** Setup of the Stefan tube experiment

In this validation, the simulation results are validated with Stefan tube experiments from Carty and Schrodt (1975) [16]. In their experiments, the binary liquid mixture of acetone (component 1) and methanol (component 2) was evaporated from the bottom and air (component 3) was used as the carrier gas at the top. The height of the tube used in the experiment is $H = 0.238\,\mathrm{m}$. The molecular weight of the components 1, 2 and 3 in $\mathrm{kg\,mol^{-1}}$are $M_1 = 58.08 \times 10^{-3}$, $M_2 = 32.04 \times 10^{-3}$ and $M_3 = 28.86 \times 10^{-3}$ respectively. The Maxwell-Stefan binary diffusivities coefficients of the mixture are $\mathcal{D}_{1,2} = 8.48 \times 10^{-6}\,\mathrm{m^2\,s^{-1}}$, $\mathcal{D}_{1,3} =$

$13.72 \times 10^{-6}\,\mathrm{m^2\,s^{-1}}$ and $\mathcal{D}_{2,3} = 19.91 \times 10^{-6}\,\mathrm{m^2\,s^{-1}}$ [96]. The total molar density is $c_t = 1000\,\mathrm{mol\,m^{-3}}$. The boundary conditions are prescribed as follows:

$BC_1$ - **Bottom:** Filled with binary liquid mixture of acetone and methanol. The mole fractions of those two components are fixed to $\chi_1^{BC_1} = 0.319$ and $\chi_2^{BC_1} = 0.528$. Assuming, that the component 3, air, does not dissolve with the binary mixture at the bottom, the mole flux of this component is set to zero i.e. $\boldsymbol{N}_3 = 0$, resulting in $\boldsymbol{v}_3 = 0$. Thus, the mole fraction boundary condition is used for liquid mixture component and mole flux boundary condition for component 3.

$BC_2$ **and** $BC_3$ - **periodic:** Assuming the diffusion takes place only in one direction from bottom to top of the tube i.e. unidirectional, the $x$-direction is considered to be periodic.

$BC_4$ - **Top:** The carrier gas (air) at the top of the tube carries away the component 1 and 2 resulting in zero mole fraction of vapor mixture. But setting the mole fraction to zero causes numerical instabilities. Therefore, a smallness parameter $\epsilon_s = 1 \times 10^{-3}$ is used to impose $\chi_1^{BC_4} = \chi_2^{BC_4} = \epsilon_s$ and the mole fraction of air is $\chi_3^{BC_4} = 1 - 2\epsilon_s$.

Note that for Dirichlet BC for the species mole fraction, the species momentum is extrapolated (Neumann boundary condition) and for Dirichlet BC for the species mole flux, the species mole fraction is extrapolated as explained in Section 3.2.2. In addition, zero kinematic pressure difference is assumed between $BC_1$ and $BC_4$ such that no convective species transport should occur (assuming that the initial condition has zero velocity). The initial conditions are

$$\begin{aligned}
\chi_1(\boldsymbol{x}) &= \chi_1^{BC_1}\left(1 - \frac{y}{H}\right) + \epsilon_s \\
\chi_2(\boldsymbol{x}) &= \chi_2^{BC_1}\left(1 - \frac{y}{H}\right) + \epsilon_s \\
\chi_3(\boldsymbol{x}) &= 1.0 - \chi_1(\boldsymbol{x}) - \chi_2(\boldsymbol{x}) \\
\boldsymbol{v}_1(\boldsymbol{x}) &= \boldsymbol{v}_2(\boldsymbol{x}) = \boldsymbol{v}_3(\boldsymbol{x}) = 0
\end{aligned}$$

where $\boldsymbol{x} = (x, y, z)$ is the spatial coordinate.

In addition to comparing the multi-component LBM results with experiments, they are also compared to the numerical solution obtained by a shooting method [76] for the steady state mass transport equations. The numerical solution of a shooting method is used to study the mesh

convergence analysis of multi-component LBM boundary conditions. Concerning this numerical solution, at constant temperature and pressure, the total molar density and the binary diffusion coefficients are constant and neglecting any external forces, the only driving force is the mole fraction gradient, i.e. $\boldsymbol{d}_k = \nabla \chi_k$. Furthermore, since the diffusion is assumed to be unidirectional, the continuity Eq. 2.14 at steady state implies constant flux i.e. $N_k = constant$. Thus, with these assumptions the Maxwell-Stefan Eq. 2.27 can be reduced to a system of one dimensional first-order linear differential equations

$$\frac{\partial \chi_k}{\partial y} = \sum_{l=1}^{n} \frac{(\chi_k \boldsymbol{N}_l - \chi_l \boldsymbol{N}_k)}{c_t \mathcal{D}_{k,l}}. \tag{6.7}$$

These equations along with the boundary conditions discussed earlier results in a boundary value problem, which is solved by the shooting method [76].

For the multicomponent LBM simulation, the height of the tube is resolved with 120 elements and due to the periodic BC in $x$-direction, the length is resolved with just one element. With BGK collision, the same relaxation parameter $\lambda_k^{\nu} = 2$ is used for all components. The simulations are ran until the steady state is reached. Figure 6.6 shows the mole

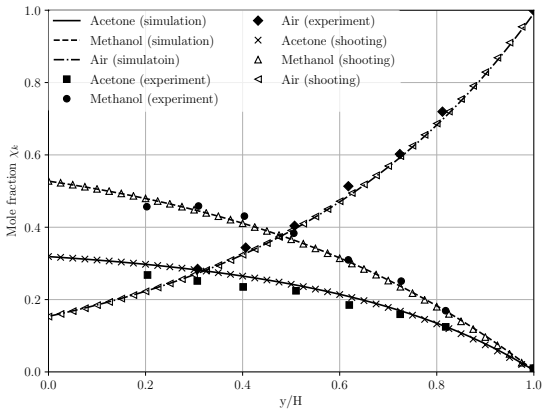



**Figure 6.6** Comparison of concentration profiles from multi-component LBM with experiment and shooting method at steady state.

fraction profiles of the three components from the multi-component LBM simulations, experiment and the shooting method at steady state. The

numerical results of LBM and shooting method are in good agreement with experiment [96]. Thus. these results point out that the presented multicomponent LBM model is able to recover the full Maxwell-Stefan equations.

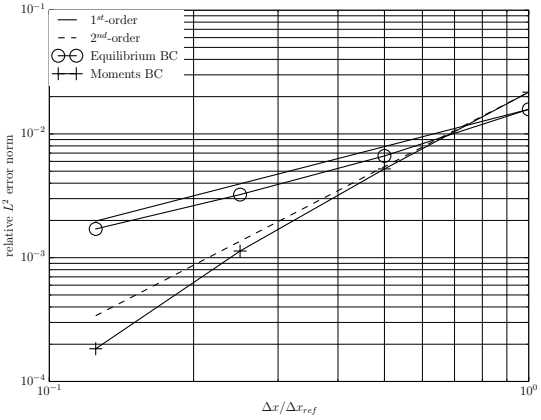



**Figure 6.7** Relative $L^2$ error norm of mole fraction of component 3 for moments and equilibrium based BC on various resolution

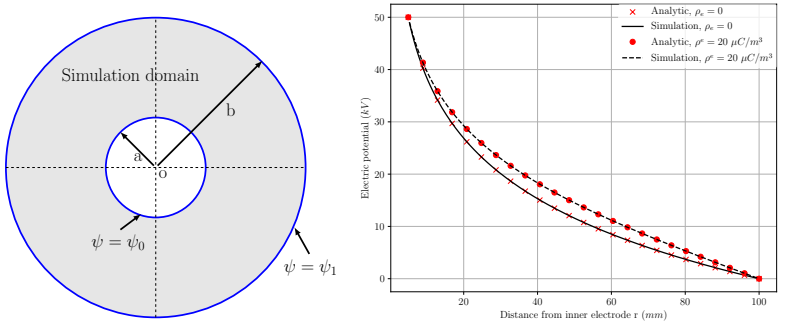
In addition, the relative $L^2$ error norm of the mole fraction profile of component 3 is shown in Figure 6.7 for moments and equilibrium Section 3.2.2 based BCs. Here, the error is computed between the multicomponent LBM results and the shooting method. The equilibrium based BC for the mole fraction are expected to be rough and of first-order accuracy (for concentration and velocity) in the diffusive asymptotic limit. However, the moment based boundary conditions [10], with imposed mole fraction for component 1 and 2, no flux for component 3 at $BC_1$, and imposed mole fraction for all components at $BC_4$, provide sufficient accuracy to recover the correct mole fraction profiles. Figure 6.7 shows second-order convergence for the moments based BC and first-order convergence for the equilibrium based BC in the Stefan tube setup, which is in good agreement with the theoretical analysis. Even though, the moment based BCs deliver more accurate simulation results, it is difficult to generalize for complex geometries, therefore the equilibrium based BC is used for mole fraction boundaries.

## 6.3 Concentric cylinders

The LBM for the electric potential is validated using the concentric cylinder test case [61]. The simulation setup is shown in Figure 6.8a. The diameter of the inner and the outer cylinders are $2a = 10\,\text{mm}$ and $2b = 200\,\text{mm}$ respectively. A very high voltage of $\psi_0 = 50\,\text{kV}$ is applied at the inner electrode and the outer electrode is grounded i.e $\psi_1 = 0\,\text{V}$. The analytical solution of the Poisson's Eq. 2.64 for this test case is available for both zero and constant charge density cases:

$$\psi(r) = \frac{\rho^e}{4\varepsilon}\left(b^2 - r^2\right) + \left[\psi_0 - \frac{\rho^e}{4\varepsilon}\left(b^2 - a^2\right)\right]\frac{ln\ r - ln\ b}{ln\ a - ln\ b}, \tag{6.8}$$

where $r$ is the radial distance from the inner cylinder center and the permittivity of the medium $\varepsilon = 8.854 \times 10^{-12}\,\text{C}^2\,\text{J}^{-1}\,\text{m}^{-1}$. Figure 6.8b shows the numerical simulation results at steady state and analytical results with charge density $\rho^e = 0$ and $\rho^e = 20\ \mu\text{C}\,\text{m}^{-3}$. The numerical results of both cases are in good agreement with the analytical results. For the numerical simulation, the diameter of the inner cylinder is resolved with 16 elements and the potential diffusivity $\gamma = 0.167$. To improve the accuracy of BC, the cylinders are approximated by q-values and the non-equilibrium extrapolation BC (Section 3.4.1) is used to impose Dirichlet potential values at the cylinders. This proves that the LBM for the electric potential can be used to solve the potential equation in the flow channel with spacer geometry.




(a) Simulation setup

(b) Comparison of simulation and analytical results

**Figure 6.8** Numerical experiment of concentric cylinder test case

## 6.4 Taylor dispersion

In this section, the convection dominated Taylor dispersion experiment is used to investigate the multi-component LBM for ideal mixture with and without an external electrical force. The Taylor dispersion is an effect in fluid mechanics in which the shear flow smears out the concentration distribution of the species in the direction of flow and enhances the effective diffusivity of the species. It is a test case that is well known in the field of multi-component flows [4, 95] and is widely used to test mass transport models since it includes laminar hydrodynamics as well as pure diffusion phenomena.

Here, a 2D fluid channel of dimensions $[0; L] \times [0; H]$ with a laminar Poiseuille flow profile for the mole flux of the species is considered. The direction of flow is the positive $x$- direction. The height $H$ and the length $L$ of the channel is 0.04 cm and 0.24 cm respectively. The diameter of the spacer filament is $d_f = 0.02$ cm which is same as in the spacer flow investigations discussed in Section 7.1.1. The simulation setups used for this Taylor dispersion experiment is depicted in Figure 6.9. Three different configurations are investigated: 1) no filaments (plain channel), 2) centered filaments and 3) zigzag filaments. The centered filaments and the zigzag filaments represent the cross-section of typical spacer flow channels. In centered filaments, spacer filaments are submerged at the center of the flow channel and in zigzag filaments, alternative spacer filaments touch the top and the bottom of the flow channel. In both cases, the filaments are 0.1 cm apart from each other. The length of the channel is slightly larger then a single spacer element length to reduce the influence of the inlet and outlet boundaries. The boundary effects on the inner domain are discussed later in this section.

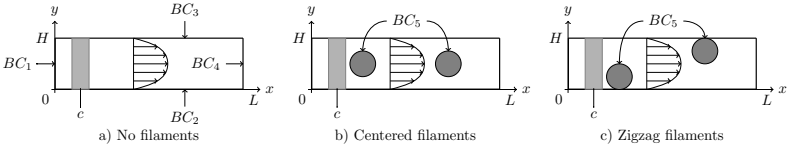



**Figure 6.9** Three configurations used for the numerical simulation of the Taylor dispersion experiment. The gray region depicts the region with high concentration of species 2 and 3. The center of the concentration stripe is $x = c$ and its width is $\Delta c$. The mole flux distribution is a laminar Poiseuille pipe flow profile.

A mixture of aqueous NaCl solution is considered for this investigation and the species 1, 2 and 3 corresponds to $H_2O$, $Na^+$ and $Cl^-$ respectively.

The parameters of aqueous NaCl solution are listed in Table 6.1. The values chosen for this experiment are similar to the salinity of seawater 35 $g/kg$ at temperature 298.15 K. The binary Maxwell-Stefan diffusivity coefficients are evaluated for the concentration of $c_{NaCl} = 512.5\,\mathrm{mol\,m^{-3}}$ in Eq. 2.30. Initially, the ionic species 2 and 3 are distributed with non-zero concentration only in a small stripe perpendicular to the flow direction (i.e. aligned in the $y$ direction) and it is depicted as grey region in the Figure 6.9. The initial conditions are chosen as ($\varkappa = -\ln(10^{-12})$)

$$\begin{aligned}
\chi_2(\boldsymbol{x}) &= \frac{c_2}{c_t} \cdot \exp\left(-\varkappa\left(\frac{x-c}{\Delta c/2}\right)^{2\kappa}\right) + \epsilon_s \\
\chi_3(\boldsymbol{x}) &= \frac{c_3}{c_t} \cdot \exp\left(-\varkappa\left(\frac{x-c}{\Delta c/2}\right)^{2\kappa}\right) + \epsilon_s \\
\chi_1(\boldsymbol{x}) &= 1.0 - \chi_1(\boldsymbol{x}) - \chi_2(\boldsymbol{x}). \\
\boldsymbol{v}_1(\boldsymbol{x}) &= \boldsymbol{v}_2(\boldsymbol{x}) = \boldsymbol{v}_3(\boldsymbol{x}) = 0
\end{aligned}$$

where $\epsilon_s = 1 \times 10^{-3}$ represents again a smallness parameter to avoid division by zero, $c = L/8$ the center of the concentration stripe, $\Delta c = 3c$ the total width of the stripe and $\kappa = 2$ determines the sharpness of the transition zone of the concentration stripe for ionic species (larger $\kappa$ leads to sharper profile). The fraction $c_2/c_t$ and $c_3/c_t$ defines the highest mole fraction of the species 2 and 3 respectively at the center of the concentration stripe.

Furthermore, the boundary conditions are imposed as follows (cf. Figure 6.9):

$BC_1$ - **inlet:** For the inlet boundary the x-component of the species mole flux ($N_{k,x}$) is defined as a parabolic Poiseuille profile

$$N_{k,x}(y) = \chi_k(y) c_t \frac{4 v_m y (H-y)}{H^2} \tag{6.9}$$

where the mole fraction $\chi_k(y)$ of species 1, 2 and 3 are $\chi_1 = 1 - 2\epsilon_s$, $\chi_2 = \chi_3 = \epsilon_s$ satisfying the initial conditions, and the maximum velocity at the middle of the channel is $v_m = 1\,\mathrm{cm\,s^{-1}}$. Thus, the Reynolds number for this setup is $Re = v_m H/\nu = 3.8$.

$BC_2$ **and** $BC_3$**:** Without an external electrical force: The bottom and top boundaries are treated as solid no-slip walls for all species i.e. $\boldsymbol{v}_1 = \boldsymbol{v}_2 = \boldsymbol{v}_3 = 0$. This type of BC is covered by the simple bounce-back rule.

| Parameter | Value |
|---|---|
| $\rho_t$ [kg m$^{-3}$] | 1025 |
| $c_1, c_2, c_3$ [mol m$^{-3}$] | 54940, 512.5, 512.5 |
| $c_t$ [mol m$^{-3}$] | 54965 |
| $\chi_1, \chi_2, \chi_3$ | 0.981 685, $9.1575 \times 10^{-4}$, $9.1575 \times 10^{-4}$ |
| $\mathcal{D}_{1,2}$ [m$^2$ s$^{-1}$] | $1.249 \times 10^{-9}$ |
| $\mathcal{D}_{1,3}$ [m$^2$ s$^{-1}$] | $2.079 \times 10^{-9}$ |
| $\mathcal{D}_{2,3}$ [m$^2$ s$^{-1}$] | $8.618 \times 10^{-11}$ |
| $\nu$ [m$^2$ s$^{-1}$] | $1.054 \times 10^{-6}$ |
| $\zeta$ [m$^2$ s$^{-1}$] | $2\nu/3$ |
| $M_1$ [kg mol$^{-1}$] | $18.015\,28 \times 10^{-3}$ |
| $M_2$ [kg mol$^{-1}$] | $22.989\,77 \times 10^{-3}$ |
| $M_3$ [kg mol$^{-1}$] | $35.4527 \times 10^{-3}$ |
| $z_k(1,2,3)$ | (0,1,-1) |
| T [K] | 298.15 |

**Table 6.1** Parameters of aqueous $NaCl$ solution used for the Taylor Dispersion test case

With an external electrical force: The bottom and top boundaries are treated as AEM and CEM for ionic species and no-slip wall for solvent species 1. The membrane black-box model is used to define the AEM and CEM. The transport number of $Na^+$ and $Cl^-$ on CEM and AEM are $T^{CEM}_{Na^+} = 0.971$, and $T^{AEM}_{Cl^-} = 0.998$.

$BC_4$ - **outlet:** At the outlet, the Neumann BC is imposed for the concentration and velocities of the species.

$BC_5$ - **spacer:** Spacer filaments are approximated by q-values and treated by higher order wall BC.

Each of these three configurations is investigated with and without an external electrical force. At first, no electrical force is applied and the species are driven mainly by the mole flux profile defined at the inlet along the channel length and by the very small diffusive force due to the concentration gradient induced by the initial condition. For a second case, a constant external electrical force $E_y = 100\,\mathrm{V\,m^{-1}}$ is applied along the $y$ direction perpendicular to the main flow direction which transports the positively charged species 2 ($Na^+$) towards the top boundary $BC_3$ and the negatively charged species 3 ($Cl^-$) towards the bottom boundary $BC_2$. As mentioned before, the bottom $BC_2$ and the top $BC_3$ boundaries for this case are treated as AEM and CEM respectively with membrane black-box model to remove ionic species from the flow channel.

For the numerical simulation, the height of the channel is resolved with 256 elements and the physical time step $\delta t = 7.82 \times 10^{-6}\,\mathrm{s}$ is chosen to maintain stability. With this time step, the maximum lattice velocity at the center of the channel at inlet is $v_m^* = 0.05$ for the physical velocity $v_m = 0.01\,\mathrm{m\,s^{-1}}$. The MRT collision operator with $D3Q19$ stencil is used here. Figure 6.10 shows the results of the numerical experiment by means of mole fraction of the third species ($Cl^-$) for three configurations of the spacer filaments in the case without external force. Due to the parabolic mole flux profile at the inlet, the ionic species are transported much faster along the center than at the channel wall boundaries (where zero velocity is assumed). For the channel with no filaments, the concentration profile follows the parabolic velocity profile of the background flow and the no-slip BC along the horizontal boundaries is accurately resolved. At outlet, the flow velocity increases due to simple equilibrium BC which extrapolates both species concentration and velocity from the fluid node to the boundary node. Notice that the concentration profile gets sharper only near the outlet due to velocity increase. Therefore. the channel length is increased by $H/2$ to reduce the influence of the outlet boundary in the inner domain. In future, this can be resolved by more sophisticated higher order outlet BC like non-equilibrium extrapolation BC [27, 61]. The evolution of the concentration profile for the species 3 ($Cl^-$) for the no filaments configuration along the center line for different points in time is shown in Figure 6.11. The subplot inside this figure shows that due to diffusion the concentration gradient gets sharper but the diffusive transport is much less than the convective transport.

It can been seen in the Figure 6.10 that with the presence of the spacer filaments in the channel, the diffusive rate of the species increases especially near the walls. The zigzag filament configuration accelerates the species transport in the area between filament and opposite wall. The flow distributions in centered and zigzag filaments are shown in Figure 6.12 with streamlines of velocity magnitude. It shows that even for very low inflow velocity of $0.01\ \mathrm{m\,s^{-1}}$, a small recirculation appears on the zigzag filaments configuration near the intersection of filament and wall. In the recirculation area, the flow velocity is very low and areas with low velocities are referred to as low flow zones. In these areas, the concentration of ionic species accumulates resulting in scaling in the flow channel. These results show that the expected advection-diffusion mechanism is accurately recovered by the presented multi-component LBM scheme. The proposed boundary conditions resolve no-slip as well as in- and outlet conditions accurately.

Figure 6.13 shows the results of the mole fraction of the positively charged

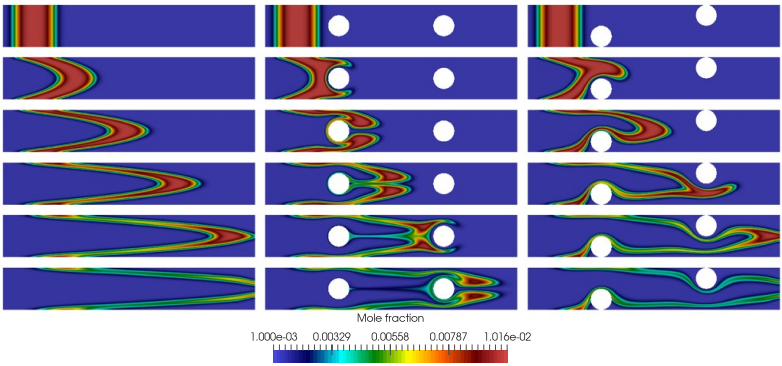



**Figure 6.10** Temporal evolution of the diffusion of the species 3 ($Cl^-$) in the Taylor dispersion experiment for $t = 0, 0.05, 0.1, 0.15, 0.2, 0.25$[s] (from top to bottom) with no external force for no filaments, centered filaments and zigzag filaments (from left to right).

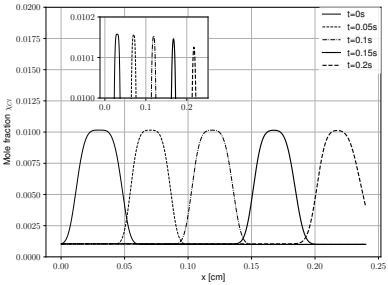



**Figure 6.11** Diffusion of the species 3 ($Cl^-$) in the Taylor dispersion experiment along the centerline of the channel over time without an external force for the no filaments configuration. Subplot shows the decay of the concentration profile peaks over time due to diffusion.

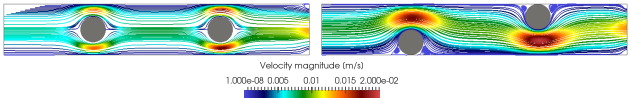
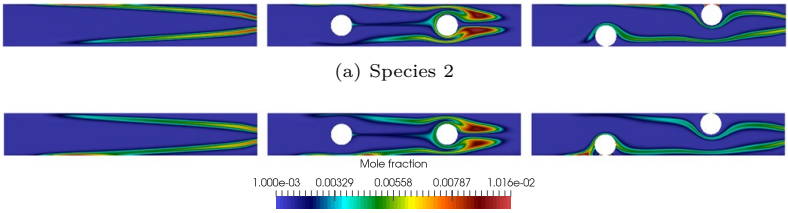



**Figure 6.12** Stream lines of flow distribution along the channel at $t = 0.25$ s for centered and zigzag filaments.

species 2 and the negatively charged species 3 in all three configurations with an external electrical force $E_y = 100\ \mathrm{V\,m^{-1}}$. In this case, the top and bottom wall are treated as CEM and AEM respectively with membrane black-box model. It can be clearly seen that the species 2

and the species 3 are transported in opposite direction i.e. towards the top and bottom boundary respectively. There is no significant difference between species 2 and species 3 in the plain channel and centered filaments configurations other than ionic species are transported in opposite direction. The increase in concentration of species 2 and species 3 on the top and the bottom membranes respectively for the centered filaments configuration at $t = 0.25$ s can be seen in Figure 6.14.




(a) Species 2

(b) Species 3

**Figure 6.13** Mole fraction of species 2 ($Na^+$) and species 3 ($Cl^-$) at $t = 0.25$[s] with black-box membrane model and a constant external electrical force $E_y = 100\ \mathrm{V\,m^{-1}}$ for three no filaments, centered filaments and zigzag filaments (from left to right).

On the other hand, in the zigzag filament configuration, a significant difference can be seen near the filament and membrane intersection. Due to the applied external electrical force, the species 2 and the species 3 are accumulated at the intersection of upper filament and lower filament respectively (see Figure 6.15). The concentration of species in this area decreases gradually with time due to membrane model at the top and bottom boundary. Even though the Figure 6.15 shows the results at $t = 0.25$ s, the concentration profile reaches the lower filament much earlier, when the concentration profile is still sharper. This results in a small deposition of species 3 ($Cl^-$) in front of the lower filament. Note that this concentration at the intersection is roughly 2 times the initial concentration and it grows with time and might affect the flow in the channel. As the concentration of species 2 and 3 increases near the top and bottom filament respectively, their concentration also decreases in the alternative filaments. The simulation gets unstable if this concentration falls below or equal to zero. In reality, the accumulation near the filaments is cleaned by switching the direction of current which alters the direction of ions transport which also swaps dilute and concentrate channels.

The membrane black-box model removes the ionic species from the flow channel depending on the transport number of certain species and the local

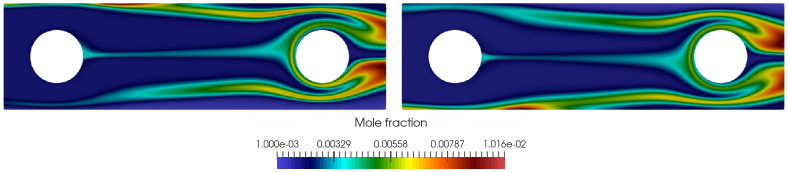



**Figure 6.14** Mole fraction of species 2 (left) and species 3 (right) at $t = 0.25$ s of centered filaments configuration with black-box membrane model and a constant external electrical force $E_y = 100\ \mathrm{V\,m^{-1}}$

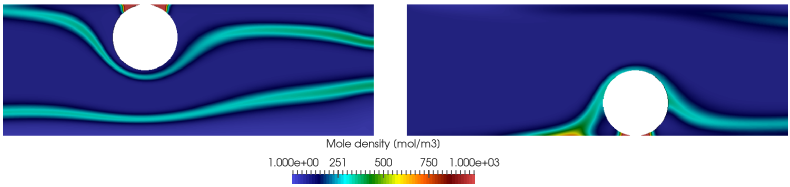
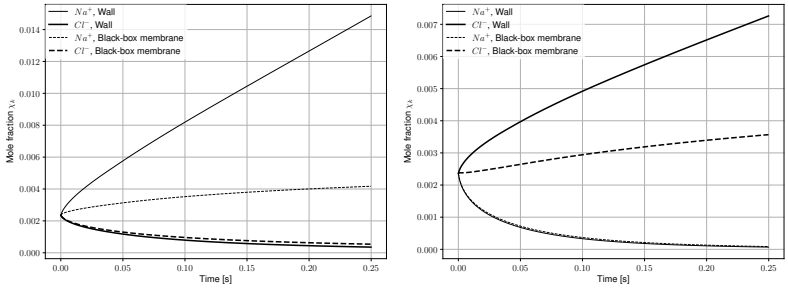



**Figure 6.15** Mole density of species 2 (left) and species 3 (right) near the top and bottom filament respectively at $t = 0.25$ s of zigzag filaments configuration with black-box membrane model and a constant external electrical force $E_y = 100\ \mathrm{V\,m^{-1}}$

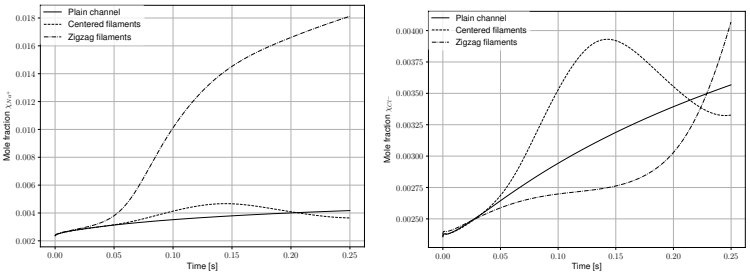



(a) Top boundary ($BC_3$)

(b) Bottom boundary ($BC_2$)

**Figure 6.16** Average mole fraction of species 2 ($Na^+$) and 3 ($Cl^-$) on the surface of the top (left) and bottom (right) boundary $BC_2$ over time for wall and black-box membrane on the top and bottom boundary with an external electrical force on the channel with no filaments.

current density. Here, the top and bottom boundary are treated as CEM and AEM resulting in a dilute flow channel. The behavior of the membrane black-box model is verified by comparing it with no-slip wall BC for the top and bottom in the flow channel with no filaments. With no-slip wall




(a) Species 2 on top surface

(b) Species 3 on bottom surface

**Figure 6.17** Average mole fraction of species 2 ($Na^+$) and species 3 ($Cl^-$) on the top and bottom surface for different configuration with black-box membrane model

BC, the ionic species that are transported towards the boundary by an external electrical force stay on the surface of the corresponding boundary. This can be observed in Figure 6.16, which shows the average mole fraction of species 3 on the surface of top and bottom boundary for no-slip wall and membrane black-box model. It can be seen that without a membrane black-box model, the mole fraction of the ionic species on the surface of the boundary increases drastically compared to the one with black-box model. Due to high transport number of species 3 ($Cl^-$) for the AEM, the average mole fraction of species 3 on the bottom surface is less than the average mole fraction of species 2 on the top surface. Furthermore, the effect of the black-box model on different channel configurations is shown in Figure 6.17. Due to the accumulation of ionic species near the filament/membrane intersection in the flow channel with zigzag filaments, the mole fraction of the corresponding species on the top and surface increases with time. These results prove that the membrane black-box model works as expected.

## 6.5 Electrical double layer

An electrical double layer (EDL) is a structure thats form due to redistribution of ions when an electrolyte solution comes in contact with a charged surface like an electrode. An EDL near a flat solid-liquid interface is illustrated in Figure 6.18. Immediately next to the charges, there exists a layer of ions which are strongly attracted to the surface and immobile. This layer is called compact layer. Due to electrostatic interaction, the

surface charges attract the oppositely charged ions or counterions in the solution and repels the corresponding same charged ions or coions. Thus, near the surface within the EDL, the net charge density is not zero so the solution is nonelectroneutral. From the compact layer, the net charge density gradually reduces to zero in the bulk solution where the solution is electroneutral. The layer in which the ions are mobile and less affected by electrostatic interaction are called the diffusive layer of the EDL. Note that the net charge of the counter ions in the EDL balances the net charges on the surface. The charges on the surface result in electrical surface potential and they are difficult to measure. However, the potential at the interface (shear plane between the compact layer and diffusive layer) is measurable and it is called zeta ($\zeta_s$) potential. Due to the non-zero net charge density near the surface, the electric potential gradient is formed in the diffusive layer of the EDL. Thus, the formation of the EDL involves an electrostatic interaction between the distribution of ions and the electric potential which makes this test case well suited to validate the coupled simulation of multicomponent LBM and LBM for electric potential using the coupling tool *APESmate* introduced in Section 5.5.

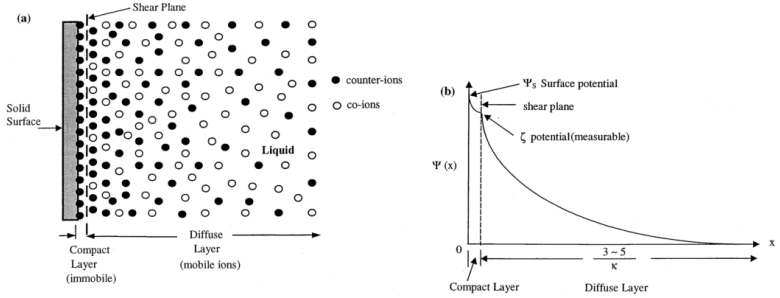



**Figure 6.18** Illustration of an EDL near a flat-solid interface. a) Ion distribution. b) electrical potential distribution [58].

According to the theory of electrostatics, the relationship between the net charge density $\rho^e = \mathcal{F} \sum_k z_k c_k$ and the electric potential distribution $\psi$ at any point in the solution can be described by the Poisson Eq. 2.64. Assuming the system is in thermodynamic equilibrium and the ionic distributions are not affected by fluid flows, the distribution of ions near the charged surface is given by the Boltzmann distribution

$$c_k = c_{k,\infty} exp\left(\frac{-z_k \mathcal{F} \psi}{RT}\right) \tag{6.10}$$

where $c_{k,\infty}$ is the concentration of ions in the bulk solution, $R$ is the gas constant and $T$ is the temperature. Note that this distribution is independent of the species molecular weight therefore it is valid only up to a certain ionic concentration. Also, the concentration of ions at the electrolyte/electrode interface increases with an increase in bulk concentration $c_{k,\infty}$ or the surface zeta potential $\zeta_s$. For symmetric 1:1 electrolyte solution like NaCl ($z_k = z$ and $c_{k,\infty} = c_\infty$), the net charge density $\rho^e$ can be written as

$$\rho^e = -2c_\infty z\mathcal{F} sinh\left(\frac{z\mathcal{F}\psi}{RT}\right). \tag{6.11}$$

Substituting this in Eq. 2.64 results in the Poisson-Boltzmann equation for the electric potential distribution in dilute electrolyte solutions

$$\nabla^2\psi = \frac{2c_\infty z\mathcal{F}}{\varepsilon} exp\left(\frac{z\mathcal{F}\psi}{RT}\right). \tag{6.12}$$

Furthermore, if $z\mathcal{F}\psi/RT$ is small or in other words, if the potential is small $|\psi| \leq 25$ mV then Eq. 6.12 can be linearized into

$$\nabla^2\psi = \frac{2z^2\mathcal{F}^2 c_{i,\infty}}{\varepsilon RT}\psi = \kappa^2\psi \tag{6.13}$$

where $\kappa = 1/\lambda_D = \sqrt{\frac{2z^2\mathcal{F}^2 c_{i,\infty}}{\varepsilon RT}}$ is the inverse of the Debye length $\lambda_D$ which is also called as the Debye-Huckel parameter. $\lambda_D$ is the characteristics thickness of the EDL. It is independent of the surface properties and dependent only on the properties of the electrolyte solution such as valance of ions and bulk concentration. Since the Debye length is inversely proportional to the bulk concentration, it decreases with increase in concentration (see Table 6.2). Due to the very small thickness of the EDL, its effects are only considered in micro- and nano-channels and usually neglected in macro-channels. Therefore, in this validation, a 2D channel with the height $H = 50$ nm in y-direction and periodic in x-direction as shown in Figure 6.19 is considered. The height of the channel spans from $-H/2$ to $H/2$. The top and bottom surface are heterogeneously charged with a potential of $\zeta_s = -5$ mV and $\zeta_s = 5$ mV respectively. With this boundary condition, the analytical solution for the electrical potential distribution in the channel is given as

$$\psi(y) = \left(\frac{cosh[\frac{y}{\lambda_D}]}{cosh[\frac{H}{2\lambda_D}]}\right)\zeta_s \quad \zeta_s = \begin{cases} 5 \text{ mV} & \text{for } y < 0 \\ -5 \text{ mV} & \text{otherwise} \end{cases}. \tag{6.14}$$

For multicomponent flows, the top and bottom surface are treated as no-slip walls using simple bounce back BC. A mixture of aqueous $NaCl$

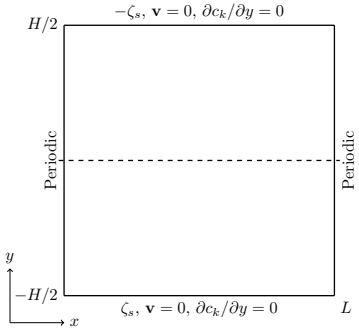



**Figure 6.19** Simulation setup for electrical double layer test case.

solution of different bulk concentrations 10, 100 and 500 $\mathrm{mol\,m^{-3}}$are investigated here. Each of these concentrations is simulated with ideal and nonideal multicomponent model. For the ideal model, the binary Maxwell-Stefan diffusivity coefficients for these bulk concentrations are evaluated using Eq. 2.30 and listed in Table 6.2. For nonideal model, the concentration dependent diffusivity coefficients are evaluated using Eq. 2.30 at every element using the local concentration. Additionally, the activity coefficients in the thermodynamic factor Eq. 2.23 are also computed from the local concentration using the NRTL-model [100]. A NRTL (Non Random Two Liquid) model is an activity coefficient model that correlates the activity coefficients $\gamma_k$ of a component with its mole fraction $\chi_k$ in a concerned liquid phase.

| $c_\infty$ | $\lambda_D$ | $\mathcal{D}_{1,2}$ | $\mathcal{D}_{1,3}$ | $\mathcal{D}_{2,3}$ |
|---|---|---|---|---|
| $[\mathrm{mol\,m^{-3}}]$ | [nm] | $1 \times 10^{-9}[\mathrm{m^2\,s^{-1}}]$ | | |
| 10 | 3.04 | 1.334 | 2.064 | 0.0077 |
| 100 | 0.96 | 1.3157 | 2.0974 | 0.0295 |
| 500 | 0.43 | 1.2508 | 2.0811 | 0.0848 |

**Table 6.2** EDL thickness and binary Maxwell-Stefan diffusivity coefficients for different bulk concentrations

The properties of an electrolyte solution like molecular weights of species, total mass density, kinematic viscosity and bulk viscosity are the same as listed in Table 6.1. The molar concentration of species 1 and total concentration are computed from the bulk concentration of $NaCl$ ($c_\infty =$

$c_2 = c_3$) using

$$c_1 = \frac{(\rho_t - c_\infty(M_2 + M_3))}{M_1}, \quad c_t = c_1 + 2c_\infty \tag{6.15}$$

respectively. Other properties are

- the permittivity of the medium $\varepsilon = 7.083 \times 10^{-10}\ \mathrm{J\,K^{-1}\,mol^{-1}}$,
- the Faraday constant $\mathcal{F} = 96\,485.3365\ \mathrm{C\,mol^{-1}}$ and
- the gas constant $\mathcal{R} = 8.314\,462\,1\ \mathrm{kg\,m^2\,s^{-2}\,mol^{-1}\,K^{-1}}$.

The species 1, 2 and 3 corresponds to $H_2O$, $Na^+$ and $Cl^-$ respectively. For the coupled simulation, the multicomponent LBM is initialized with uniform ionic species distribution of bulk concentration and fluid at rest

$$\chi_2 = \chi_3 = \frac{c_\infty}{c_t}, \quad \chi_1 = 1 - \chi_2 - \chi_3$$
$$\boldsymbol{v}_1(\boldsymbol{x}) = \boldsymbol{v}_2(\boldsymbol{x}) = \boldsymbol{v}_3(\boldsymbol{x}) = 0$$

and the LBM for electric potential is initialized with zero potential ($\psi(t = 0) = 0$). For multicomponent LBM, the MRT collision operator is used and for LBM for electric potential, the BGK collision operator is used. For both models, the $D2Q9$ layout is used. The potential diffusivity $\gamma = 0.167$. The physical time step is meaningful only for multicomponent flows because the LBM for electric potential equation is solved for steady state at each time step of the multicomponent LBM. The analytical solutions for ions distribution Eq. 6.10 and the potential distribution Eq. 6.14 are steady state solutions, so coupled simulations are performed until the concentration of the multicomponent flow reaches the steady state.

Steady state concentration profiles of species 2 and 3 along the entire height for the applied potential drop of 100 mV (i.e. $\zeta_s = 50$ mV at $y = -H/2$ and $\zeta = -50$ mV and $y = H/2$) for different bulk concentrations are shown in Figure 6.20. From top to bottom, the concentration is 10 $\mathrm{mol\,m^{-3}}$, 100 $\mathrm{mol\,m^{-3}}$ and 500 $\mathrm{mol\,m^{-3}}$. These results show the change in concentration profile especially near the interface for different concentrations. For all profiles, the formation of the EDL near the interface can be clearly seen and in this region, the local electroneutrality does not apply. The trend observed here is consistent with those explained in the literature of electrochemical systems [36, 72]. At low concentration of 10 $\mathrm{mol\,m^{-3}}$, a very small discrepancy between ideal and nonideal model can be seen only in the concentration profiles of co-ion[1] (which is

[1] charge of the ion similar to the surface charge

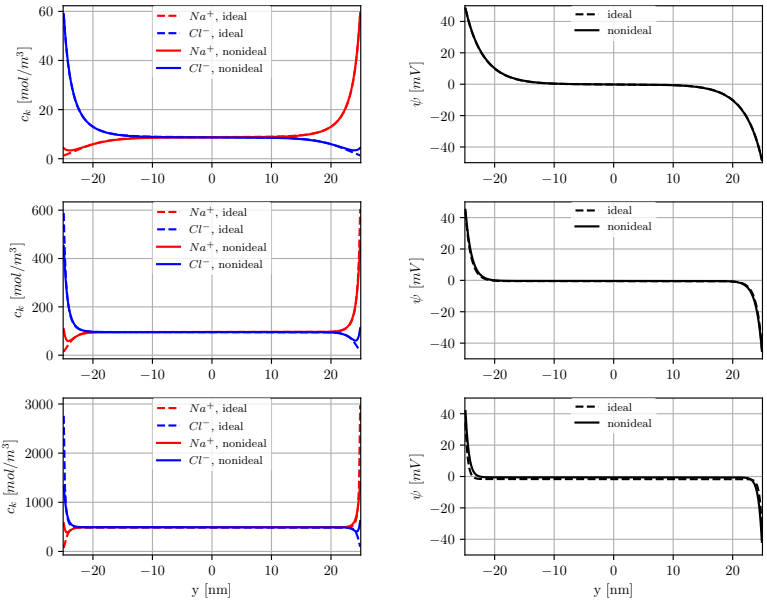
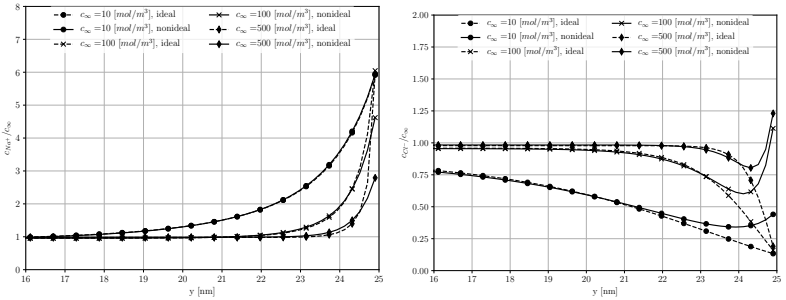



**Figure 6.20** Steady state concentration profiles of species 2 ($Na^+$) and species 3 ($Cl^-$) on the left and potential distribution on the right with applied potential drop of 100 mV for both ideal and nonideal multicomponent LBM model for different concentration from top to bottom 10 mol m$^{-3}$, 100 mol m$^{-3}$ and 500 mol m$^{-3}$.

species 2 at $y = -H/2$ and species 3 at $y = H/2$) at the interface. The concentration profiles of counter-ion[1] and potential distribution profiles of ideal and nonideal simulation are indistinguishable. However increasing the concentration, the nonideal model forms a bulge like structure near the interface because of the nonideal multicomponent effects where counter-ions attract co-ions at the interface. Additionally, the concentration profile of counter-ion especially at the interface differs between ideal and nonideal model. The concentration of counter-ions at the interface predicted by the ideal model is higher than for the nonideal model. At a high concentration of 500 mol m$^{-3}$, the concentration of counter-ions at the interface predicted by ideal model is roughly twice the nonideal model prediction. For both models as the bulk concentration increases, the concentration of counter-ion at the interface differs, i.e. species 3 at

[1] charge of the ion different than the surface charge

$y = -H/2$ and species 2 at $y = H/2$ are not same. This is due to the different molecular weight of species 2 and 3. The molecular weight of the species corresponds to the partial molar volumes ($\tilde{V}_k = M_k/\rho_k$) of the species. Since the partial molar volume of species 2 ($Na^+$) is less than that of species 3 ($Cl^-$), species 2 occupies more space at the interface than species 3. Even though the concentration of species at the interface differs, the potential profile is symmetric because the total concentration of species 2 in the EDL of $y = H/2$ is same as the total concentration of species 3 in the EDL at $y = -H/2$, i.e. the areas under the concentration profile in the EDL are equal. It can be also seen that away from the EDL, the local electroneutrality applies and there are no differences between ideal and nonideal models in the concentration profiles. On the other hand, the potential profile between the two models differs slightly as the bulk concentration increases. The potential profile of the nonideal model preserves the symmetry and the zero potential in the bulk where as the ideal model deviates slightly from zero potential in the bulk due to the fixed binary Maxwell-Stefan diffusivity coefficient. Its worth mentioning at this point that even though the steady state solution of ideal and nonideal model are quite the same, their transient behavior was different.




(a) Species 2 ($Na^+$) (b) Species 3 ($Cl^-$)

**Figure 6.21** Normalized concentration ($c_k/c_\infty$) profiles of species 2 on the left and species 3 on the right at steady state for the applied potential drop of 100 mV and different bulk concentrations near the interface up to 9 nm.

For better comparison, normalized concentration ($c_k/c_\infty$) profiles of species 2 and species 3 for ideal and nonideal multicomponent model for different concentrations and the potential drop of 100 mV are shown in Figure 6.21. It can be seen that the normalized concentration of species 2 at the interface remains the same for all three bulk concentrations with

the ideal model where as the nonideal model shows a significant drop in the normalized concentration as the bulk concentration increases. The difference between ideal and nonideal model is seen only very close to the interface. Even at the high concentration 500 mol m$^{-3}$ the difference is only within 2 nm from the interface. Similarly, the difference in concentration profiles of species 3 is also near the interface. The nonideal model forms a bulge structure near the interface due to inclusion of thermodynamic factor in the macroscopic force term. These results shows that coupled simulation works as expected and the next step is to validate the numerical result with analytical solution.

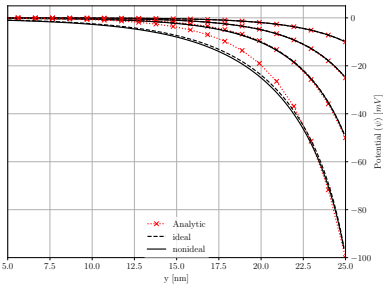
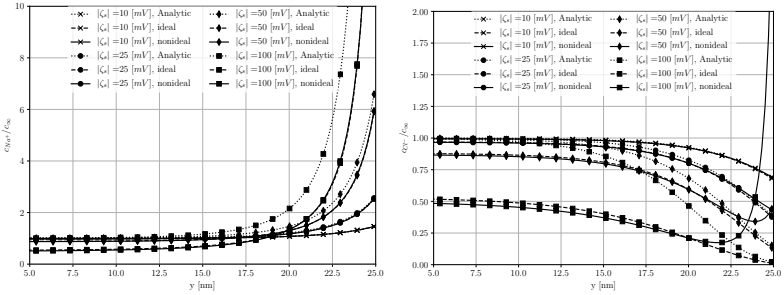



**Figure 6.22** Comparison of potential distribution near the interface up to 20 nm between analytical solution and coupled numerical simulation of ideal and nonideal multicomponent model with electric potential equation for different zeta potential $\zeta_s$ and bulk concentration $c_\infty = 10$ mol m$^{-3}$ at steady state.

In order to validate the coupled simulation, a dilute aqueous $NaCl$ solution with concentration of 10 mol m$^{-3}$ is considered at first. The potential distributions near the interface at steady state of the coupled simulation with the ideal and the nonideal multicomponent model, along with the analytical solution for different zeta potentials are shown in Figure 6.22. The electric potential in the solution decreases nonlinearly near the interface, and at the middle (bulk) of the channel the electrical potential becomes zero. Even though the analytical solution of the linearized Poisson-Boltzmann equation is valid only for zeta potential $\leq |25|$ mV, for low concentration of 10 mol m$^{-3}$, the numerical simulation of both ideal and nonideal model matches perfectly with the analytical solution up to a zeta potential of $|50|$ mV. However, with an increase in zeta potential to $|100|$ mV, a discrepancy occurs between numerical and analytical solution.

The electric potential distribution shows only a very small discrepancy between ideal and nonideal simulations for high zeta potential $\zeta_s = |100|$ mV.




(a) Species 2 ($Na^+$) (b) Species 3 ($Cl^-$)

**Figure 6.23** Normalized concentration profiles of species 2 (left) and species 3 (right) up to 20 nm distance from interface for different zeta potential $\zeta_s$ and bulk concentration $c_\infty = 10\ \mathrm{mol\,m^{-3}}$ at steady state. Solid lines are nonideal multicomponent model, dashed lines are ideal multicomponent model and dotted lines are analytical solution

Yet, in the Figure 6.23b which shows the normalized concentration profile of species 3 (co-ions) near the interface of negatively charged potential, the effect of nonideal simulation can seen even at zeta potential $\zeta_s = |25|$ mV. As the concentration of species 2 (counter-ions) increases at the interface with an increase in zeta potential, it attracts more of species 3 (co-ions) resulting in a sudden increase of species 3 concentration at the interface and forms a bulge near the interface. This is due to nonideal multicomponent effects where diffusivity and activity coefficients are calculated from the local concentration. This nonideal effect is observed only in the concentration profile of species 3 (co-ion) and not on the concentration profile of species 2 (counter-ion) where ideal and nonideal results are indistinguishable as shown in Figure 6.23a. However, the discrepancy between numerical simulation and analytical solution occurs for zeta potential $> |25|$ mV and it increases with increase in zeta potential. This proves that the linearized analytical solution is valid only for zeta potential $< |25|$ mV.

The electric potential distribution of ideal, nonideal and analytical solution for different zeta potentials and $c_\infty = 100\ \mathrm{mol\,m^{-3}}$ are shown in Figure 6.24. It can been seen that with increasing the concentration of $NaCl$ solution to $100\ \mathrm{mol\,m^{-3}}$, a discrepancy between ideal and nonideal model on the potential distribution can be seen for zeta potential $> |25|$ mV. Still for zeta potential $\leq |25|$ mV, the numerical results of both ideal and

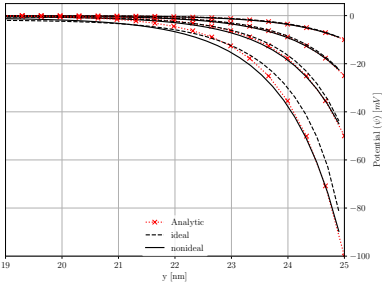
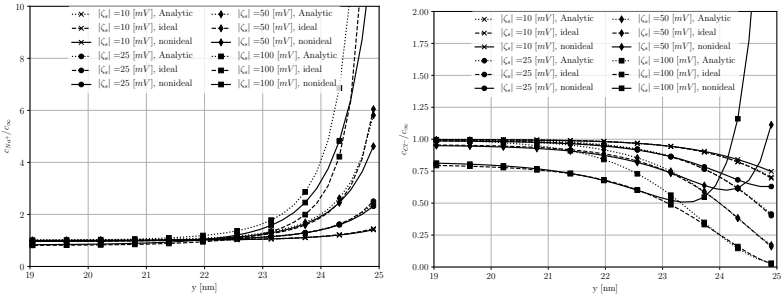



**Figure 6.24** Comparison of potential distribution near the interface up to 6 nm between analytical solution and coupled numerical simulation of ideal and nonideal multicomponent model with electric potential equation for different zeta potential $\zeta_s$ and bulk concentration $c_\infty = 100\ \mathrm{mol\,m^{-3}}$ at steady state.

nonideal model matches perfectly with a linearized analytical solution. The electric potential distribution within the EDL of nonideal model is less steep than of the ideal model and also the electric potential of the nonideal at the interface matches well with the analytical solution. It seems that with increasing concentration in the EDL, the potential distribution of the ideal model produces asymmetry due to the species molecular weights which results in a non-zero potential in the bulk. But considering nonideal effects the profile becomes symmetric and the potential drops to zero in the middle of the channel. This shows the importance of applying a nonideal model for the aqueous NaCl liquid mixture.

Figure 6.25 shows the normalized concentration profiles of species 2 and 3 near the interface up to 6nm for concentration 100 $\mathrm{mol\,m^{-3}}$ and different zeta potential. With 100 $\mathrm{mol\,m^{-3}}$, a difference between ideal and nonideal modeal can be seen in the distribution of species 2 (counter-ion) near the interface. This difference increases as the zeta potential increases. It can been seen that the concentration of species 2 at the interface predicted by the nonideal model is less than by the ideal model. For $\zeta_s = |50|$ mV, the concentration predicted by nonideal is $\approx 4 \times c_\infty$ and ideal is $\approx 5.8 \times c_\infty$. The distribution of species 3 (co-ion) near the interface is the same as before except for nonideal, the magnitude of the concentration at the interface is higher and also the bulge is bigger. Also, the nonideal effects start to occur even for the very low zeta potential of $|10|$ mV. Note that the difference between ideal and nonideal is only near the interface and in the bulk they are same.

The multicomponent LBM model implementation in *Musubi* can handle




(a) Species 2 ($Na^+$)

(b) Species 3 ($Cl^-$)

**Figure 6.25** Normalized concentration profiles of species 2 (left) and species 3 (right) up to 6 nm distance from interface for different zeta potential $\zeta_s$ and bulk concentration $c_\infty = 100\ \mathrm{mol\,m^{-3}}$ at steady state. Solid lines are nonideal multicomponent model, dashed lines are ideal multicomponent model and dotted lines are analytical solution

arbitrary number of components and to demonstrate that, a liquid mixture of $NaCl$ and $KCl$ solution is simulated. Here, the species 1, 2, 3 and 4 corresponds to $H_2O$, $Na^+$, $Cl^-$ and $K^+$ respectively. The concentration of the $NaCl$ and $KCl$ are 90 $\mathrm{mol\,m^{-3}}$ and 10 $\mathrm{mol\,m^{-3}}$ respectively with total ionic species concentration of 100 $\mathrm{mol\,m^{-3}}$. Thus, the bulk concentration of ionic species $Na^+$, $Cl^-$ and $K^+$ are $c_{Na^+,\infty} = 90\ \mathrm{mol\,m^{-3}}$, $c_{Cl^-,\infty} = 100\ \mathrm{mol\,m^{-3}}$ and $c_{K^+,\infty} = 10\ \mathrm{mol\,m^{-3}}$ respectively. The multicomponent model is initialized with uniform distribution of ionic species with their corresponding concentrations. Since the NRTL model was limited to binary electrolyte, this simulation is performed only with the ideal model. The Maxwell-Stefan binary diffusivity coefficients for species 1, 2 and 3 are the same as before for the concentration of 100 $\mathrm{mol\,m^{-3}}$. For species 4, they are chosen as $\mathcal{D}_{1,4} = \mathcal{D}_{1,3}$, $\mathcal{D}_{2,4} = \mathcal{D}_{2,3}$ and $\mathcal{D}_{3,4} = \mathcal{D}_{2,3}$. The molecular weight of species 4 is $M_4 = 39.0983 \times 10^{-3}\ \mathrm{kg\,mol^{-1}}$. Figure 6.26 shows the concentration profiles of all three ionic species at steady state for the potential drop of 100 mV.

These coupled simulation results show that the presented coupling tool *APESmate* can be used to perform coupled multiphysics simulations which are required to simulate electro-convection transport of ions in the ED process. In most previous works, the Nernst-Planck equations were used for multicomponent transport, only few deployed the Maxwell-Stefan equations but only for ideal mixture. From these results, it can be concluded that

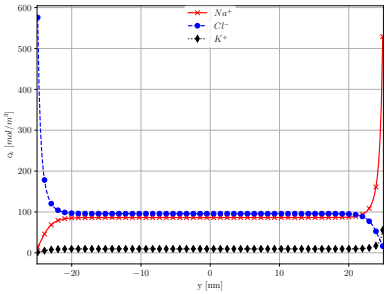



**Figure 6.26** Steady state concentration profiles of species 2 ($Na^+$) , 3 ($Cl^-$) and 4 ($K^+$) for total ionic species concentration of 100 mol m$^{-3}$ and potential drop 100 mV.

as the concentration of ionic species increases, the nonideal model shows significantly different behavior from the ideal model. The concentration of ionic species near the interface can increase either by initial concentration or by applied potential drop. A high initial concentration 500 mol m$^{-3}$ is close to the salinity of seawater and for this concentration, the nonideal effects are higher near the interface. In the ED stack, the concentration of ionic species in the concentrate channel increases even higher than the inlet seawater concentration. Thus, for this application, nonideal effects must be considered for the multicomponent flow simulations.

## 6.6 Conclusion

In this chapter, the numerical methods presented in Chapter 3 which are implemented in the LBM solver *Musubi* are validated against analytical solution or laboratory experiment. At first, the individual physical system were validated:

- The single component LBM is validated using a well-known Poiseuille flow test case. The numerical results of velocity profile across the channel height and the pressure drop across the channel length are compared to the analytical solution and also a mesh convergence analysis was performed to show the accuracy of the implemented BC.

- The convection in multicomponent LBM is validated also with Poiseuille flow but the diffusion phenomena modeled by Maxwell-Stefan equations are validated using experimental data of the Stefan tube test case.

- The LBM for electric potential is validated against analytical solution of a concentric cylinder test case. The cylinders are approximated by q-values and higher order non-equilibrium extrapolation BC was used to treat the Dirichlet potential boundaries.

After the validation of the individual physical systems, the coupled multiphysical system was verified and validated.

- The coupled multicomponent LBM and membrane black box model is verified using the Taylor dispersion test case. In this test case, the flow is initialized with non-uniform concentration in a stripe near the inlet and the ions are transported along the length due to flow velocity defined at inlet. A membrane black box model was used to treat the AEM and CEM on the bottom and top boundary respectively. Since no analytical solution was available for this case, the results are only qualitatively verified by comparing the transport of ions across the membrane with a no-slip wall boundary. The numerical results prove that the implementation of the membrane black-box BC works as expected.

Finally, the coupled multicomponent LBM and the LBM for the electric potential is validated.

- The formation of EDL test case is used to validate both ideal and nonideal multicomponent LBM models. This test case was chosen for the availability of analytical solution for distribution of ions and the potential distribution. Both analytical solutions are valid only under certain conditions:
  - The Boltzmann distribution for distribution of ions near the electrode/electrolyte interface is valid only when the system is at thermodynamic equilibrium and when ions distributions are not affected by the fluid flow.
  - The analytical solution of the linearized Poisson-Boltzmann equation is valid only for surface zeta potential $\leq |25|$ mV.

  Under these conditions, the numerical results agree pretty well with analytical solutions. The comparison of ideal and nonideal model revealed that the nonideal effects occur when the concentration of counter-ions in the EDL region is higher than 50 $\mathrm{mol\,m^{-3}}$. For the co-ion concentration near the EDL region, the nonideal effects occur even at low concentrations. Since in the ED stack, the concentration of salt at inlet is around 500 $\mathrm{mol\,m^{-3}}$ and in concentrate channels, the concentration of salt increases even further along the length, it is

necessary to simulate this system with the nonideal multicomponent model. These results show that the presented multicomponent LBM with external electrical force predicts the behavior of the physical system very well. Also, the coupling tool *APESmate* developed enables us to perform coupled multiphysics simulations with high numerical accuracy and can be used to simulate the multiphysics in the ED process.

# 7 Results

This chapter presents large-scale flow simulations with woven and nonwoven spacer geometries. For both spacer geometries, the angle between the filament and the flow direction is varied to study its effect on the pressure drop across the channel and the flow distribution in the channel. In total five distinct spacer configurations are investigated and for each the mean inflow velocity is varied from 0.01 $\mathrm{m\,s^{-1}}$ to 0.8 $\mathrm{m\,s^{-1}}$. Before the velocity study, a mesh convergence analysis is performed on one spacer configuration to identify the resolution required to produce mesh independent results. The pure hydrodynamic flow simulations are performed at first on spacers with periodic width, i.e periodic BC in $z$-direction to identify the optimal spacer configuration. Then, the effect of sealed corner in the ED stack is investigated on nonwoven spacer geometries for two different flow velocities. The simulations on periodic width spacers are run on the Horus cluster in the University of Siegen. But the simulations near the sealed corner have a much larger mesh of around 500 million elements and they are run on the Hermit system, HLRS, Stuttgart. Next, the multicomponent flow simulations are performed on the five distinct spacer configurations for one specific inflow velocity with a constant electrical force and membrane black-box model to identify the optimal spacer design with respect to ions transport through the membranes. Finally, this chapter is concluded with the simulation results of a repeating single ED unit consisting of a dilute and a concentrate channel.

## 7.1 Flow simulations with spacers

In this section, the spacer geometries used in the flow channel of the ED process are investigated. As mentioned in Chapter 1, in the ED process, the spacers are placed between the IEMs to create the flow channel and maintain mechanical stability of the stack. The spacer design is known to influence the pressure drop across the channel and the ions transport in the channel and through the IEM. Thus, the optimal spacer design should have a low pressure drop across the channel and also uniform flow distribution with reduced low flow zones (extremely small velocity areas). Even though there are several spacer parameters to define the spacer geometry (see

Section 7.1.1), the investigations in this work are restricted to the woven and nonwoven spacers with different inner angle between the filament and the main bulk (feed) flow direction.

At first, the mesh convergence study is performed on the woven spacer geometry with hydrodynamic angle $\beta = 90^{\circ}$ to determine the required mesh resolution which provides good numerical accuracy with reasonable runtime in large scale systems. Then, the effect of different spacer designs on the velocity flow distribution in the channel and the pressure drop across the channel are studied using pure hydrodynamic flow simulations. The previous investigations performed together with Johannink [39] were focused only on the pressure drop across the channel for various spacer designs to develop the pressure drop prediction model. So, this work focuses on the flow distribution inside the channel for different spacer geometries. The initial investigations are performed in the middle of the flow channel far from the corners. Near the corner the flow velocity is expected to be low which is the main cause for scaling and fouling effect in the flow channel. Therefore, the nonwoven spacers with different angles between filaments are investigated near the sealed corner of the ED stack. Finally, the effect of spacer design on the transport of ionic species in the seawater is studied using the multi-component flow simulations with constant external electrical force.

Most of the simulations are computed on the supercomputer Hazel Hen in HLRS, Stuttgart and few of them on the HorUS cluster of the University of Siegen. The Hazel Hen is based on Intel Haswell Processor and the Cray Aries network that provides 7712 compute nodes with 24 processors per node. The memory per node is 128 GB. The HorUS cluster in the University of Siegen has 136 computation nodes with 12 processors per node. Due to the limit of virtual memory and disk space in HorUS, the supercomputer Hazel Hen was used for most of the large simulations with problem sizes large than 100 million elements.

### 7.1.1 Spacer geometry

In this section, the different spacer geometries investigated in this work are introduced. Additionally, the simulation setup including tracking areas on which the physical quantities like pressure ($P$) and velocity ($\boldsymbol{v}$) are measured over time and space are illustrated using figures. The spacers are composed of filaments with rounded cross-section. The spacer design can be categorized by the arrangement of these filaments as woven and nonwoven spacers. In the woven spacer, the filaments are interweaved by running one over the other where as in the nonwoven spacer the filaments

are arranged on top of each other.

The geometrical parameters of the spacer and the simulation setup are illustrated using the nonwoven spacer in Figure 7.1. The main flow is along the $x-$ direction and the two layers of parallel filaments are oriented along the $xz-$plane. Within this plane, two angles characterize the orientation of the filaments: 1) The angle $\alpha$ between the filaments in different layers, which describes the change in direction of the fluid as it flows along the channel and 2) The inner angle between the filament and the main feed flow direction $\beta$. In some publications [18, 82], the angle $\alpha$ is referred to as the hydrodynamic angle. In addition to these angles, the geometry of spacer design is completely defined by the diameter of the spacer filaments $d_f$, the orthogonal distance between two consecutive parallel filaments $l_m$, the height of the channel $h_{ch}$, the length of the channel $l_{ch}$ and the width of the channel $w_{ch}$. Here, the diameter of the filament $d_f$ running on top and bottom are considered to be equal, which leads to a channel height, $h_{ch} = 2d_f$ and the spacer mesh is considered to be a rhombus. Figure 7.1 also shows the zigzag arrangement of the filaments in the cross-section of $xy$-plane.

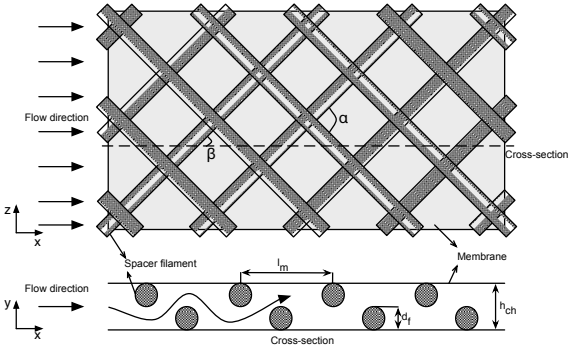



**Figure 7.1** Geometrical parameters of nonwoven spacer

For spacer-filled flow channels, the Reynolds number $Re_{sp}$ is defined in terms of the hydraulic diameter of the channel $d_h$ as characteristics height and the mean inflow velocity $\bar{v}_{in}$ as characteristic velocity [18]

$$Re_{sp} = \frac{\bar{v}_{in} d_h}{\nu}. \tag{7.1}$$

The hydraulic diameter $d_h$ for the spacer-filled channel is computed by

$$d_h = \frac{4\epsilon}{\frac{2}{h_{ch}} + (1-\epsilon)S_{vsp}} \tag{7.2}$$

where $\epsilon$ is the voidage and $S_{vsp}$ is the specific surface of the spacer. Voidage is the most important characteristics of spacers since spacers occupy quite a large space in the spacer-filled flow channel resulting in less free space for the fluid flow and it is defined by [18]

$$\epsilon = 1 - \frac{Spacer\ volume}{Total\ volume}. \tag{7.3}$$

For rhomboid type spacers, the voidage and the specific surface of the spacer are given as

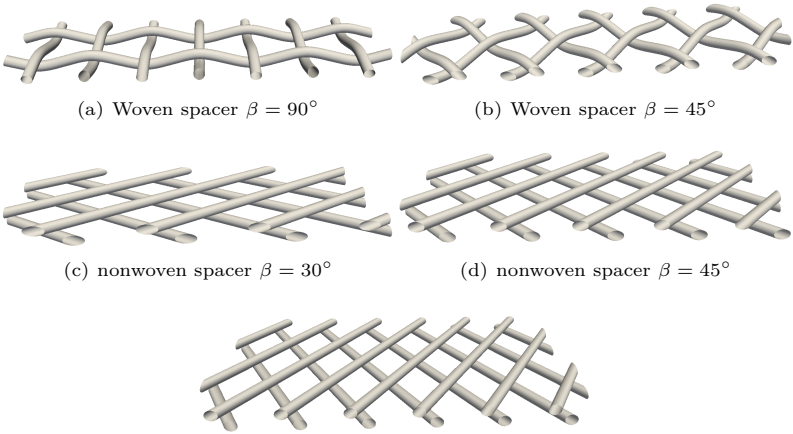
$$\epsilon = 1 - \frac{\pi d_f^2}{2 l_m h_{ch} sin(\alpha)} \tag{7.4}$$

and

$$S_{vsp} = \frac{4}{d_f}. \tag{7.5}$$

In this work, five distinct spacer configurations are considered: Woven spacer with an angle $\beta$ of 90° and 45° and nonwoven spacer with an angle $\beta$ of 30°, 45° and 60°. Figure 7.2 shows the woven and the nonwoven spacers with different angle $\beta$. The nonwoven spacers are configured such that the angle $\alpha$ is twice the angle $\beta$ i.e. $\alpha = 2\beta$, while in the woven spacer, the angle $\alpha$ is fixed to 90° due to interweave arrangement of filaments. For all configurations, the diameter of the filament and the distance between the filaments are fixed to $d_f = 0.02$ cm, $l_m = 0.1$ cm respectively which are similar to the woven spacer introduced in Section 5.3.1. Since $d_f$ and $l_m$ are fixed, the hydraulic diameter depends only on the angle $\alpha$. The hydraulic diameter $d_h$ for the hydrodynamic angles $\alpha = 60°$, 90° and 120° are 0.03795 cm, 0.04141 cm and 0.03795 cm respectively. Since for the chosen configurations, the hydraulic diameters are almost close to the channel height i.e. $d_h \approx h_{ch}$, the hydraulic diameter in the Reynolds number is replaced by the channel height as

$$Re_{sp} = \frac{\bar{v}_{in} h_{ch}}{\nu}. \tag{7.6}$$



(a) Woven spacer $\beta = 90°$ (b) Woven spacer $\beta = 45°$

(c) nonwoven spacer $\beta = 30°$ (d) nonwoven spacer $\beta = 45°$

(e) nonwoven spacer $\beta = 60°$

**Figure 7.2** Woven and nonwoven spacer geometries

#### 7.1.1.1 Simulation setup

The simulation domain and the boundary conditions are depicted in Figure 7.3 using woven spacer $\beta = 90°$. For better comparison of the simulation results, the length of the simulation domain for all spacer configurations is fixed to $l_{ch} = 0.7$ cm. However, due to design constraint between the width of the channel and the angle $\beta$ or $\alpha$, the width $w_{ch}$ increases with an increase in angle $\beta$ resulting in a larger computational area (more number of fluid elements). The simulation domain is periodic along the $z-$direction. Therefore, the width of the channel $w_{ch}$ is chosen such that the $xy-$plane cross section at $z = 0$ and $z = w_{ch}$ are exactly the same. The width of the nonwoven spacer is chosen to be twice the periodic width to show that the flow is symmetric between the periodic planes. The values of the width $w_{ch}$ of each spacer configuration are listed in Table 7.1.

| Spacer | $\beta$ | Width [cm] |
|---|---|---|
| woven | 45° | 0.15 |
| woven | 90° | 0.20 |
| nonwoven | 30° | 0.23 |
| nonwoven | 45° | 0.28 |
| nonwoven | 60° | 0.40 |

**Table 7.1** Width of the flow channel for different spacer configuration

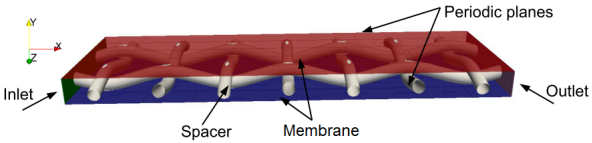



**Figure 7.3** Simulation domain of the woven spacer $\beta = 90°$ and the boundary conditions

The initial and boundary conditions are dependent on the physical system. In this section, the simulation parameters, initial conditions and boundary conditions used for the incompressible flow simulations (pure hydyodynamics) are given and for multi-component flow simulations, they are given later in Section 7.1.3. The physical properties of the fluid are the density $\rho = 998.2071\ \mathrm{kg\,m^{-3}}$ and the kinematic viscosity $\nu = 1.082 \times 10^{-6}\ \mathrm{m^2\,s^{-1}}$. The lattice parameters: the lattice velocity and density are fixed to $v_L = 0.02$ and $\rho_L = 1$ respectively. The relaxation parameter $\lambda_\nu$ is computed from the kinematic viscosity $\nu$ and the physical time step $\delta t$ using Eq. 3.22. The physical time step $\delta t$ is computed from the physical mean inflow velocity $\bar{v}_{in}$, the physical element size $\delta x$ and the lattice velocity $v_L$ using Eq. 3.89. For all flow simulations with spacer geometries, the lattice Mach number is fixed to $Ma_L = 0.035$. For the incompressible flow simulations, the MRT collision model and the $D3Q19$ layout are used.

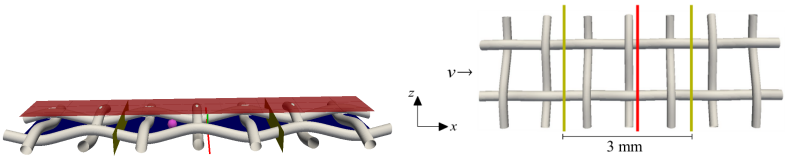
The initial conditions are specified as $\boldsymbol{v}(\boldsymbol{x}, t = 0) = 0$ and $P(\boldsymbol{x}, t = 0) = P_{atm}$, where $P_{atm}$ is the atmospheric pressure. From this initial state, simulations are run up to 1 s. The simulation is terminated if the simulation reaches steady state i.e. the difference in the average of pressure and velocity for 1000 time steps with the current time step is less than or equal to $1 \times 10^{-10}$. At inlet $x = 0$, the mean inflow velocity ($\bar{v}_{in} = \frac{2}{3} v_m$) is specified with parabolic velocity profile

$$v_x(\boldsymbol{x}.t) = \frac{8 v_m}{H^2} y(y - H), \quad v_y(\boldsymbol{x}, t) = v_z(\boldsymbol{x}, t) = 0 \tag{7.7}$$

using velocity bounce back BC Eq. 3.32. $v_m$ is the maximum velocity at the center of the parabolic profile. At outlet $x = l_{ch}$, the pressure ($P_o = P_{atm}$) is prescribed using pressure extrapolation BC, Eq. 3.33. The top ($y = h_{ch}$) and the bottom ($y = 0$), corresponding to the membranes, are treated with no-slip wall BC. Finally, the spacer geometry is approximated by q-values to improve the accuracy of the simulation and treated with higher order wall BC, Eq. 3.31. The sudden inflow of velocity at inlet results in pressure fluctuations since the PDFs are initialized with the equilibrium

function and it is reduced by ramping the inflow velocity from 0 to $v_m$ with a smooth function from 0 s to 0.025 s.

For this investigation, the physical quantities like pressure (P) and velocity $\boldsymbol{v}$ are measured at certain areas of the simulation domain. To study the transient behavior, the physical quantities are averaged over an area and measured over the time. The tracking planes, lines and points at which the physical quantities are measured are illustrated in Figure 7.4 on the woven spacer with $\beta = 90°$. For all spacer configurations, the pressure drop $\Delta P$ is calculated from difference between the average pressure at two planes (yellow) which are 0.3 cm apart. The planes are 0.2 cm away from inlet and outlet to avoid an influence of boundaries on the pressure drop. The pressure gradient $\nabla P$ is obtained by dividing the pressure drop $\Delta P$ by the distance of 0.3 cm. For unsteady flows, the pressure drop is averaged over the last 50 values which are measured at every 100 iterations. In addition to the average pressure on the planes, the velocity profiles are measured over the height and the width of the channel represented by the green and the red line respectively in Figure 7.4a. The origin of the line over the height and the width are $[\frac{l_{ch}}{2} + d_f, 0, \frac{w_{ch}}{2}]$ and $[\frac{l_{ch}}{2} + d_f, \frac{h_{ch}}{2}, 0]$ respectively. Both origins are shifted by diameter of the filament along the length to avoid the filament. Furthermore, the physical quantities are measured over time at every 100 iterations at a single point (pink) located at $[3 \cdot l_m, \frac{h_{ch}}{2}, \frac{w_{ch}}{2}]$.




(a) Side view (b) Top view

**Figure 7.4** Tracking points, lines and planes in woven spacer $\beta = 90°$

## 7.1.2 Pure hydrodynamic flow

### 7.1.2.1 Convergence

The mesh convergence analysis was performed to determine the minimum resolution required to resolve the spacer geometry with sufficient numerical accuracy. The resolution with the least numerical error compared to the previous resolution is considered for the later investigations. The woven spacer with $\beta = 90°$ is used for this analysis. In LBM, to ensure the

stability of the simulation, the lattice velocity must be below 0.1 due to the incompressible limit and the relaxation parameter $\lambda_\nu$ must be between 0.5 and 2.0. Therefore, for fixed lattice velocity or relaxation parameter, the mesh resolution must be chosen properly to satisfy these stability conditions. In this study, the mesh resolution is varied by varying the number of elements in the height (nHeight) from 16 to 128 and the mean inflow velocity $\bar{v}_{in}$ is 0.2 $\mathrm{m\,s^{-1}}$. The element size ($\delta x$), the number of fluid elements (nFluids), the number of fluid elements with boundary neighbor (nBnds) and the number of fluid elements with boundary neighbor with qValues or elements next to the spacer geometry (nQVals) for various nHeight are listed in Table 7.2.

| nHeight | $\delta x$ [$\mu$m] | nFluids | nBnds | nQvals |
|---|---|---|---|---|
| 16 | 25 | 299 692 | 89 518 | 41 130 |
| 32 | 12.5 | 2 403 907 | 370 305 | 161 677 |
| 64 | 6.25 | 19 262 832 | 1 506 304 | 643 799 |
| 128 | 3.125 | 154 430 305 | 6 073 031 | 2 566 855 |

**Table 7.2** Number of elements for different mesh resolution

The simulations for the mesh up to $\delta x_{64}$ were performed on 10 nodes of the HorUS cluster in the University of Siegen while the mesh with $\delta x_{128}$ was performed on 10 nodes in the Hazel Hen in HLRS, Stuttgart. The pressure drop between the planes for different resolutions is listed in Table 7.3 along with number of iterations till steady state and the required wall clock time. Figure 7.5 shows the pressure drop between the planes over

| nHeight | dt [$\mu$s] | $\lambda_\nu$ | nIteration | Wall clock | $\Delta P$ [$1 \times 10^2$ Pa] |
|---|---|---|---|---|---|
| 16 | 2.5 | 1.95 | 40 280 | 22 min 53 s | 8.18 |
| 32 | 1.25 | 1.90 | 45 910 | 30 min 7 s | 6.32 |
| 64 | 0.625 | 1.81 | 91 150 | 2 h 51 min | 5.75 |
| 128 | 0.3125 | 1.66 | 193 790 | 9 h 26 min | 5.59 |

**Table 7.3** Pressure drop, number of iterations till steady state and run time for different mesh resolution

the time. Except $\delta x_{16}$, all other resolutions reached steady state before 0.1 s. The subplot inside the figure shows small fluctuations at the early stage due to the equilibrium initial condition, and the ramping of the inflow velocity can be observed in the evolution of the pressure drop up to 0.025 s. For the coarsest resolution $\delta x_{16}$, the pressure drop is very high due to numerical inaccuracy. It can be seen from the figure that the pressure drop converges to unique solution with an increase in resolution. From this analysis, a resolution $\delta x_{128}$ seems to be a reliable resolution but this

resolution requires high computational cost due to large mesh. Therefore, to reduce the computational cost, a resolution $\delta x_{64}$ was chosen because the relative difference between the resolution $\delta x_{64}$ and $\delta x_{128}$ is less than 3%.

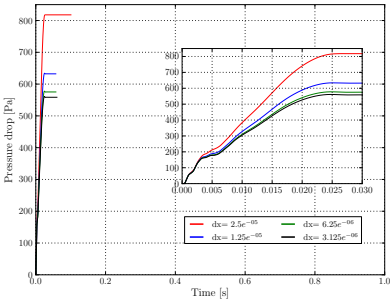
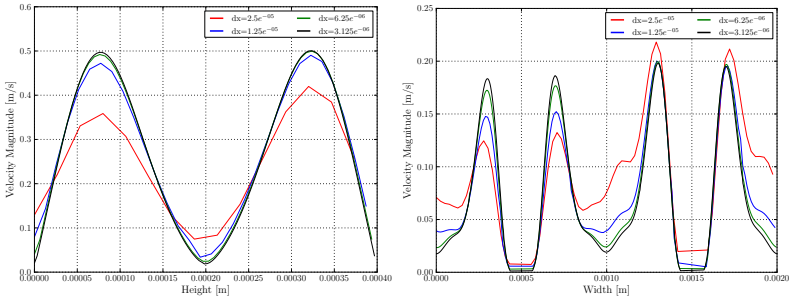



**Figure 7.5** Pressure drop over the time for different resolution

The velocity magnitude profile along the height and the width of the channel for different resolutions is shown in Figure 7.6. It can be seen in the figure that the velocity profiles converges with increasing resolution (decreasing element size). In general, LBM computes only cell center values but to compare different resolution, exact point values along the line are evaluated using linear interpolation. Two maxima along the height of the channel are due to the spacer filament in the middle resulting in flow going around the filament. The magnitude of the maxima are not equal because of the woven structure of the spacer. In the velocity profile along the width of the channel, the velocity is close to zero in two areas because of the spacer filaments running along the length. The tracking line along the width does not intersect with the spacer filament but runs slightly above and below the spacer filaments. Once again due to the woven structure of the spacer the magnitude of the first two maxima and the last two maxima are different while the magnitude of the two maxima next to the spacer filament are symmetric. The flow recirculation occurs in the exact middle of the channel. From both these profiles, it is evident that the resolution $\delta x_{64}$ is close to $\delta x_{128}$ and it is sufficient for the spacer flow simulations.

**Experimental validation** The numerical simulation is validated with a laboratory experiment. The experiment was performed for different number of flow channels with a woven spacer $\beta = 90°$ and the pressure drop is measured for various inflow velocities. The aim of this experiment is to




(a) Along the height (b) Along the width

**Figure 7.6** Velocity magnitude profile along the height and width of the channel

measure the pressure drop and compare with simulation results, so the drinking water is used with no external electrical field. Figure 7.7 shows the pressure drop over inflow velocities for *Musubi* simulations, experiments and the pressure drop prediction model. For inflow velocity below 0.1 $\mathrm{m\,s^{-1}}$, the pressure drop measured from simulation is much lower than the experiment and with increase in velocity, the simulation results match with experiment. Figure 7.7 also shows that the effect of the number of flow channels in the pressure drop measurement is small especially for velocities $< 0.1\ \mathrm{m\,s^{-1}}$. The discrepancy between simulation and experiment for small velocities is due to the pressure drop in the distribution channels because the pressure drop measured in experiment consists of the pressure drop in spacer-filled flow channels and the distribution system i.e. $\Delta P^{tot} = \Delta P^{ch} + \Delta P^{dis}$. However, in the numerical simulation only the spacer flow channel is considered. Therefore, the contribution from distribution channels must be subtracted from the total pressure drop to obtain the pressure drop in the spacer channel. Since it is not possible to measure the pressure drop in the distribution channel in experiment, its contribution is neglected in the developed prediction model. The semi-empiric prediction model for the pressure drop is based on the most commonly used model structure [98] for the incompressible flow through an arbitrary geometry and it is given as

$$\Delta P = \zeta \frac{l_g}{d_g} \frac{\rho v_{mean,in}^2}{2}. \tag{7.8}$$

Here, $\zeta$ is the friction coefficient, $l_g$ is the characteristic length and $d_g$ is the characteristic diameter of the geometry. For this validation, the characteristic length is chosen to be the laboratory spacer length ($l_g = l_{sp}$)

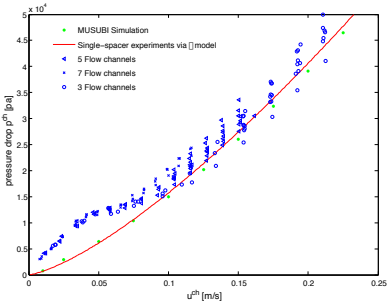



**Figure 7.7** Comparison of pressure drop over an $l_{sp} = 0.2$ m spacer channel by the numerical simulation with experiments and prediction model Eq. 7.9.

and the characteristic diameter is chosen be the height of the channel $d_g = h_{ch}$. As mentioned before, the total pressure drop measured consists of the pressure drop in spacer-filled flow channels and the distribution channels i.e. $\Delta P^{tot} = \Delta P^{ch} + \Delta P^{dis}$. Thus, both these contributions are accounted for in the total pressure drop by introducing friction coefficients $\zeta^{sp}$ and $\zeta^{dis}$ for spacer and distribution channels. With this, Eq. 7.8 can be written as

$$\Delta P^{tot} = \left(\zeta^{dist} + \zeta^{sp}\frac{l_{sp}}{h_{ch}}\right)\frac{\rho v_c^2 h}{2}. \tag{7.9}$$

The friction coefficients are obtained by the parameter estimating using least-square approximation on single spacer flow channel experiment data. Neglecting the friction coefficient of the distribution channels, the prediction $\zeta$-model matches pretty well the numerical simulation as shown in the figure. For more information on the calculation of friction coefficient refer to [39].

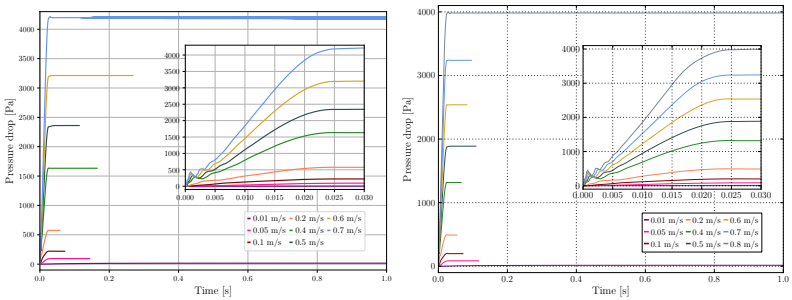
#### 7.1.2.2 Periodic width

The five different spacer configurations with periodic width are investigated by varying the inflow mean velocity from 0.01 $\mathrm{m\,s^{-1}}$ to 0.8 $\mathrm{m\,s^{-1}}$. As mentioned earlier, the simulations are run up to 1 s unless the steady state is reached. The evolution of the pressure drop over time between the planes which are 0.3 cm apart from each other for various $\bar{v}_{in}$ and for woven and nonwoven configurations are shown in Figure 7.8 and Figure 7.9 respectively. The behavior of the flow for the different configurations can be identified from this figure. For all spacer configurations, numerical fluctuations are observed due to initial conditions, which results in reflection of pressure

waves from the outlet boundary. As it can be seen in the subplots of Figure 7.8 and Figure 7.9a, these initial numerical fluctuations gets washed away even before the ramping time of 0.025 s. Also note that these initial fluctuations increase with increase in velocity.

In Figure 7.8 and Figure 7.9, the lines stop at different times because it takes a different number of time steps until it reaches steady state. See Section 5.4.4, for more information on how the steady state check is performed by the solver. Here, the steady state is checked by measuring the pressure and velocity magnitude variables at every 10th time step. The sliding window is 100 resulting in 100 values from 1000 time steps. The threshold is $1 \times 10^{-8}$. Thus, the time average is computed for 100 values and its difference against the current value is computed for both the pressure and velocity magnitude variables. The steady state is assumed to have reached only when the computed difference of both the variables are less then the threshold of $1 \times 10^{-8}$. Since the physical time is directly proportional to the physical time step $\delta t$ and the physical time step $\delta t$ is inversely proportional to the physical mean inflow velocity $\bar{v}_{in}$ (Eq. 3.89). The physical time required to reach the steady state for the fixed number of 1000 time steps decreases with increase in the mean inflow velocity $\bar{v}_{in}$. This behavior can be clearly seen in Figure 7.9a for the nonwoven spacer $\beta = 30°$ for entire velocity range. However, for other configurations except the nonwoven spacer $\beta = 60°$, this behavior is observed for $\bar{v}_{in}$ only up to 0.2 $\mathrm{m\,s^{-1}}$. For higher velocities, the physical time to reach steady state differs from each other due to initial numerical fluctuations. In addition to the evolution of pressure drop, the pressure drop for various inflow velocities for five spacer configurations are listed in Table 7.4. For steady state flows, the pressure drop is taken at the last time step and for unsteady flows, the pressure drop is averaged over last 5000 iterations.

For the woven spacer $\beta = 90°$, the flow reached steady state for $\bar{v}_{in}$ up to 0.6 $\mathrm{m\,s^{-1}}$ and from 0.7 $\mathrm{m\,s^{-1}}$, the flow becomes unsteady with periodic fluctuations. The simulation for 0.8 $\mathrm{m\,s^{-1}}$ was crashed due to insufficient resolution. Just rotating the woven spacer $\beta = 90°$ by $\beta = 45°$ along the flow direction results in a steady state solution for all velocities. Furthermore, the ratio of the pressure drop of the woven spacer $\beta = 45°$ to $\beta = 90°$ decreases with increase in velocity. In other words, the pressure drop of the woven spacer $\beta = 45°$ is 0.8-0.9 times less than woven spacer $\beta = 90°$ for the same $\bar{v}_{in}$. Additionally, the pressure drop of woven spacer $\beta = 45°$ seems to increase linearly with increase in velocity. The pressure drop of the nonwoven spacer $\beta = 30°$ is much less than the woven spacers and reaches steady state for all velocities. In comparison to the woven spacers, the pressure drop of the nonwoven spacer $\beta = 30°$ is



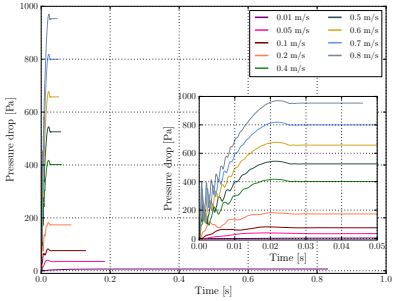

(a) Woven spacer $\beta = 90°$ (b) Woven spacer $\beta = 45°$

**Figure 7.8** Pressure drop over time for various inflow mean velocities of woven spacer configurations

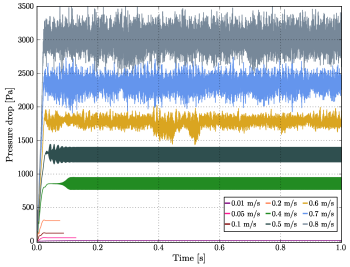
| $\bar{v}_{in}$ | woven | | nonwoven | | |
|---|---|---|---|---|---|
| $[\mathrm{m\,s^{-1}}]$ | $\beta = 90°$ | $\beta = 45°$ | $\beta = 30°$ | $\beta = 45°$ | $\beta = 60°$ |
| 0.01 | 0.1659 | 0.1527 | 0.065 | 0.0883 | 0.1254 |
| 0.05 | 0.9469 | 0.8587 | 0.35 | 0.5025 | 0.7914 |
| 0.1 | 2.20 | 1.98 | 0.76 | 1.20 | 2.13 |
| 0.2 | 5.75 | 4.90 | 1.73 | 3.07 | $\overline{6.38}$ |
| 0.4 | 16.33 | 13.17 | 4.02 | $\overline{8.68}$ | $\overline{22.68}$ |
| 0.5 | 23.6 | 18.87 | 5.25 | $\overline{12.7}$ | $\overline{35.5}$ |
| 0.6 | 32.12 | 25.38 | 6.57 | $\overline{18.33}$ | $\overline{48.97}$ |
| 0.7 | $\overline{41.96}$ | 32.39 | 8.00 | $\overline{23.75}$ | NA |
| 0.8 | NA | 39.83 | 9.53 | $\overline{30.3}$ | NA |

**Table 7.4** Pressure drop between the planes ($\Delta P$) in $1 \times 10^2$ Pa for 5 different spacer configuration over various inflow mean velocity $\bar{v}_{in}$. A bar over the value represents the unsteady flows. NA represents the simulations which crashed due to insufficient resolution.

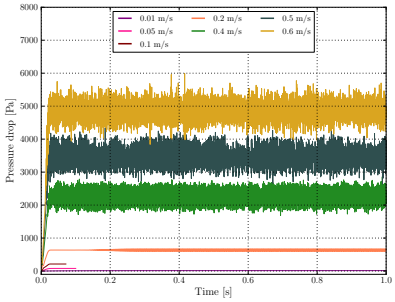
roughly 0.4 times smaller for velocity 0.01 $\mathrm{m\,s^{-1}}$ and 0.2 times smaller for velocity 0.7 $\mathrm{m\,s^{-1}}$. For other two nonwoven spacers, with the increase in hydrodynamic angle, the pressure drop increases and also the flow becomes unsteady and fluctuations increase with increase in velocity. The nice transition from steady flow to unsteady periodic laminar flow to unsteady flow with large fluctuations can be observed for both nonwoven spacers $\beta = 45°$ and $\beta = 60°$. For the nonwoven spacer $\beta = 45°$, the flow changes from steady state flow to unsteady periodic flow for velocity 0.4 $\mathrm{m\,s^{-1}}$. From velocity 0.6 $\mathrm{m\,s^{-1}}$ the flow fluctuations becomes chaotic and behaves more like a turbulent flow. With increasing the hydrodynamic angle for the




(a) nonwoven spacer $\beta = 30°$




(b) nonwoven spacer $\beta = 45°$

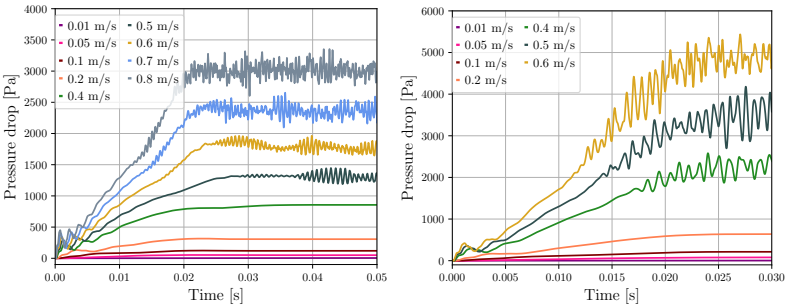



(c) nonwoven spacer $\beta = 60°$

**Figure 7.9** Pressure drop over time for various inflow mean velocities of nonwoven spacer configurations

nonwoven spacer to $\beta = 60°$, the flow transition from steady state flow to unsteady periodic flow occurs at earlier velocity of 0.2 $\mathrm{m\,s^{-1}}$ and the high fluctuations are observed already at velocity 0.4 $\mathrm{m\,s^{-1}}$. Due to increase in high fluctuations with velocity, the simulations crashed for velocity $\geq 0.7$ $\mathrm{m\,s^{-1}}$ due to insufficient resolution. Thus, it can be concluded that the flow becomes more turbulent with increase in hydrodynamic angle.

A closer look at the evolution of the pressure drop (see Figure 7.10) shows that for the velocities with high fluctuations, the fluctuation starts to appear even before the end of ramping time. To check whether these fluctuations for high velocity are physical or numerical, the high inflow velocities are ran with increased resolution of $\delta x_{128}$. The fluctuations were almost the same which shows that they are physical.



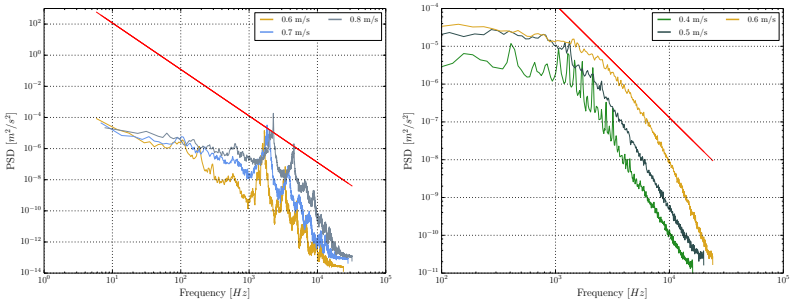

(a) nonwoven spacer $\beta = 45°$ (b) nonwoven spacer $\beta = 60°$

**Figure 7.10** Evolution of pressure drop during the velocity ramping for the nonwoven spacer $\beta = 45°$ (left) and $\beta = 90°$ (right).

The turbulent flow behavior observed for nonwoven spacers $\beta = 45°$ and $\beta = 60°$ are investigated further using Kolmogorov $-5/3$ scale energy decay. For the high fluctuation flows, the power spectrum density (PSD) over the frequency was computed from the velocity measured at the center the simulation domain $[3 \cdot l_m, \frac{h_{ch}}{2}, \frac{w_{ch}}{2}]$. The PSD over frequency for both those nonwoven spacers with high fluctuations are shown in Figure 7.11. The left figure shows the PSD for the nonwoven spacer $\beta = 45°$ for velocities from 0.6 $\mathrm{m\,s^{-1}}$ to 0.8 $\mathrm{m\,s^{-1}}$ and the right figure shows the PSD for the nonwoven spacer $\beta = 60°$ for velocities 0.4 $\mathrm{m\,s^{-1}}$ to 0.6 $\mathrm{m\,s^{-1}}$. A red line in the figure represents the Kolmogorov $-5/3$ energy decay and for the flow to remain turbulent, the PSD of the flow should be parallel to the slope. In general, the turbulent flow is unsteady and chaotic with unpredictable random motion of the vortices or eddies in different scales. The kinetic energy of the turbulent flows enters through the large scale eddies which are created by the flow separation influenced by the geometry (boundaries). These large eddies are unstable and break up, transferring their energy into small eddies. These smaller eddies undergo a similar break up process until the eddies are so small and get dissipated due to viscosity of the flow. According to Kolmogorov theory, the net energy transferred from energy containing large eddies is in equilibrium with the energy in smaller eddies where it is dissipated. The frequency range in which this energy transfer happens is called the inertial subrange and the slope of the energy spectrum at this range remains constant. Kolmogorov showed thats this slope is $-5/3$ based on dimensional arguments [74].

From Figure 7.11, it can be seen that the inertial subrange exists for

nonwoven spacer $\beta = 45°$ between frequency $10^3$ and $10^4$. This frequency range is self similar for all three velocities and only the magnitude of energy decreases with decrease in velocity. On the other hand, with increased hydrodynamic angle to $\beta = 60°$ for woven spacer, the inertial subrange is different for each velocity but they all dissipate much faster with a sharp decrease in slope of PSD. For velocity 0.4 $\mathrm{m\,s^{-1}}$ and 0.5 $\mathrm{m\,s^{-1}}$, the inertial subrange exists between frequency $10^3$ Hz and $3 \cdot 10^3$ Hz and for velocity 0.6 $\mathrm{m\,s^{-1}}$, the inertial subrange is between frequency $3 \cdot 10^3$ Hz and $5 \cdot 10^3$ Hz. After the inertial subrange, the slope of PSD gets steeper due to the faster decay of energy. Thus, the existence of an inertial subrange in the small frequency range suggests that the turbulent energy in the spacer-filled flow channels does not last long and gets dissipated at much faster rate.




(a) nonwoven spacer $\beta = 45°$ (b) nonwoven spacer $\beta = 60°$

**Figure 7.11** Power spectrum density over frequency for the inflow mean velocities 0.6 $\mathrm{m\,s^{-1}}$ to 0.8 $\mathrm{m\,s^{-1}}$ for the nonwoven spacer $\beta = 45°$ (left) and $\beta = 90°$ (right).

The pressure drop in Table 7.4 is plotted against the inflow mean velocity in Figure 7.12. Only the pressure drop of the nonwoven spacer $\beta = 30°$ shows linear increase with velocity while others increase nonlinearly with velocity. It can be clearly seen that with increase in hydrodynamic angle $\beta$, the pressure drop increases drastically. For example, the pressure drop of nonwoven spacer $\beta = 60°$ is roughly 7.5 times larger than nonwoven spacer $\beta = 30°$ for the velocity of 0.6 $\mathrm{m\,s^{-1}}$. For the same hydrodynamic angle of $\beta = 45°$, the ratio of the pressure drop of woven spacer to nonwoven spacer is roughly 1.3-1.7 times and this ratio increases with increase in velocity. For low velocities, the pressure drop of nonwoven spacer $\beta = 60°$ is slightly lower than the woven spacer $\beta = 90°$ and with increase in velocity, the

pressure drop of nonwoven spacer $\beta = 60^\circ$ becomes higher than the woven spacer $\beta = 90^\circ$ i.e. roughly 1.5 times higher for velocity 0.6 $\mathrm{m\,s^{-1}}$. The pressure drop of nonwoven spacer $\beta = 45^\circ$ is roughly 0.5-0.7 times smaller than the woven spacers.

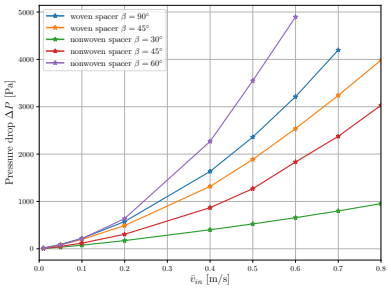



**Figure 7.12** Pressure drop $\Delta P$ over inflow mean velocity for different spacer configurations

In ED application, the pressure drop across the channel is directly proportional to the pump energy which in turn relates to the operational cost. Therefore, the spacer design with the least pressure drop is preferred. On the other hand, the spacer with low pressure drop increases low flow zones which are the main cause for scaling and fouling effect in spacer-filled flow channels. Thus, the optimal spacer design should have low pressure drop and less low flow zones. So, to identify the optimal spacer design, in addition to the pressure drop study, the flow distribution in the channel must also be studied in detail. To study the flow distribution, the streamlines of velocity magnitude for different spacer configurations are compared for two velocities: 0.01 $\mathrm{m\,s^{-1}}$ (low) and 0.5 $\mathrm{m\,s^{-1}}$ (high). In addition to streamlines, the velocity magnitude profile in the $xz-$plane at $y = h_{ch}/2$ and $y = 3h_{ch}/4$ are also compared. It is worth mentioning that in addition to aforementioned conditions for the optimal spacer design, the optimal spacer is also desired to maximize the limiting current density. But it is not considered in this work and it will be considered in the future work.

Before discussing the flow distribution in the spacer-filled flow channels, the Re of different runs are discussed. The Reynolds number at the beginning of the simulation ($Re_{sp.B}$) is calculated from the inlet mean velocity $\bar{v}_{in}$ and the Reynolds number at the end of the simulation ($Re_{sp,E}$) is calculated from the maximum velocity magnitude $v_{max,E}$ in the simulation domain. The $v_{max,E}$ is measured at the last time for steady state flows and

averaged over last 5000 time steps for unsteady flows. Like the pressure drop, the Reynolds number at the beginning $Re_{sp,B}$ and at end $Re_{sp,E}$ of the simulation are listed in Table 7.5. The ratio of $Re_{sp,E}/Re_{sp,B}$ is plotted against $\bar{v}_{in}$ in Figure 7.13. Between those two Re, the characteristic height and the kinematic viscosity are kept constant and only the characteristic velocity differs so the ratio of $Re_{sp,E}/Re_{sp,B}$ is equal to ratio of $v_{max,E}/\bar{v}_{in}$. Thus, the figure can also be interpreted as increase in velocity magnitude with respect to the inflow mean velocity. The ratio is always larger than 1 which shows that the flow is accelerated through the channel due to the presence of spacer geometry. From these results, it can be concluded that for nonwoven spacer with $\beta \geq 45°$, the flow changes from steady to unsteady for $Re_{sp,E} > 200$ and the flow becomes turbulent for $Re_{sp,E} > 700$.

| $\bar{v}_{in}$ | $Re_{sp,B}$ | $Re_{sp,E}$ | | | | |
|---|---|---|---|---|---|---|
| | | woven | | nonwoven | | |
| $[\mathrm{m\,s^{-1}}]$ | | $\beta = 90°$ | $\beta = 45°$ | $\beta = 30°$ | $\beta = 45°$ | $\beta = 60°$ |
| 0.01 | 3.7 | 17.34 | 13.95 | 7.07 | 9.58 | 14.09 |
| 0.05 | 18.48 | 79.76 | 64.32 | 40.2 | 51.05 | 64.55 |
| 0.1 | 36.97 | 151.54 | 116.48 | 81.2 | 100.45 | 138.95 |
| 0.2 | 73.94 | 294.31 | 212.56 | 174.14 | 194.85 | 218.96 |
| 0.4 | 147.87 | 594.09 | 414.12 | 345.45 | 410.82 | 707.48 |
| 0.5 | 184.84 | 744.11 | 539.22 | 440.82 | 545.46 | 966.4 |
| 0.6 | 221.81 | 894.72 | 666.25 | 537.04 | 720.1 | 1198.46 |
| 0.7 | 258.78 | 1045.59 | 801.19 | 633.8 | 865.31 | NA |
| 0.8 | 295.95 | NA | 937.41 | 730.94 | 1008.89 | NA |

**Table 7.5** Re computed using $\bar{v}_{in}$ ($Re_{sp,B}$) and Re computed with $v_{max,E}$ in the simulation domain at the of end of simulation ($Re_{sp,E}$) for different spacer configuration over various inflow mean velocity.

Figure 7.14 shows streamlines of the fluid flow colored with velocity magnitude for different spacer configurations for $\bar{v}_{in} = 0.01\ \mathrm{m\,s^{-1}}$. A streamline represents a path followed by a fluid particle as it moves with the flow. For all spacer configurations, the flow reaches its maximum velocity between the filament and the membrane due to the reduced cross-sectional area. Also, for this inflow velocity, the flow is steady and laminar for all configurations. In woven spacers, the fluid goes around the filaments along the flow direction due to their sinusoidal arrangement of filaments. Even with a rotated angle of $\beta = 45°$, the fluid flows almost along the main flow direction with a zigzag pattern. On the other hand, the fluid in the nonwoven spacers changes its direction at every filament intersection

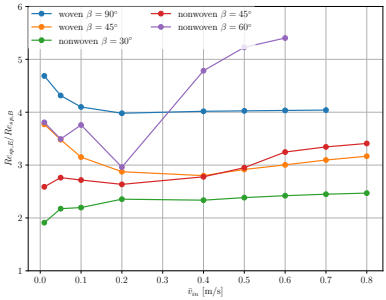
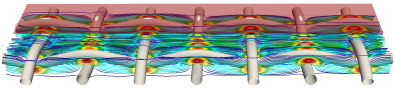
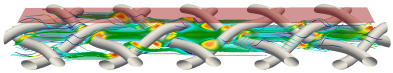


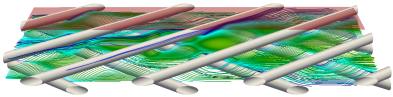

**Figure 7.13** Ratio of Reynolds number at the end to the beginning of the simulation over inflow mean velocity for different spacer configurations



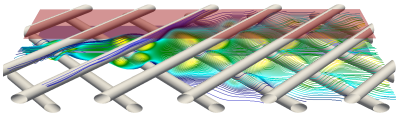
(a) Woven spacer $\beta = 90°$



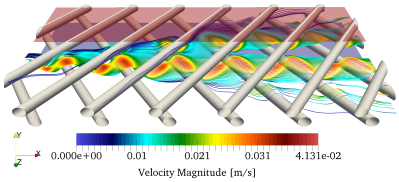
(b) Woven spacer $\beta = 45°$



(c) nonwoven spacer $\beta = 30°$



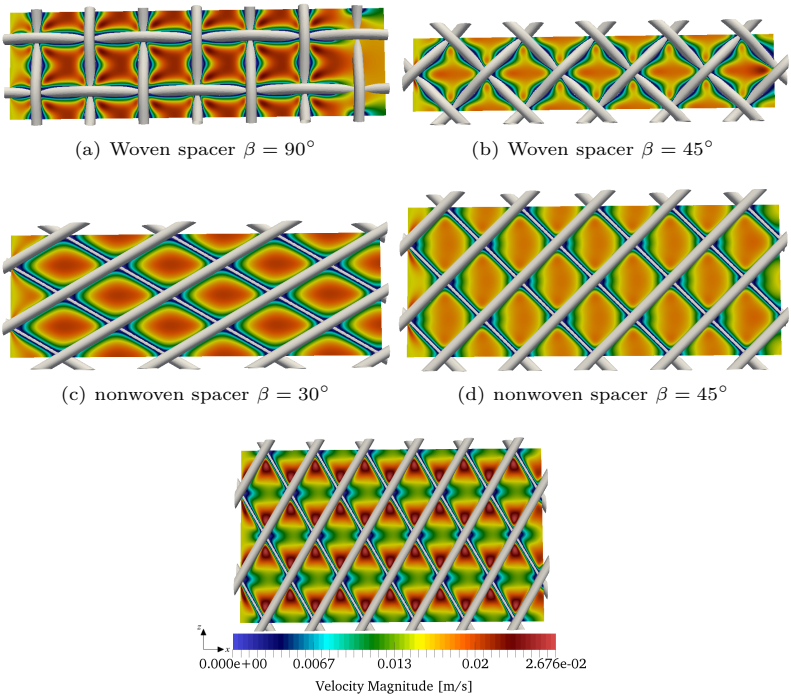
(d) nonwoven spacer $\beta = 45°$




(e) nonwoven spacer $\beta = 60°$

**Figure 7.14** Stream lines of velocity magnitude for inflow mean velocity 0.01 $\mathrm{m\,s^{-1}}$ of different spacer configurations

and flows along the orientation of the filament. Thus, with increasing hydrodynamic angle, the fluid stays longer in the flow channel and it takes a longer path until it reaches the outlet. Figure 7.14 also shows that near the filament, the flow velocity is very low since spacers act like walls. In nonwoven spacers the fluid with low velocities flow in parallel to the filaments. In comparison to other spacers, the flow in the woven spacer




(a) Woven spacer $\beta = 90°$

(b) Woven spacer $\beta = 45°$

(c) nonwoven spacer $\beta = 30°$

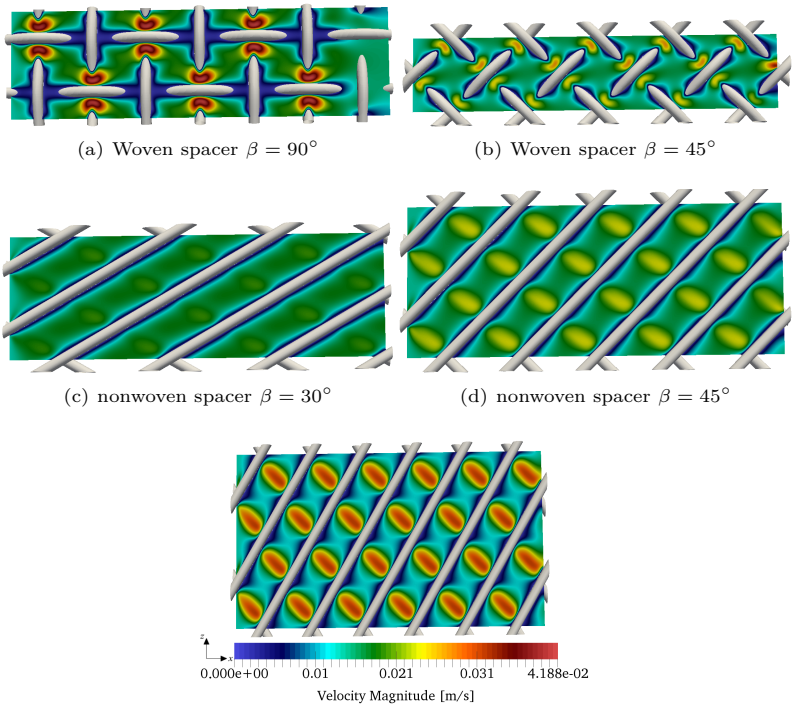
(d) nonwoven spacer $\beta = 45°$

(e) nonwoven spacer $\beta = 60°$

**Figure 7.15** Velocity magnitude profile for inflow mean velocity 0.01 $\mathrm{m\,s^{-1}}$ of different spacer configurations at the middle slice $y = h_{ch}/2$ along the length of the spacer

$\beta = 45°$ is more uniform with less low flow areas. It can be clearly seen that with increasing hydrodynamic angle, the velocity magnitude in the domain increases, notably between the filament and the membrane closer to the filaments intersection. Near this intersection, the distance between the consecutive zigzag filaments (one up and one down) decreases with increased angle $\alpha$ which causes change in fluid direction which in turn increases the flow velocity. It shows that the fluid velocity in the domain increases when the distance between the filaments $l_m$ decreases. As the fluid moves towards the center of the intersections, the velocity decreases which can be more clearly seen in Figure 7.15e.

The flow distribution in the $xz$-plane at $y = h_{ch}/2$ for different spacer




(a) Woven spacer $\beta = 90°$

(b) Woven spacer $\beta = 45°$

(c) nonwoven spacer $\beta = 30°$

(d) nonwoven spacer $\beta = 45°$

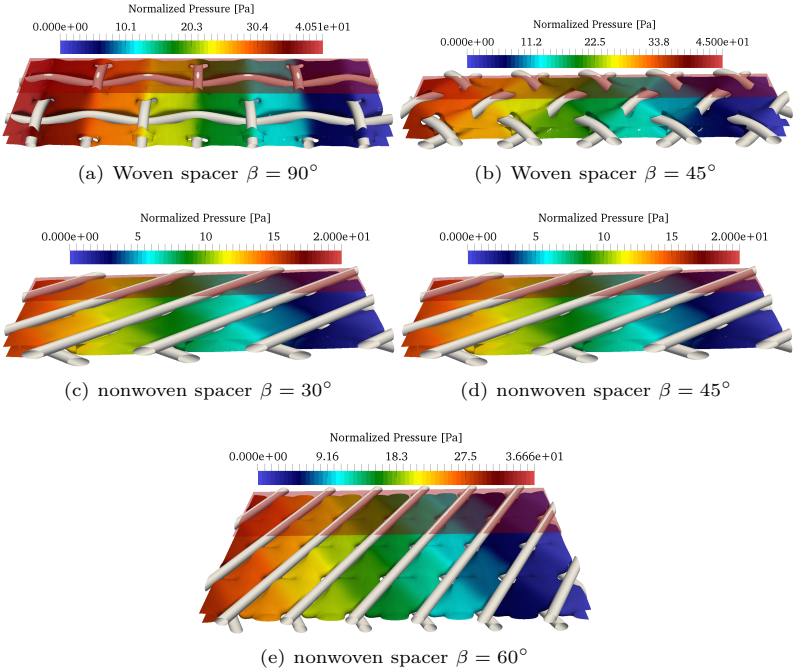
(e) nonwoven spacer $\beta = 60°$

**Figure 7.16** Velocity magnitude profile for inflow mean velocity 0.01 $\mathrm{m\,s^{-1}}$ of different spacer configurations at the upper slice $y = 2h_{ch}/3$ along the length of the spacer

configuration for $\bar{v}_{in} = 0.01\ \mathrm{m\,s^{-1}}$ is shown in Figure 7.15. In all configurations except the nonwoven spacer $\beta = 60°$ the flow velocity is high in the free flow area which is between the filament intersections. In the nonwoven spacer $\beta = 60°$, the flow velocity is high near the filament intersection as observed in the streamlines figure and in the middle of the free flow area the velocity is close to inflow velocity $\bar{v}_{in} = 0.01\ \mathrm{m\,s^{-1}}$.

Shifting the $xz$- plane to $y = 2h_{ch}/3$ shows different distribution of the flow for the spacer configurations (see Figure 7.16). In the woven spacer $\beta = 90°$, the flow velocity is very low above the filament running along the flow direction and very high above the filament running perpendicular

to the flow direction. The low flow velocity at the filaments intersection looks like a pool where ions could accumulate. In the rotated woven spacer $\beta = 45°$, the pool disappears and the flow is more evenly distributed. In this slice, the effect of the outlet BC in the simulation domain can been seen in the woven spacer $\beta = 90°$ up to the second last filament running along the z-direction. In nonwoven spacers, the high velocity between the bottom filament and top membrane are clearly visible and the magnitude of this velocity increases with increase in angle $\beta$. Even though the nonwoven spacer $\beta = 60°$ has the highest velocity, it also has larger low velocity zones near the filaments. The nonwoven spacer $\beta = 45°$ has more even flow distribution with relatively less low flow zones.




(a) Woven spacer $\beta = 90°$

(b) Woven spacer $\beta = 45°$

(c) nonwoven spacer $\beta = 30°$

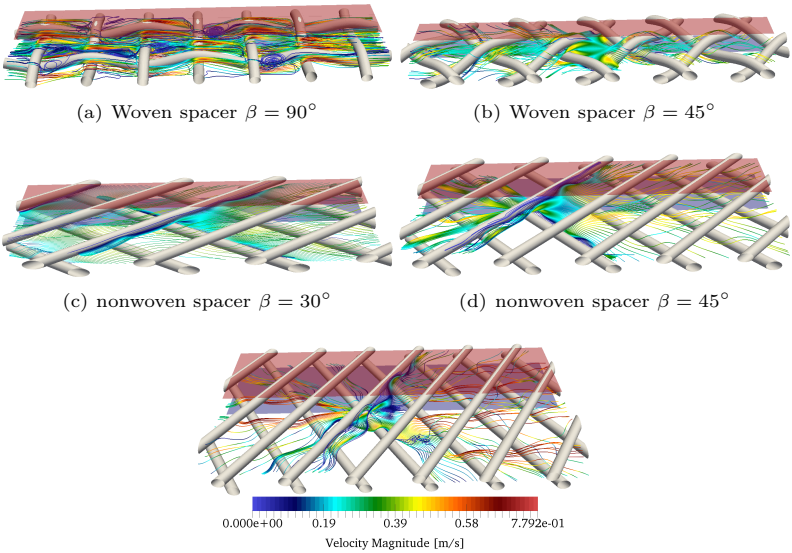
(d) nonwoven spacer $\beta = 45°$

(e) nonwoven spacer $\beta = 60°$

**Figure 7.17** Contours of velocity magnitude 0.01 $\mathrm{m\,s^{-1}}$ colored with normalized pressure for different spacer configurations

The pressure drop along the spacer-filled flow channel for different spacer configurations is shown in Figure 7.17. The surface contour represents the velocity magnitude of 0.01 $\mathrm{m\,s^{-1}}$ and the colors represents the normalized

pressure, i.e. $P - P_{atm}$. Figure 7.17 shows that the pressure decreases almost linearly along the length similar to the straight channel.




(a) Woven spacer $\beta = 90°$

(b) Woven spacer $\beta = 45°$

(c) nonwoven spacer $\beta = 30°$

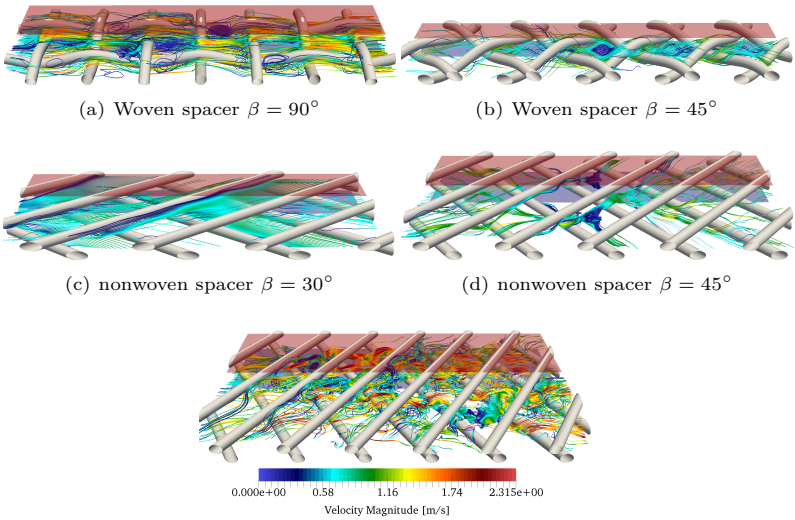
(d) nonwoven spacer $\beta = 45°$

(e) nonwoven spacer $\beta = 60°$

**Figure 7.18** Streamlines of velocity magnitude for inflow mean velocity 0.2 $\mathrm{m\,s^{-1}}$ of different spacer configurations

Increasing the inflow velocity to 0.2 $\mathrm{m\,s^{-1}}$ introduces vortices behind the filaments in the woven spacer $\beta = 90°$ and the nonwoven spacer $\beta = 60°$ (see Figure 7.18). In the woven spacer $\beta = 90°$, the vortices appear above the filament running along the flow direction. It's the same area in which very low velocity was observed for $\bar{v}_{in} = 0.01\ \mathrm{m\,s^{-1}}$. Likewise, the vortex in the nonwoven spacer $\beta = 60°$ appears in the center of the free flow area, where the low flow velocity was observed for $\bar{v}_{in} = 0.01\ \mathrm{m\,s^{-1}}$. In the other three spacer configurations, the twist in the flow direction can be seen at every filament intersection.

With further increase in inflow velocity to 0.5 $\mathrm{m\,s^{-1}}$ vortices are also introduced in the free flow area of the woven spacer $\beta = 45°$ and the nonwoven spacer $\beta = 45°$ as shown in Figure 7.19. Still the flow in the nonwoven spacer $\beta = 30°$ is laminar and uniform with no vortices. The size of vortices in the woven spacer $\beta = 90°$ increases but still the flow is




(a) Woven spacer $\beta = 90°$ (b) Woven spacer $\beta = 45°$

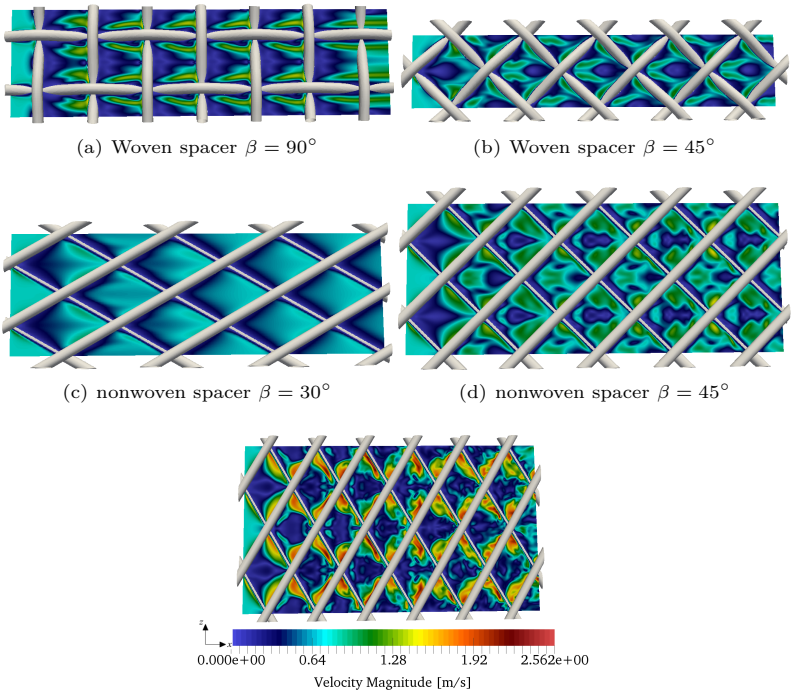
(c) nonwoven spacer $\beta = 30°$ (d) nonwoven spacer $\beta = 45°$

(e) nonwoven spacer $\beta = 60°$

**Figure 7.19** Streamlines of velocity magnitude for inflow mean velocity 0.5 $\mathrm{m\,s^{-1}}$ of different spacer configurations

laminar and reaches a steady state. On the other hand, the flow in the nonwoven spacer $\beta = 60°$ became chaotic with many small vortices and the velocity magnitude is more than 5 times larger than the inflow velocity 0.5 $\mathrm{m\,s^{-1}}$. This chaotic large fluctuations in the nonwoven spacer $\beta = 60°$ may not be desirable for effective operation of ED stack. However, it could be effective to clean the scaling in the spacer-filled flow channels and also effective in mixing.

In Figure 7.20 and Figure 7.21, the effect of inlet and outlet BC on the flow distribution can be seen up to the second filament and second last filament running perpendicular to the flow direction respectively. This was the main reason for measuring the pressure drop between the planes that are 0.3 cm apart and located at a distance of 0.2 cm away from inlet and outlet boundary. Except for the nonwoven spacer $\beta = 30°$, all other other configurations have very low velocity in the free flow area which is due to the presence of vortices. A chaotic flow distribution in the nonwoven spacer $\beta = 60°$ shows a symmetric distribution along the z-direction but a very different distribution at every filament intersection along the flow direction. However, all other spacer configurations show self




(a) Woven spacer $\beta = 90°$ (b) Woven spacer $\beta = 45°$

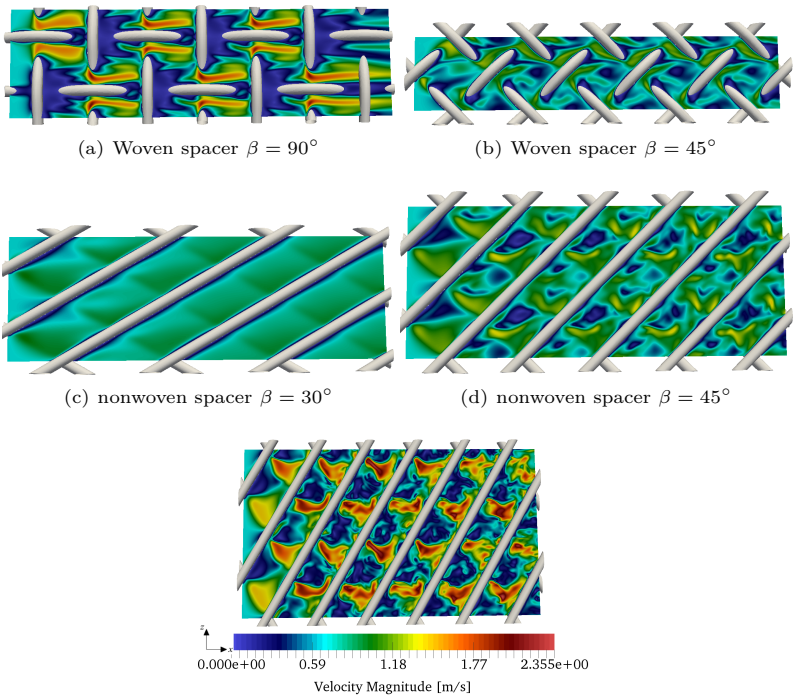
(c) nonwoven spacer $\beta = 30°$ (d) nonwoven spacer $\beta = 45°$

(e) nonwoven spacer $\beta = 60°$

**Figure 7.20** Velocity magnitude profile for inflow mean velocity 0.5 $\mathrm{m\,s^{-1}}$ of different spacer configurations at the middle slice $y = h_{ch}/2$ along the length of the spacer

similar flow distribution at every filament cross section and in the free flow area which are away from inlet and outlet boundary. These results show that the flow distribution is always periodic in z-direction and differ only in the main flow direction. Also, there are no differences in the main flow direction between each spacer element (which consists of two filaments in x- and y-direction) away from inlet and outlet boundaries. The difference in the flow direction is observed only for the nonwoven spacer for inflow velocitity 0.5 $\mathrm{m\,s^{-1}}$. Therefore, for inflow velocities up to 0.5 $\mathrm{m\,s^{-1}}$, the pressure drop measured in the middle of the spacer-filled flow channel can be extrapolated to the entire spacer length.

Here is the short summary of this investigation. The existence of the




(a) Woven spacer $\beta = 90°$

(b) Woven spacer $\beta = 45°$

(c) nonwoven spacer $\beta = 30°$

(d) nonwoven spacer $\beta = 45°$

(e) nonwoven spacer $\beta = 60°$

**Figure 7.21** Velocity magnitude profile for inflow mean velocity 0.5 $\mathrm{m\,s^{-1}}$ of different spacer configurations at the upper slice $y = 2h_{ch}/3$ along the length of the spacer

spacer reduces the area of the flow, thus the flow is accelerated. The acceleration should corresponds to the area reduction. As the different spacers correspond to different areas, the maximum flow velocity is different. The second phenomenon is the uniformity of the flow. The nonwoven spacer is expected to be more uniform than the woven spacer, as the flow does not need to flow around the filaments. Furthermore, the fluid in the nonwoven spacers changes its direction at every filament intersection and flows along the orientation of the filament where as in the woven spacers, the fluid flows almost along the main flow direction. And the third is the, hydrodynamic angle $\beta$ as it changes the distance between the filaments. With increasing hydrodynamic angle, the fluid stays longer in

the flow channel and it takes a longer path until it reaches the outlet. This phenomena should enhance the mass transport of ionic species through the membranes. For both woven and nonwoven spacers, the flow becomes more transient towards turbulence with an increase in hydrodynamic angle $\beta$ as concluded in [46]. Thus, the spacer with larger hydrodynamic angle can be used as a turbulence promoter, but is not suitable for an ED process for seawater desalination which requires a more uniform flow with little mixing to promote ions transfer through the membrane. The turbulence is often induced in the ED process to increase the limiting current density, to minimize boundary layer thickness and to wash accumulated ions in the low flow areas. However, there is a limit on the operating velocity, and the pressure drop along the spacer and through the stack, to prevent external leakages. For a more detailed analysis, refer to [75].

#### 7.1.2.3 Near sealed corner

In the previous investigation, the width of the channel was considered to be periodic assuming the simulation domain is far away from the sidewalls and corners. In reality, the spacers have sealed corners, which are used to seal the spacer sheet in the ED stack. The illustration of a square spacer sheet with sealed corners used by SIEMENS Water Technology in their ED prototype is shown in Figure 7.22. The sealed corners are prone to have low flow zones where the fluid velocity becomes low. This low flow zones are the main cause for scaling and fouling effects in the spacer channels because they lead to an accumulation of ions. Thus, it leads to failure of the system and the only solution to this problem is disassembling the entire stack and cleaning the spacer. Usually, this failure is noticed only when the performance of the system is reduced. From previous section, it is known that the flow distribution in the spacer-filled flow channel depends on the spacer design and the inflow velocity but it is unknown how much they affect the flow in the corners. Therefore, it is necessary to study the flow distribution of different spacer configurations near the sealed corner to determine the optimal spacer design.

The investigations are performed only on nonwoven spacers because they are used more often than the woven spacers. The nonwoven spacers are also cheaper than the woven spacers. Thus, nonwoven spacers with different hydrodynamic angle $\beta$ and number of filaments per inch density $nF/inch$ are investigated. The geometric parameters of such a nonwoven spacer are given in Figure 7.1. Three different spacer configurations are chosen:

1. $\beta = 30^{\circ}, nF/inch = 14$,

2. $\beta = 60°, nF/inch = 14$ and

3. $\beta = 45°, nF/inch = 18$.

Each of these configurations is simulated with two inflow velocities of 0.03 $\mathrm{m\,s^{-1}}$ and 0.05 $\mathrm{m\,s^{-1}}$. Thus, totally 6 simulations are performed for this investigation. The highlighted corner area in Figure 7.22 is of

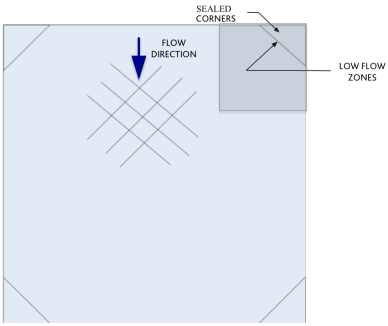



**Figure 7.22** Spacer sheet with sealed (blocked) corners. Simulations are performed on the highlighted part which covers low flow zone.

dimension $L \times W = 7.1 \times 7.1$ cm. The fluid mesh with spacer structure generated by *Seeder* and the setup of the boundary conditions used for the simulations are shown in Figure 7.23. The zoomed part shows the grid resolution of spacer filament. The spacer filament is approximated by 10 elements (lattice cells) i.e 20 elements in channel height resulting in a total problem size of around 520 million elements. Due to memory requirements to generate such big meshes, the pre- and post- processing node with 128 GB memory in the Hermit system is used. Each mesh generation consumed a machine memory of roughly 70 GB and a wall clock time of roughly 3 h. The disk space used by each mesh is roughly 22 GB.

Here, the BGK collision operator and $D3Q19$ layout are used. At inlet, the inflow mean velocity ($\overline{v}_{in}$) is imposed using velocity bounce back BC and at outlet, atmospheric pressure is prescribed using pressure extrapolation BC. The sealed corner walls and spacer structure are treated as no-slip bounce back BC. At the plane opposite to the blocked wall, free-slip boundary is imposed i.e flow velocity normal to this wall is set to zero and the tangential velocity is extrapolated. Each simulation was run on 2048 cores in the Hermit system, HLRS, Stuttgart, simulating up to 20 s in a wall clock time of $40hrs$. The simulations are run for long

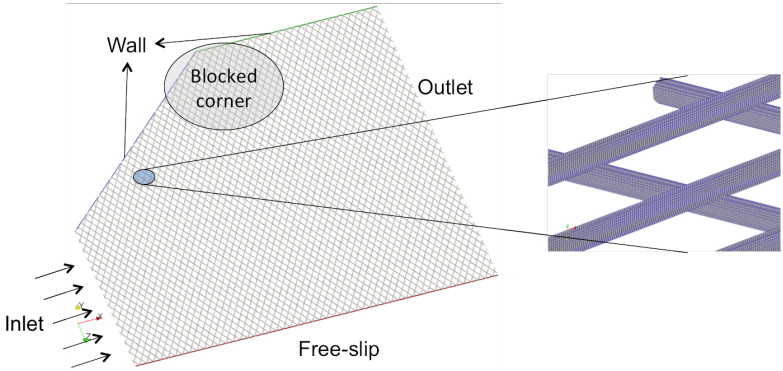
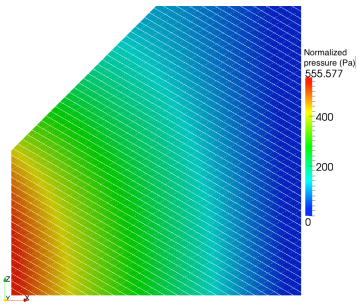



**Figure 7.23** Spacer generated by Seeder marked with boundary condition setup. Zoomed part shows the resolution of spacer filament used for simulations.

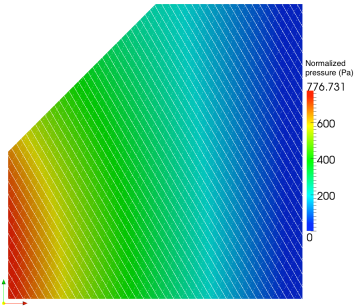
time to ensure that the flow reaches the steady state. All simulations reached steady state with in the simulation end time of 20 s except the configuration No. 2 with inflow velocity 0.03 $\mathrm{m\,s^{-1}}$, which took longer to reach steady state. So this simulation alone was ran up to 60 s.

Pressure distribution at steady state for different spacer configurations on the $xz$- plane at $y = h_{ch}/2$ for $\bar{v}_{in} = 0.03\ \mathrm{m\,s^{-1}}$ are shown in Figure 7.24. Due to the sealed corner, the pressure gradient exist in the z-direction i.e it decreases from slip wall to the corner wall. The slope of this gradient is greater for configuration No. 1 than others. Furthermore, the pressure gradient along the z-direction decreases towards the slip wall. The pressure drop measured across the channel for all configurations for both inflow velocities is listed in Table 7.6. The magnitude of the normalized pressure given in the color scale of the figure is different from the pressure drop listed in the table because the later is averaged along the entire inlet and outlet plane. As observed in the previous section, the pressure drop increases with increase in hydrodynamic angle $\beta$.

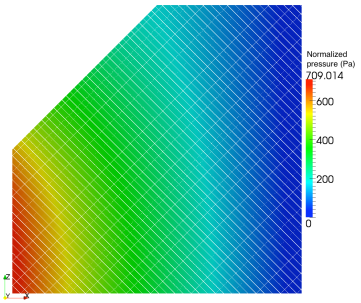
Similar to the pressure distribution, the flow distribution in the entire simulation domain along $xz$-plane at $y = h_{ch}/2$ for different spacer configurations for inflow mean velocity $\bar{v}_{in} = 0.03\ \mathrm{m\,s^{-1}}$ is shown in Figure 7.25. The blue color near the corner represents the flow with very low velocity which is below 0.01 $\mathrm{m\,s^{-1}}$. The configuration No.1 with angle $\beta = 30°$ has a large low flow area. The low flow area decreases with increase in hydrodynamic angle $\beta$. Thus it is possible to decrease the low flow area but very hard to get rid of them. Far from the corner towards the slip




(a) Configuration No. 1




(b) Configuration No. 2

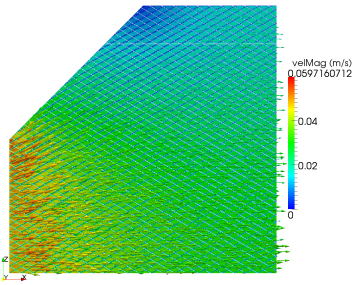



(c) Configuration No. 3

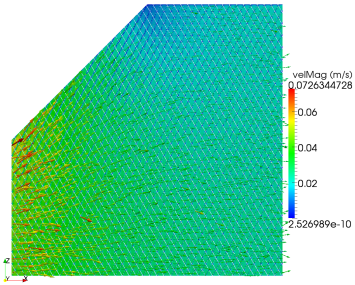
**Figure 7.24** Pressure distribution in the entire simulation domain along $xz$-plane at $y = h_{ch}/2$ for different spacer configurations for inflow mean velocity $\bar{v}_{in} = 0.03\ \mathrm{m\,s^{-1}}$.

| Configuration | $\beta$ | nF/inch | Inflow velocity $[\mathrm{m\,s^{-1}}]$ | $\Delta P$ for 7.1 cm kPa |
|---|---|---|---|---|
| 1 | 30° | 14 | 0.03 | 0.515 |
| | | | 0.05 | 0.863 |
| 2 | 60° | 14 | 0.03 | 0.742 |
| | | | 0.05 | 1.342 |
| 3 | 45° | 18 | 0.03 | 0.668 |
| | | | 0.05 | 1.121 |

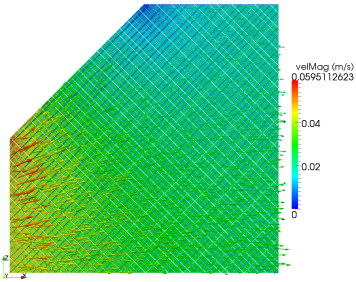
**Table 7.6** Pressure drop $\Delta P$ across the simulated length on 3 different mesh configuration and each with 2 different inflow mean velocities.




(a) Configuration No. 1




(b) Configuration No. 2

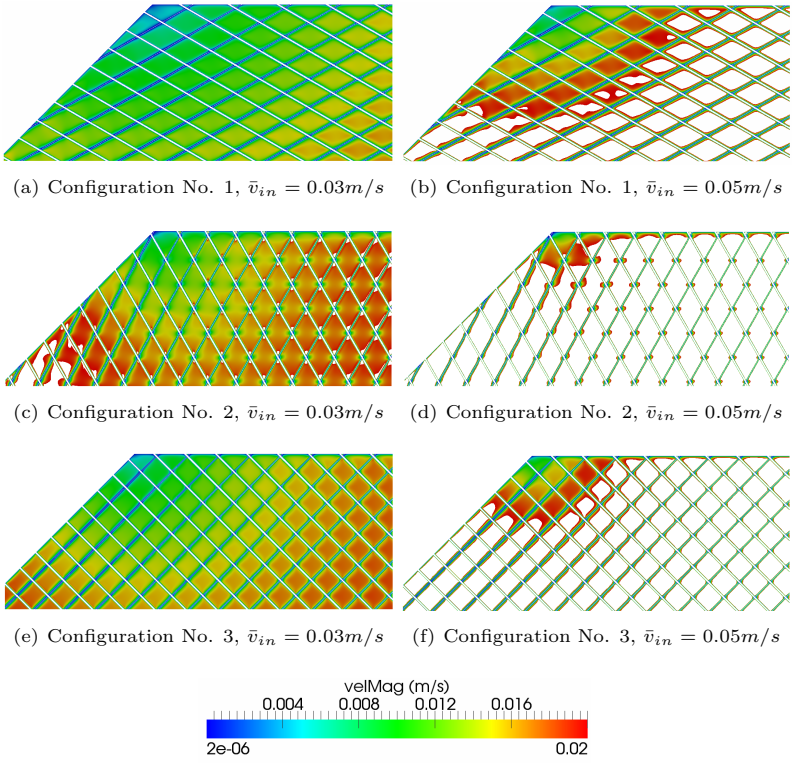



(c) Configuration No. 3

**Figure 7.25** Flow distribution and velocity vector in entire simulation domain along $xz$-plane at $y = h_{ch}/2$ for different spacer configurations for inflow mean velocity $\bar{v}_{in} = 0.03\ \mathrm{m\,s^{-1}}$.

wall, the flow is more uniformly distributed in all three configurations. The flow velocity is high near the inlet region and then it decreases close to the inflow velocity along the length. Only for the larger angle $\beta = 60°$, the flow velocity decreases slightly below the inflow velocity. Thus, the velocity magnitude in the bulk of the spacer-filled flow channel is close to the inflow velocity and the velocity increases only near the inlet region. In previous analysis, this was not observed because the length of the simulation domain was 10 times smaller. Another important difference between the two investigation is the diameter of the filament $d_f$ and the distance between the filament $l_m$ are different. The $l_m$ is larger and $d_f$ is smaller than previous investigation.

To compare low flow zones of different configuration, the area near the blocked corner with the length of 2.5 cm and the width of 1.0 cm is used.




(a) Configuration No. 1, $\bar{v}_{in} = 0.03m/s$ (b) Configuration No. 1, $\bar{v}_{in} = 0.05m/s$

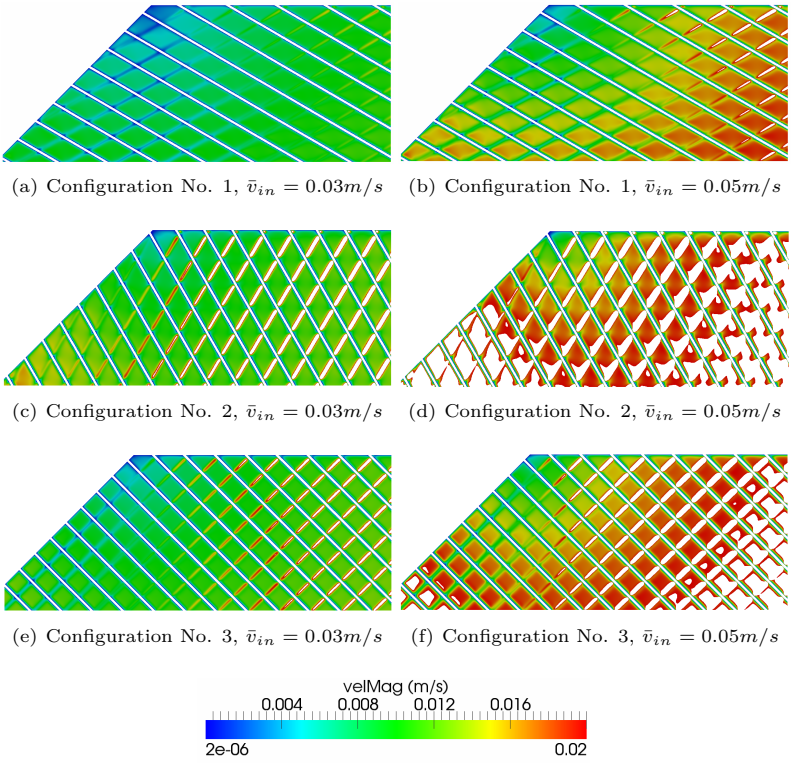
(c) Configuration No. 2, $\bar{v}_{in} = 0.03m/s$ (d) Configuration No. 2, $\bar{v}_{in} = 0.05m/s$

(e) Configuration No. 3, $\bar{v}_{in} = 0.03m/s$ (f) Configuration No. 3, $\bar{v}_{in} = 0.05m/s$

(g) Color map

**Figure 7.26** Velocity distribution at plane $y = h/2$ near blocked corner on different spacer configuration and inflow velocities. Velocity is shown only for velocity above threshold of $0.02m/s$, thus white space between the filaments have velocity $> 0.02m/s$.

Additionally, the velocity magnitude threshold is set to 0.02 $\mathrm{m\,s^{-1}}$ to see only the flow with low velocities since only they result in scaling and fouling effects in the spacer-filled flow channels. Figure 7.26 and Figure 7.27 show the velocity distribution near the blocked corner in the $xy$-plane at $y = h_{ch}/2$ and $y = 3h_{ch}/4$ for all 6 simulations. The left side of the figure is for $\bar{v}_{in} = 0.03$ $\mathrm{m\,s^{-1}}$ and the right for $\bar{v}_{in} = 0.05$ $\mathrm{m\,s^{-1}}$. The white spaces in the flow channel are either spacer channel or area




(a) Configuration No. 1, $\bar{v}_{in} = 0.03m/s$ (b) Configuration No. 1, $\bar{v}_{in} = 0.05m/s$

(c) Configuration No. 2, $\bar{v}_{in} = 0.03m/s$ (d) Configuration No. 2, $\bar{v}_{in} = 0.05m/s$

(e) Configuration No. 3, $\bar{v}_{in} = 0.03m/s$ (f) Configuration No. 3, $\bar{v}_{in} = 0.05m/s$

(g) Color map

**Figure 7.27** Velocity distribution at plane $y = 3h/4$ near blocked corner on different spacer configuration and inflow velocities. Velocity is shown only for velocity above threshold of $0.02m/s$, thus white space between the filaments have velocity $> 0.02m/s$.

with velocity above the threshold of 0.02 $\mathrm{m\,s^{-1}}$. The flow velocity in the $xz$- plane $y = h_{ch}/2$ is higher than in the $xz$- plane $y = 3h_{ch}/4$ which is similar to Figure 7.15 and Figure 7.16. In other words, the low flow area is higher in the plane $y = 3h_{ch}/4$ than in the plane $y = h_{ch}/2$. As observed before, the flow velocity is high in the free flow area and low near filaments. The flow has very low velocity only at the corner and the area decreases when the inflow velocity is increased to 0.05 $\mathrm{m\,s^{-1}}$. In both

planes, the low flow zone is more apparent with 60° angle and reduces with increase in angle. It is obvious that with higher velocity the most part of the domain has a velocity above the threshold value. Thus, a high angle $\beta$ reduces the low flow zones but increases the pressure drop across the channel. Therefore, the optimal configuration would be a 45° angle which has moderate pressure drop and uniform flow distribution around the blocked corner.

### 7.1.3 Multicomponent flows

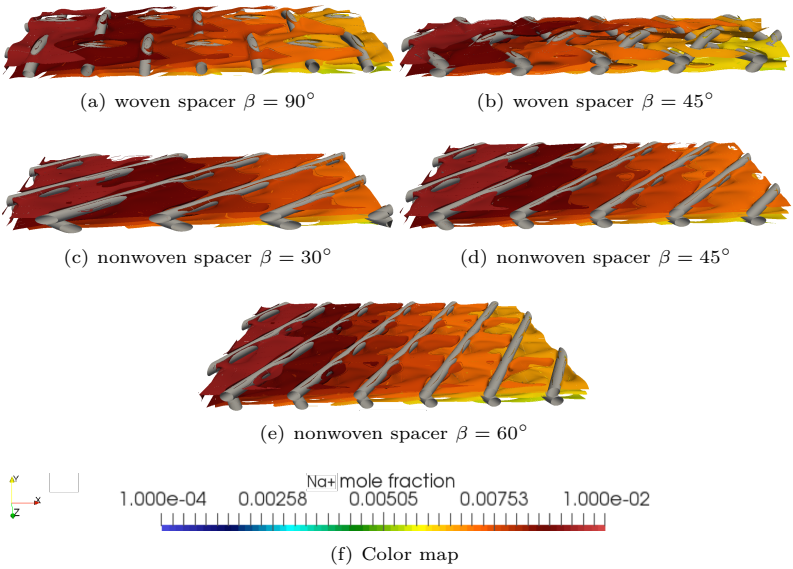
In this section, multicomponent flow simulations performed on the five different spacer configurations introduced in Section 7.1.1 are presented. This investigation is performed to see the distribution of charged species and charge density in the spacer-filled flow channels. A ternary liquid of mixture of $NaCl$ solution with a concentration of $c_{Na^+} = c_{Cl^-} = 512.5\ \mathrm{mol\,m^{-3}}$ is used. A molar concentration of solvent is $c_{H_2O} = 55965\ \mathrm{mol\,m^{-3}}$. Here, the species 1, 2 and 3 corresponds to $H_2O$, $Na^+$ and $Cl^-$. Other properties of the mixture are chosen as listed in Table 6.1 except the diffusivity coefficient of the species. The chosen diffusivity coefficient is $10^3$ times larger than the actual diffusivity coefficient of this mixture. This choice was due to the stability of the simulation w.r.t the chosen resolution $nHeight = 64$. The large diffusivity coefficient can have an effect on the numerical accuracy but should be sufficient to study the species transport on different spacers. The valence $z_k$ of species 1, 2 and 3 are 0, 1 and -1 respectively.

The initial conditions are chosen as

$$\begin{aligned} \chi_2 = \chi_3 = \epsilon_s, \quad \chi_1 = 1 - \chi_2 - \chi_3 \\ \boldsymbol{v}_1(\boldsymbol{x}) = \boldsymbol{v}_2(\boldsymbol{x}) = \boldsymbol{v}_3(\boldsymbol{x}) = 0 \end{aligned}$$

where $\epsilon_s = 1 \times 10^{-4}$ represents a smallness parameter to avoid division by zero. The boundary condition are imposed as follows: At the inlet, the mole fraction of the species according to the concentration of a chosen solution are specified using mole fraction equilibrium BC and at the outlet, the velocity profile is specified according to Eq. 6.3 with a velocity of $0.1\ \mathrm{m\,s^{-1}}$. The bottom and top boundaries are treated as AEM and CEM for ionic species and no-slip wall for solvent species 1. The membrane black-box model is used to define the AEM and CEM. The transport number of $Na^+$ and $Cl^-$ on CEM and AEM are $T^{CEM}_{Na^+} = 0.971$ and $T^{AEM}_{Cl^-} = 0.998$. The spacer filaments are approximated by q-values and treated by higher order wall BC. This boundary conditions are similar

to the Taylor dispersion verification test case with an external electrical force discussed in Section 6.4 except for inlet and outlet. Furthermore, a constant electrical force of $E_y = 5 \times 10^{-5}$ V m$^{-1}$ is applied along the y-direction which is perpendicular to the main flow direction to transport the ionic species towards the membranes. The inflow velocity is ramped up to 0.025 s and an electrical force is activated only after this ramping time is reached. The simulations are then ran long enough up to 0.35 s until they reach the steady state.




(a) woven spacer $\beta = 90°$

(b) woven spacer $\beta = 45°$

(c) nonwoven spacer $\beta = 30°$

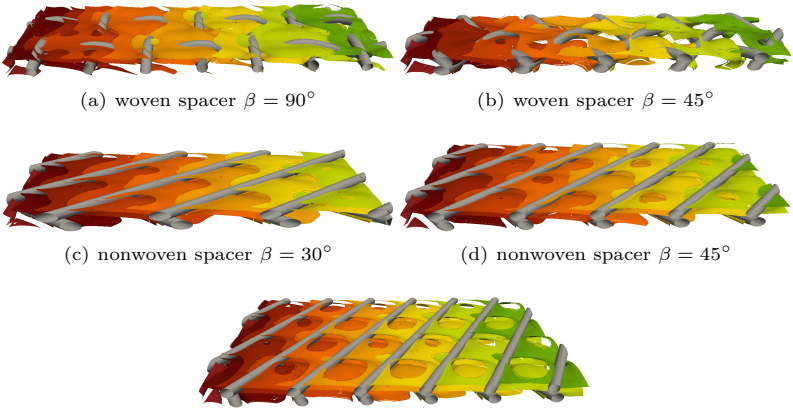
(d) nonwoven spacer $\beta = 45°$

(e) nonwoven spacer $\beta = 60°$

(f) Color map

**Figure 7.28** Contours of species 2 ($Na^+$) mole fraction for inflow mean velocity $\bar{v}_{in} = 0.1$ m s$^{-1}$ and a constant electrical field in y-direction for different spacer configurations

Figure 7.28 and Figure 7.29 show the contours of mole fraction of species 2 and 3 respectively in different spacer configurations. The dark red color at the inlet is the inlet mole fraction of ionic species i.e. 0.0092 and along the length the concentration of ionic species decreases due to membrane boundary conditions on top and bottom. Since an electrical field is applied along the positive y- direction, the positive species $Na^+$ is driven towards the top while the negative species $Cl^-$ is driven in opposite direction towards to the bottom. Also due to the higher transport number of AEM




(a) woven spacer $\beta = 90°$

(b) woven spacer $\beta = 45°$

(c) nonwoven spacer $\beta = 30°$

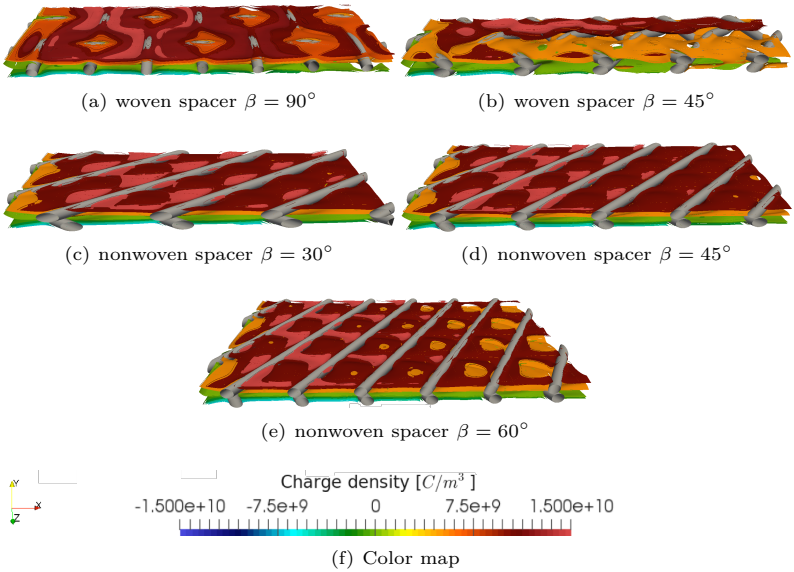
(d) nonwoven spacer $\beta = 45°$

(e) nonwoven spacer $\beta = 60°$

(f) Color map

**Figure 7.29** Contours of species 3 ($Cl^-$) mole fraction for inflow mean velocity $\bar{v}_{in} = 0.1\ \mathrm{m\,s^{-1}}$ and an electrical field in y-direction for different spacer configurations

at the bottom, the concentration of $Cl^-$ gets depleted faster than $Na^+$ along the length of the channel. Among the nonwoven spacers, the angle $\beta = 60°$ has the least concentration at the outlet. This might be due to the fact that the fluid takes a longer path to the outlet along the filament as observed in pure hydrodynamic simulations and thus the IEM have more time to deplete ions out of the flow channel. On the other hand, in woven spacers due to sinusoidal flow around the filaments, the fluid takes an even longer time than in the nonwoven spacer and it results in the decrease in concentration at the outlet of the woven spacer compared to the nonwoven spacers. Between woven spacers, the concentration at the outlet of rotated spacer $\beta = 45°$ is lower than the spacer $\beta = 90°$. This could be because in the woven spacer $\beta = 45°$, the fluid changes its direction at every filament intersection and also travels around the filament which might have increased the travel path even longer. Thus, the longer the travel distance of the fluid from inlet to outlet, the better is the transport of ions through the membranes.




(a) woven spacer $\beta = 90°$

(b) woven spacer $\beta = 45°$

(c) nonwoven spacer $\beta = 30°$

(d) nonwoven spacer $\beta = 45°$

(e) nonwoven spacer $\beta = 60°$

(f) Color map

**Figure 7.30** Contours of charge density $\rho^e$ for inflow mean velocity $\bar{v}_{in} = 0.1\ \mathrm{m\,s^{-1}}$ and an electrical field in y-direction for different spacer configurations

Similar to mole fraction contours, the charge density contours on different spacer configurations are shown in Figure 7.30. The charge density is computed from the sum of the concentrations with its valance so this figure can be considered as a composite plot of mole fraction contours of species 2 and 3, i.e. Figure 7.28 and Figure 7.29. Most of the domain is nonelectroneutral due to an applied external electrical force. A green contour represents a charge density of exact zero (electroneutral), which is along the middle of the channel for nonwoven spacers. For woven spacers, they are below the middle plane. In all spacer configurations, the charge density is high at low flow zones observed in the hydrodynamic simulations and the concentration increases in those areas with time. In woven spacer $\beta = 90°$, the charge density between the filament running upwards along the z-direction and the membrane is high. Furthermore, the concentration in the free flow area between filaments is low. Thus the larger the free flow area is the better the transport of ions through the membranes. This free flow area can be increased by increasing the angle between the filaments

$\alpha$ and increasing the distance between the filaments $l_m$. On the contrary, increasing the angle increases the pressure drop across the channel and introduces high fluctuations in the flow channel. Thus, from all the spacer investigations the optimal angle is $\beta = 45°$ for both woven and nonwoven spacers.

## 7.2 Repeating unit in ED stack

Finally, a single repeating unit of the ED stack with a dilute and concentrate channel as in Figure 4.2 is simulated in this section. A setup for the repeating unit is just an extension to the setup discussed in Section 6.4 for the Taylor dispersion test case with an external electrical force. Instead of a single flow channel with membranes, two flow channels are simulated simultaneously with an alternate arrangement of membranes resulting in dilute and concentrate channels. The dimensions of each channel are $L \times H$ : $0.2 \times 0.04$ cm which are similar to that test case. Also, the properties of an aqueous $NaCl$ solution like viscosities, density, molecular weight of the species, etc, and the transport number of the membranes are same as in Taylor dispersion test case. Once again the species 1, 2 and 3 corresponds to $H_2O$, $Na^+$ and $Cl^-$ respectively.

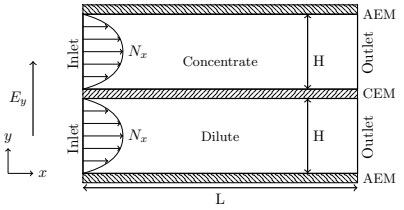



**Figure 7.31** Simulation setup for a repeating unit in ED stack without spacer filaments.

Unlike a strip of concentration in the flow channel, here, the flow is uniformly distributed with following initial conditions

$$\chi_2 = \chi_3 = \epsilon_s, \quad \chi_1 = 1 - \chi_2 - \chi_3$$
$$\boldsymbol{v}_1(\boldsymbol{x}) = \boldsymbol{v}_2(\boldsymbol{x}) = \boldsymbol{v}_3(\boldsymbol{x}) = 0$$

where $\epsilon_s = 1 \times 10^{-3}$ represents a smallness parameter to avoid division by zero. The boundary conditions are defined as follows

**inlet:** At the inlet of both channels, the x-component of the species mole

flux ($N_{k,x}$) is defined as a parabolic Poiseuille profile

$$N_{k,x}(y) = c_k(y)\frac{4v_m y(H-y)}{H^2} \tag{7.10}$$

where the molar concentration $c_k(y)$ of species 1, 2 and 3 are $c_1 = 54940\ \mathrm{mol\,m^{-3}}$, $c_2 = 512.5\ \mathrm{mol\,m^{-3}}$ and $c_3 = 512.5\ \mathrm{mol\,m^{-3}}$. The maximum velocity at the middle of the channel is $v_m = 0.1\ \mathrm{cm\,s^{-1}}$.

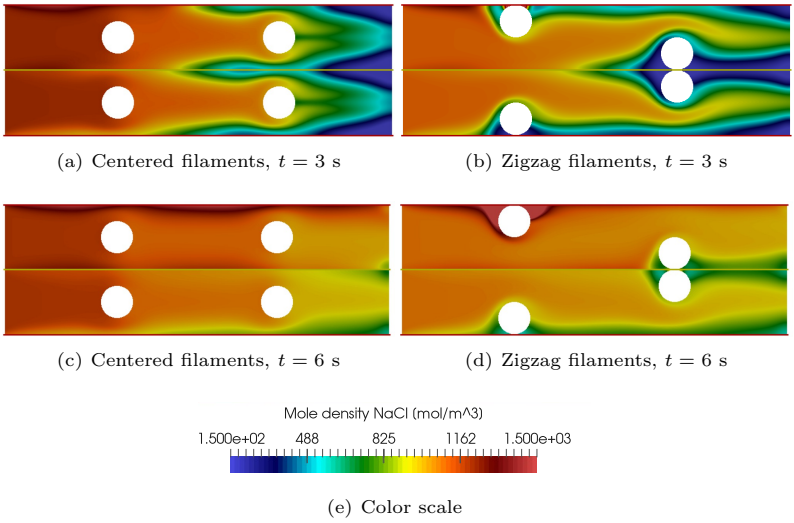
**AEM:** Solvent $H_2O$ is treated with simple no-slip wall and ionic species are treated with membrane black box model with the transport number for ionic species as $T_{Na^+,AEM} = 1 - 0.998$ and $T_{Cl^-,AEM} = 0.998$.

**CEM:** Treated same as AEM, with different transport number of ionic species $T_{Na^+,CEM} = 0.971$ and $T_{Cl^-,CEM} = 1 - 0.971$.

**Outlet:** At both outlets, a simple equilibrium based BC is imposed by extrapolating the concentration and velocities of the species.

**Spacer:** Spacer filaments are approximated by q-values and treated by higher order wall BC.

Each channel is resolved with 128 elements in the height. The MRT collision operator with $D3Q19$ layout was used. Here, a constant electrical field is applied along the $y$- direction perpendicular to the main flow direction. Simulations are performed on two configurations: 1) centered filaments and 2) zigzag filaments. In zigzag filaments configuration, the position of filaments in the dilute and concentrate channels are swapped. Both configurations are run for $E_y = 5\ \mathrm{V\,m^{-1}}$ and $E_y = 2.5\ \mathrm{V\,m^{-1}}$, to see the effect of the externally applied electrical field on the species transport. Due to constant electrical field, there is no need to solve the LBM for electric potential so this is just a multicomponent flow simulation with membrane black box model using *Musubi*. Another reason for not performing a coupled multicomponent and electric potential simulation is that the current implementation of membrane black box model treats each channel as independent flow channel and does not exchange concentrations between them. In other words, the membrane black box model either removes or adds species concentration depending on the transport number of ionic species. To exchange concentration between the channels through membrane, a species transport in membranes must be modeled by the Nernst-Planck equations. There exist a LBM to solve the Nernst-Planck equation [101] which can be used and it will be implemented in *Musubi* in the near future.




(a) Centered filaments, $t = 3$ s

(b) Zigzag filaments, $t = 3$ s

(c) Centered filaments, $t = 6$ s

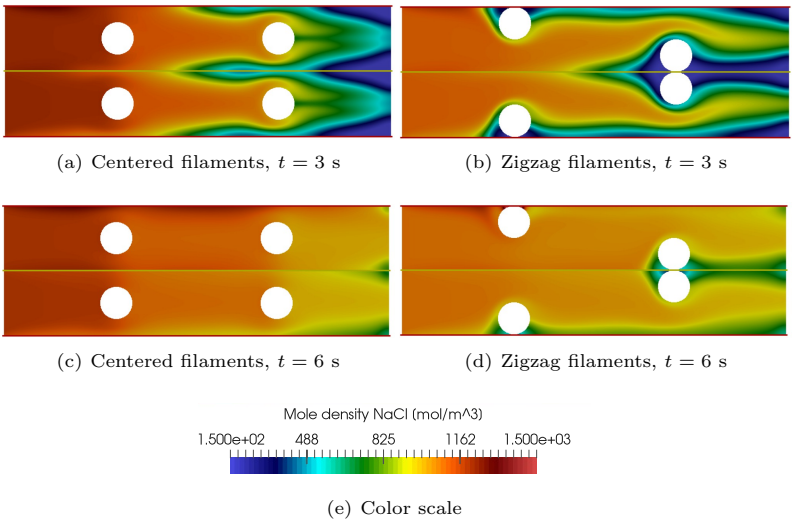
(d) Zigzag filaments, $t = 6$ s

(e) Color scale

**Figure 7.32** Molar concentration of $NaCl$ at $t = 3$ s (top) and $t = 6$ s (bottom) for centered filaments (left) and zigzag filaments (right) in a repeating unit with a constant electrical field $E_y = 5\ \mathrm{V\,m^{-1}}$.

Figure 7.32 shows the molar concentration of $NaCl$ ($c_{Na^+} + c_{Cl^-}$) at physical time $t = 3$ s and $t = 6$ s on centered and zigzag filaments configuration for a constant electrical field $E_y = 5\ \mathrm{V\,m^{-1}}$. It can be seen that in both configurations at $t = 6$ s, the concentration of $NaCl$ decreases along the length of the channel and there is a noticeable difference in concentration between dilute and concentrate channel. In both channels, the concentration of $NaCl$ is higher on the top boundary than the bottom because the transport number of $Na^+$ on the CEM is less than $Cl^-$ on the AEM so $Cl^-$ is removed from channel much faster than $Na^+$. Also, $Na^+$ migrates faster towards the CEM with an electrical force since its molecular weight is less than $Cl^-$. This is similar to the observation in Section 6.5 that with increased concentration the concentration profile of ionic species becomes asymmetric due to their different molecular weights. In centered filaments, the concentration of $NaCl$ on the top boundary in the concentrate channel is higher between the filaments because in the area between the filament and the membrane, the flow velocity is higher which transports the ionic species. On the other hand, in zigzag filaments, the concentration of $NaCl$ accumulates near the filament on the top boundary

in the concentrate channel and it increases with time. At $t = 6$ s, the concentration of $NaCl$ near the top filament in the concentration almost covers the entire filament due to the very low velocity in this region. This accumulation of ions near the filaments would block the bulk flow and reduce the overall efficiency of the ED process.




(a) Centered filaments, $t = 3$ s

(b) Zigzag filaments, $t = 3$ s

(c) Centered filaments, $t = 6$ s

(d) Zigzag filaments, $t = 6$ s

(e) Color scale

**Figure 7.33** Molar concentration of $NaCl$ at $t = 3$ s (top) and $t = 6$ s (bottom) for centered filaments (left) and zigzag filaments (right) in a repeating unit with a constant electrical field $E_y = 2.5\ \mathrm{V\,m^{-1}}$.

Now, the electrical force is reduced by half i.e. $E_y = 2.5\ \mathrm{V\,m^{-1}}$ and the molar concentration of $NaCl$ for both configurations at the same physical times are shown in Figure 7.33. As it can be seen in the figure, the accumulation of $NaCl$ near the filament in the concentrate channel of zigzag configuration is reduced drastically. Still there is a high concentration near the filament but it is much closer to the intersection of membrane and filament. The accumulation in the corner of filament/membrane intersection is unavoidable with such circular spacer filaments but can be reduced by decreasing the applied potential drop. Reducing an electrical force has no significant effect in centered filaments except the concentration of $NaCl$ on the top boundary in the concentration channel is less than before. Also, note that the concentration of $NaCl$ at the outlet of the zigzag configuration is less than in the centered filaments. This shows that

the zigzag configuration promotes the ionic species transport through the membranes and with the right potential drop, ionic species accumulation can be reduced.

The purpose of these simulations is to demonstrate that it is possible to simulate a targeted single ED unit with a dilute and concentrate channel using the presented simulation framework. Thus, we can conclude that using the *APES* framework, we could simulate the multiphysics phenomena in ED process. Using numerical simulations we could understand the process better and also find the optimal spacer configuration and operating conditions like inflow velocity and an applied potential across the ED stack.

# 8 Summary and Future work

In this final chapter, the summary of contributions of this thesis and the drawn conclusions are presented. Along with that, some future work in the direction of further development of the simulation framework and next steps towards more detailed coupled multiphysics simulations of ED process are described. In general, this thesis presented mathematical equations to describe multiphysics (electro-convection-diffusion) phenomena involved in the ED process and the lattice Boltzmann method to numerically solve those physical equations and the coupled simulation framework with the implementation details and algorithm, and finally the results of large scale flow simulations with spacer geometry. It also presents the strategy of coupling domains of different physics.

## 8.1 Summary

The conclusions drawn and the contributions of this thesis are summarized as follows

**Mesh generator - *Seeder*:** An effective automatic octree mesh generator algorithm was developed and implemented as *Seeder* in the *APES* framework. It generates computational meshes of cubical elements from STL files and predefined shapes. The fluid elements are linearized using SFC and they are written to disk along with boundary information. The performance of *Seeder* to generate computational mesh with spacer geometry was presented and it was shown that it can generate mesh with several hundred million of fluid elements in less than an hour.

**Incompressible LBM:** A prior implementation of our highly scalable LB solver only supported the weakly compressible LBE for acoustic flow simulation. Therefore, the incompressible LBE and boundary conditions to handle Dirichlet inlet velocity and Dirichlet outlet pressure boundaries required to simulate the fluid flow in spacer-filled flow channels was implemented in *Musubi* as part of this thesis. This implementation was validated with a well-known Poiseuille flow test case.

**Multicomponent LBM:** The multicomponent LBM for ideal and nonideal liquid mixtures with external forces presented in Section 3.2 was developed from the Asinari's ideal gas mixture model [6] and implemented in the highly scalable LB solver *Musubi*. The nonideal multicomponent model recovers the complex equations of nonideal mass and momentum transport, i.e. the Maxwell-Stefan equations with external diffusive force for species and the Navier-Stokes equation with external force for the mixture. In contrast to the ideal model, the nonideal model includes thermodynamic factors and concentration dependent diffusivity coefficients. The comparison of ideal and nonideal model are presented in Section 6.5. It showed that nonideal effects are higher for concentrated solutions especially in the EDL region. the ED process is widely used for seawater desalination and its concentration is rather higher. Also it is known that a concentration gradient exist at the membrane/electrolyte interface so it is necessary to use the nonideal model to simulate the multicomponent flow with spacer geometry. To best of my knowledge, there are no other LBM solvers capable of simulating nonideal liquid mixtures. Also, it is the first time, a detailed comparison study was performed on ideal and nonideal model on an aqueous $NaCl$ solution with different concentrations and surface potentials. The implementation of multicomponent LBM was validated and verified using various test cases like Stefan tube, Taylor dispersion and EDL. The EDL test case is used to validate the fully coupled setup of the multicomponent LBM and the LBM for the electric potential.

**Electric potential LBM:** The Poisson equation, which describes the distribution of electric potential with respect to local change in charge density distribution, is also implemented in the LB solver *Musubi*. A non-equilibrium extrapolation boundary condition for both Dirichlet and Neumann boundary were implemented. The implementation was validated using a concentric cylinder test case in which cylinders are approximated by q-values and the numerical results showed very good agreement with the analytical solution.

**Performance:** The multicomponent LBM compute kernel was implemented for arbitrary number of species and stencil. Additionally, an optimized compute kernel for the three species mixture on the $D3Q19$ stencil was implemented and its performance on the Hermit system, HLRS, Stuttgart was presented in Section 5.4.5. In addition to multicomponent performance, the single component LBM performance

was also presented on the complex spacer structure. Both single component and multicomponent LBM show very good scalability on the Hermit system and they achieved a sustained performance of 4.2% and 7.2% respectively. The performance of the multicomponent solver is better due to the additional number of floating point operations involved in solving the linear equation system for species momentum.

**Load balancing:** Dynamic load balancing from *TreElM* was deployed for the multicomponent flow simulation with spacer to reduce the load imbalances from the inlet and outlet boundaries.

**Membrane model:** A black-box membrane model was introduced to mimic the behavior of membranes in the ED process. It uses a transport number of for each species for a particular membrane to define the mass flux of a species through that membrane. This model was implemented as a BC in the LBM solver *Musubi* and verified using the Taylor dispersion test case.

**Integrated coupling tool - *APESmate*** . The development of this coupling tool to couple *APES* solvers to perform coupled multiphysics simulations is the major contribution of this work. In *APESmate*, a multiphysics domain is partitioned into several sub-domains according to the physical system and sub-domains can interact with each other either via surface or via volume coupling. The BC and source terms in solver are used to exchange data at the surface and volume coupling interfaces respectively between sub-domains. *APESmate* can handle both these couplings by exchanging coupling variables at coupling points evaluated using the solver data structure thus without losing numerical accuracy at the coupling interface. The two key features used to realize this exchange of coupling variables and points are space-time function and variable system, which were presented in detail. *APESmate* is developed with scalability in large-scale systems in mind so it can distribute sub-domains across several processes. The global MPI communicator is used to exchange coupling data between sub-domains and MPI sub-communicators are used for communication within a sub-domain. Two types of domain decomposition were implemented: 1) one domain per processes 2) all domains per process. The former one is used especially for surface coupling where only few processes contain a coupling surface and are involved in coupling data exchange and evaluation. Whereas in volume coupling, all processes contain coupling data so the latter is

beneficial. An advantage of *APESmate* is that it does mesh independent coupling because it obtains variables at physical coordinates either by interpolation or polynomial evaluation depending on the solver. I.e, it does interpolation or polynomial evaluation for a given set of points in space to exchange data between sub-domains with coinciding or non-coinciding meshes. It can handle an arbitrary number of sub-domains. Data are exchanged between sub-domains at every synchronization time and this time is determined by the maximum physical time steps of all sub-domains except for sub-domains, which need to solve a steady state problem. The current implementation allows for sub-cycling in a domain when its time step is below the maximum time step but no data will be exchanged at the sub-cycling step, which might result in numerical inconsistency. The time interpolation is required to resolve this issue and it is planned for future work. Thus, the current version of *APESmate* limits sub-domains to have same physical time steps.

**Flow simulation with spacer** Hydrodynamic flow simulations on five distinct spacer configurations i.e. both woven and nonwoven with different hydrodynamic angle for various inflow velocities were investigated. For each configuration, the pressure drop across the channel and the flow distribution in the channel were measured and compared with each other. Comparison revealed that the pressure drop increases with hydrodynamic angle and the nonwoven spacer $\beta = 30^\circ$ has the least pressure drop of all configurations for all inflow velocities. For the same hydrodynamic angle $\beta = 45^\circ$, the nonwoven spacer has the least pressure drop in comparison to the woven spacer. Furthermore, for all velocities, the woven spacer $\beta = 45^\circ$ and the nonwoven spacer $\beta = 30^\circ$ reach steady state and show no fluctuations in the flow. Increasing the hydrodynamic angle increases fluctuations in the flow and for larger angle $\beta = 60^\circ$, high fluctuations are observed from inflow velocity of 0.4 $\mathrm{m\,s^{-1}}$. For those high fluctuating flows, the power spectrum density over frequency was analyzed and compared with Kolmogorov $-5/3$ scale energy decay. It was found that this flow looses its energy much faster than other turbulent flows due to presence of spacer structure. In ED process, an optimal spacer is one with low pressure drop across the channel and reduced low flow zones. The configuration that satisfies both of these conditions are both the both woven and nonwoven spacer with $\beta = 45^\circ$. The main difference between the flow in woven and nonwoven spacer is that in the woven spacer the fluid flows around the filament in a zigzag pattern and

flows mainly along the main flow direction but in nonwoven spacer, the fluid changes its direction at every filament intersection which results in a longer path to outlet. This observation was confirmed by the multicomponent flow simulations with those spacer configurations and showed that the spacer with $\beta = 45^\circ$ improves the ions transport through the membrane. In addition to the flow simulations with periodic width, the flow near a sealed corner was also simulated because in real ED stack the spacers have a seal at all four corners and this corner is known to have low flow zones which are the main cause for scaling and fouling in spacer channels. Therefore, very large scale simulations were performed on three nonwoven spacer configurations with the sealed corner to identify the configurations with reduced low flow zones. It was found that the nonwoven spacer with $\beta = 45^\circ$ has less low flow zones compared to others. From these results, we can conclude with $\beta = 45^\circ$ is the optimal hydrodynamic angle for spacer flow channels.

**Repeating unit:** At last, a targeted simulation setup with a dilute and concentrate channel was presented for centered and zigzag filament configurations. These simulations showed that it is possible to simulate two channels simultaneously with different membrane boundary conditions. The results showed that in zigzag configuration, in the concentrate channel, an accumulation of ionic species near the filament and membrane can be reduced by decreasing the potential drop across the ED stack.

## 8.2 Future work

This work can be continued in two directions: software development and application (ED process). Software development would involve implementation in *APES* simulation framework and application would involve systematic detailed investigation of fully coupled multiphysics simulation of ED process. Here are the following aspects for future work in those two directions:

**Parallel *Seeder*:** The presented mesh generation algorithm needs to be MPI parallelized to reduce the mesh generation time and also to generate larger computational mesh with billion fluid elements, which up to now are limited because of available memory of a single node.

**Second-order force in *Musubi*:** In LBM, the fully discrete LBE of second-order is obtained by integrating the semi-discrete LBE along the

characteristics and approximating the integral by the trapezoidal rule [21]. For the time discretization of LBE see App. A.1.1.1. Applying the trapezoidal rule results in an implicit equation so to convert the equation back to explicit form, a variable transformation technique is used by introducing a new set of distribution functions. The LBE equation is then written using a new set of distribution functions (transformed PDF) just by altering the relaxation parameter. Due to the collision invariance, the conserved quantities like mass and momentum are readily reconstructed from transformed PDF. However, this transformation affects the shear stress [21]. To maintain the second-order accuracy of the scheme, an external force term in LBE is also integrated with trapezoidal rule and this affects the momentum calculation [20]. In *Musubi*, the stream-collide step and application of the force term are performed in two separate routines which makes the current implementation unsuitable for second-order force term. Therefore, to circumvent this problem, i.e. to avoid an external force contribution on the momentum calculation, the forward Euler is applied to the integration of the external force term in LBE resulting in first-order accurate forcing term. Thus, with an external force, the current implementation would result in a first-order accurate scheme. In the future, the second-order force term can be implemented to improve numerical accuracy and stability. This implementation would require quite some redesign of the code because an external force must be provided whenever the momentum is computed from PDF which is compute kernel, apply force, boundary condition and tracking. This limitation is also applicable to an external force term in multicomponent LBE and source term in LBE for electric potential, and must be resolved in the future.

**BC for multicomponent LBM:** More accurate and robust BC than the simple equilibrium BC is required to treat multicomponent outflow. The second-order accurate non-equilibrium extrapolation BC [27] would suit well for this purpose.

**Time interpolation in *APESmate*:** This would allow for sub-domains with different time steps i.e. sub-cycling of a sub-domain. At each sub-cycling step, the coupling variables can be extrapolated in time and the accuracy of this extrapolation depends on the number of previous time step values stored in memory. Of course, this would increase the memory consumption but it would payoff when the computational cost of a sub-domain decreases by using larger element size and time

step size.

**Coupling performance:** In *APESmate* algorithm, the initialization of the coupling can be expensive because it needs to identify the right process to communicate through round-robin fashion but at each time step, it only evaluates a variable using solver routines and exchange data. So, the only potential routine, which would affect the performance of the coupling is the variable evaluation. However, performing a code profiling would provide more insight into identifying performance bottlenecks. Additionally, a rigorous scaling analysis needs to be done on large-scale systems similar to the solver scaling analysis presented in Section 5.4.5.

**Membrane model:** A sophisticated membrane model described by Nernst-Planck equation can be developed. In particular, LBM based Nernst-Planck model proposed by [87, 88] can be implemented in *Musubi*.

**ED process simulation:** After implementing LBM for Nernst-Planck mode, a fully coupled multiphysics in ED process can be simulated using *APESmate* i.e. coupling multicomponent LBM for flow channel and LBM based Nernst-Planck for membrane and LBM based electric potential for the entire domain. In this coupled setup, the spacer structure can be considered and also a nonideal multicomponent LBM model can be used. Thus, using *APESmate*, a detailed multiphysics simulations of ED process can be performed by varying parameters like inflow velocity, applied potential drop, spacer design, membrane properties and so on. These simulations would help to find the optimal operating conditions for ED processes.

**Coupled simulations:** *APESmate* is an efficient coupling tool, which can be used to simulate any multiphysics applications like aero-acoustic, electro-osmotic flow and so on. Electro-osmotic flow simulations can be performed without any implementation because the physical phenomena required are electrostatic interaction and fluid flow that are already available as mutlicomponent LBM and LBM for electric potential in *Musubi*.

**Extension of *APESmate* for external solvers:** At the moment, *APESmate* is limited only to solvers in *APES* and it can extended by developing a wrapper for an external solver to support space-time functions and variable system features.

# A Appendix

## A.1 Asymptotic analysis

In this session, the asymptotic analysis [41] is used to recover macroscopic equations from the semi-discrete Boltzmann equation, i.e. the Boltzmann equation discretized in velocity space and continuous in space and time. In some literatures, the semi-discrete Boltzmann equation is referred to as the continuous finite discrete velocity model (FVDM) Boltzmann equation. In App. A.1.1, the incompressible Navier-Stokes equation for fluid flow is recovered. In App. A.1.2, the multicomponent Maxwell-Stefan equations for species transport and the incompressible momentum equation for mixture transport are recovered. Finally, in App. A.1.3 the Poisson equation for electrical potential is recovered. In each of these sections, the continuous Boltzmann equation is discretized in time to obtain the fully discrete lattice Boltzmann equation.

### A.1.1 Lattice Boltzmann Method for Fluid Flow

Following derivation of an asymptotic analysis, recovers the incompressible Navier-Stokes equations from the semi-discrete Boltzmann equation Eq. 3.1. In [41], the zero order density $\rho^{(0)}$ is set to 1, whereas in the derivation below, it is fixed to initial density $\rho^{(0)} = \rho_{t=0}$. The asymptotic analysis for the generalized MRT continuous Boltzmann equation is given in [41]. Lets start with the BGK semi-discrete Boltzmann equation with body force on discrete velocity sets $\boldsymbol{u}^m$ which is same as Eq. 3.1,

$$\partial_t f^m + \boldsymbol{u}^m \cdot \nabla f^m = \underbrace{\lambda_\nu (f^{eq,m} - f^m)}_{:=\Omega^m} + F^m. \tag{A.1}$$

Here, we apply diffusive scaling proposed in [41] i.e. $\widetilde{\delta x} = \epsilon \delta x$ and $\widetilde{\delta t} = \epsilon^2 \delta t$ to above equation. A smallness parameter $\epsilon$, corresponds to Mach number $Ma = U/c_s$ with the flow velocity $U$ and the speed of sound $c_s$ or Knudsen number $Kn = l_m/L$ with molecular mean free path $l_m$ and physical length scale $L$

$$\epsilon^2 \partial_t f^m + \epsilon \boldsymbol{u}^m \cdot \nabla f^m = \lambda_\nu (f^{eq,m} - f^m) + F^m. \tag{A.2}$$

Now, we expand the particle distribution functions $f^m$, the body force term $F^m$ and the macroscopic moments in an asymptotic expansion of the form:

$$f^m = \sum_{i=0}^{\infty} \epsilon^i f^{m,(i)} \tag{A.3a}$$

$$F^m = \sum_{i=0}^{\infty} \epsilon^i F^{m,(i)} \tag{A.3b}$$

$$\rho = \sum_{i=0}^{\infty} \epsilon^i \rho^{(i)} \tag{A.3c}$$

$$\mathbf{p} = \sum_{i=0}^{\infty} \epsilon^i \mathbf{p}^{(i)} \tag{A.3d}$$

Furthermore, the macroscopic moments such as density, momentum and momentum flux are defined via moments of $f$ as

$$\rho = \sum_m f^m, \quad \mathbf{p} = \rho^{(0)} \boldsymbol{v} = \sum_m \boldsymbol{u}^m f^m, \quad \boldsymbol{\Pi} = \sum_m \boldsymbol{u}^m \boldsymbol{u}^m f^m. \tag{A.4}$$

Lets consider the following equilibrium distribution function

$$f^{eq,m}(\rho^{(0)}, \rho, \boldsymbol{v}) = \omega^m \Bigg( \rho + \rho^{(0)} \Big( \frac{1}{c_s^2} (\boldsymbol{u}^m \cdot \boldsymbol{v}) + \frac{1}{2c_s^4} (\boldsymbol{u}^m \cdot \boldsymbol{v})^2 - \frac{1}{2c_s^2} (\boldsymbol{v} \cdot \boldsymbol{v}) \Big) \Bigg). \tag{A.5}$$

which is the Maxwell-Boltzmann distribution function for isothermal incompressible low Mach number flows. For convenience we split the equilibrium distribution function Eq. A.5 into linear, bi-linear and quadratic parts as follows,

$$f^{eq,m}(\rho, \rho^{(0)}, \boldsymbol{v}) = f^{eq,m,L}(\rho) + f^{eq,m,BL}(\rho^{(0)}, \boldsymbol{v}) + f^{eq,m,Q}(\rho^{(0)}, \boldsymbol{v}) \tag{A.6}$$

where,

$$f^{eq,m,L}(\rho) = \omega^m \rho \tag{A.7}$$

$$f^{eq,m,BL}(\rho^{(0)}, \boldsymbol{v}) = \omega^m \rho^{(0)} \frac{1}{c_s^2} (\boldsymbol{u}^m \cdot \boldsymbol{v}) \tag{A.8}$$

$$f^{eq,m,Q}(\rho^{(0)}, \boldsymbol{v}) = \omega^m \rho^{(0)}(\frac{1}{2c_s^4}(\boldsymbol{u}^m \cdot \boldsymbol{v})^2 - \frac{1}{2c_s^2}(\boldsymbol{v} \cdot \boldsymbol{v})). \tag{A.9}$$

In above equation, $\rho^{(0)}$ is the zero order density which is fixed at $t = 0$. Using Eq. 3.13 in linear, bi-linear and quadratic part of equilibrium function,

$$\langle 1, f^{eq,L} \rangle = \rho \tag{A.10a}$$
$$\langle 1, \boldsymbol{u} f^{eq,L} \rangle = 0 \tag{A.10b}$$
$$\langle 1, \boldsymbol{u} \otimes \boldsymbol{u} f^{eq,L} \rangle = c_s^2 \rho I \tag{A.10c}$$

$$\langle 1, f^{eq,BL} \rangle = 0 \tag{A.11a}$$
$$\langle 1, \boldsymbol{u} f^{eq,BL} \rangle = \rho^{(0)} \boldsymbol{v} \tag{A.11b}$$
$$\langle 1, \boldsymbol{u} \otimes \boldsymbol{u} f^{eq,BL} \rangle = 0 \tag{A.11c}$$

$$\langle 1, f^{eq,Q} \rangle = 0 \tag{A.12a}$$
$$\langle 1, \boldsymbol{u} f^{eq,Q} \rangle = 0 \tag{A.12b}$$
$$\langle 1, \boldsymbol{u} \otimes \boldsymbol{u} f^{eq,Q} \rangle = \rho^{(0)}(\boldsymbol{v} \otimes \boldsymbol{v}) \tag{A.12c}$$

Where, $I$ denotes the identity operator and $\otimes$ denotes tensor product between two vectors ($(\boldsymbol{a} \otimes \boldsymbol{b})_{\alpha\beta} = \frac{1}{2}(a_\alpha b_\beta + a_\beta b_\alpha)$).

Divide the semi-discrete Boltzmann equation Eq. A.2 by $\epsilon^2$ and apply asymptotic expansion of PDF Eq. A.3a with $f^{m,(i)} = 0$ for $i < 0$. We get the following expression in the order $\epsilon^{i+2}$ for $i \geq -2$:

$$\begin{aligned} \partial_t f^{m,(i)} + \boldsymbol{u}^m \cdot \nabla f^{m,(i+1)} &= \lambda_\nu f^{eq,m,L}(\rho^{(i+2)}) \\ &+ \lambda_\nu f^{eq,m,BL}(\rho^{(0)}, \boldsymbol{v}_1^{(i+2)}) \\ &+ \lambda_\nu \sum_{\substack{j,l \geq 0 \\ j+l=i+2}} f_k^{eq,m,Q}(\rho^{(0)}, \boldsymbol{v}^{(j)}, \boldsymbol{v}^{(l)}) \\ &- \lambda_\nu f^{m,(i+2)} + F^{m,(i+2)} \end{aligned} \tag{A.13}$$

We can rewrite above equation in terms of density moment using $\rho = \sum_m f^m$ and making use of Eq. A.10, Eq. A.11 and Eq. A.12,

$$\begin{aligned} \partial_t \rho^{(i)} + \nabla \cdot \mathbf{p}^{(i+1)} &= \lambda_\nu(\langle 1, f^{eq,L,(i+2)} \rangle + 0 + 0 - \rho^{(i+2)}) + g^{(i+2)} \\ &:= g^{(i+2)} \end{aligned} \tag{A.14}$$

where

$$g^{(i)} = \sum_m F^{m,(i)}. \tag{A.15}$$

Similarly, the momentum equation can be obtained by taking first moment of Eq. A.13 i.e. $\boldsymbol{p} = \sum_m u^m f^m$ and making use of Eq. A.10, Eq. A.11 and Eq. A.12

$$\begin{aligned}\partial_t \mathbf{p}^{(i)} + \nabla \cdot \langle 1, \boldsymbol{u} \otimes \boldsymbol{u} f^{(i+1)} \rangle = \lambda_\nu (0 + \langle 1, \boldsymbol{u} f^{eq,BL,(i+2)} \rangle + 0 - \mathbf{p}^{(i+2)}) \\ + \boldsymbol{h}^{(i+2)} := \boldsymbol{h}^{(i+2)}\end{aligned} \tag{A.16}$$

where

$$\boldsymbol{h}^{(i)} = \sum_m \boldsymbol{u}^m F^{m.(i)}. \tag{A.17}$$

Additionally, from [41] following relation holds true

$$F^{m,(i)} = \begin{cases} \omega^m \boldsymbol{u}^m \cdot \boldsymbol{G}^{(i)} & \text{, if } i \geq 3 \\ 0 & \text{, else} \end{cases} \tag{A.18}$$

where $\boldsymbol{G}^{(i)}$ is the macroscopic body force of form $\boldsymbol{G} = \rho \boldsymbol{g}$. The mass conservation equations can be obtained by setting $i = 0$ in Eq. A.14 and using $g^{(2)} = 0$ gives

$$\partial_t \rho^{(0)} + \nabla \cdot \mathbf{p}^{(1)} = 0. \tag{A.19}$$

Applying properties of incompressible fluids, $\rho^{(0)} = const$ and $\mathbf{p}^1 = \rho^{(0)} \boldsymbol{v}^{\mathbf{1}}$, results in the following continuity equation with velocity of order $\epsilon$

$$\nabla \cdot \boldsymbol{v}^{(1)} = 0. \tag{A.20}$$

Likewise, the momentum conservation of the incompressible Navier-Stokes equations can be obtained by setting $i = 1$ in Eq. A.16.

$$\partial_t \mathbf{p}^{(1)} + \nabla \cdot \langle 1, \boldsymbol{u} \otimes \boldsymbol{u} f^{(2)} \rangle = \boldsymbol{h}^{(3)}. \tag{A.21}$$

To determine unknown variable $f^{m,(2)}$, we must first determine its leading order coefficients $f^{m,(0)}$ and $f^{m,(1)}$. Setting $i = -2$ in Eq. A.13 and keeping in mind $f^{m,(i)} = 0$ and $F^{m,(i)} = 0$ for $i < 0$ results in,

$$f^{m,(0)} = f^{eq,L,m}(\rho^{(0)}) = \omega^m \rho^{(0)} \tag{A.22}$$

$$\Rightarrow \boldsymbol{v}^0 = 0 \tag{A.23}$$

Furthermore, in [41] it has been shown, that the following relations hold true:

$$\begin{aligned} \rho^{(2i+1)} &= 0 \quad \forall i \in \mathbb{N}_{\geq 0} \\ \boldsymbol{v}^{(2i)} &= 0 \quad \forall i \in \mathbb{N}_{\geq 0} \end{aligned} \tag{A.24}$$

For $i = -1$ in Eq. A.13,

$$\boldsymbol{u}^m \cdot \nabla f^{m,(0)} = \lambda_\nu (f^{eq,m,BL}(\rho^{(0)}, \boldsymbol{v}^{(1)}) - f^{m,(1)}) + F^{m,(1)}$$

Using Eq. A.22 and $\nabla \rho^{(0)} = 0$ (since $\rho^{(0)} = const$) and $F^{m,(1)} = 0$,

$$f^{m,(1)} = f^{eq,m,BL}(\rho^{(0)}, \boldsymbol{v}^{(1)}) = \frac{\omega^m}{c_s^2} \rho^{(0)} \boldsymbol{u}^m \cdot \boldsymbol{v}^{(1)}. \tag{A.25}$$

Finally to obtain $f^{m,(2)}$, set $i = 0$ in Eq. A.13,

$$\partial_t f^{m,(0)} + \boldsymbol{u}^m \cdot \nabla f^{m,(1)} = \lambda_\nu (f^{eq,m,L}(\rho^{(2)}) \tag{A.26}$$

$$+ f^{eq,m,BL}(\rho^{(0)}, \boldsymbol{v}^{(2)}) \tag{A.27}$$

$$+ f^{eq,m,Q}(\rho^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(1)}) \tag{A.28}$$

$$- f^{m,(2)}) + F^{m,(2)} \tag{A.29}$$

Rewrite above equation in terms of $f^{m,(2)}$ by taking advantage of Eq. A.24 in bi-linear part and $\partial_t f^{m,(0)} = \omega^m \partial_t \rho^{(0)} = 0$ and $F^{m,(2)} = 0$,

$$\begin{aligned} f^{m,(2)} &= f^{eq,m,L}(\rho^{(2)}) + f^{eq,m,Q}(\rho^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(1)}) \\ &\quad - \frac{1}{\lambda_\nu} \boldsymbol{u}^m \cdot \nabla f^{m,(1)} \end{aligned} \tag{A.30}$$

Since $f^{m,(1)}$ is already known, it enables us to calculate second order tensor in momentum transport equation Eq. A.21. Deriving second order tensor for index $\alpha, \beta$,

$$\begin{aligned} \langle 1, u_\alpha u_\beta f^{(2)} \rangle &= \sum_m u_\alpha^m u_\beta^m f^{eq,m,L}(\rho^{(2)}) + \sum_m u_\alpha^m u_\beta^m f^{eq,m,Q}(\rho^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(1)}) \\ &\quad - \frac{\rho^{(0)}}{\lambda_\nu c_s^2} \nabla \cdot \sum_m \omega^m u_\alpha^m u_\beta^m \boldsymbol{u}^m (\boldsymbol{u}^m \cdot \boldsymbol{v}^{(1)}) \end{aligned}$$

Using Eq. A.10c and Eq. A.12c together gives,

$$\begin{aligned}\langle 1, u_\alpha u_\beta f^{(2)}\rangle &= c_s^2\rho^{(2)} + \rho^{(0)}(\boldsymbol{v}^{(1)}\otimes\boldsymbol{v}^{(1)})_{\alpha\beta} \\ &- \frac{\rho^{(0)}}{\lambda_\nu c_s^2}\sum_\gamma \partial_{x_\gamma}\Big(\sum_m \omega^m u_\alpha^m u_\beta^m u_\gamma^m \sum_\delta u_\delta^m v_\delta^{(1)}\Big) \\ &= c_s^2\rho^{(2)} + \rho^{(0)}(\boldsymbol{v}^{(1)}\otimes\boldsymbol{v}^{(1)})_{\alpha\beta} \\ &- \frac{\rho^{(0)}}{\lambda_\nu c_s^2}\sum_\gamma \partial_{x_\gamma}\sum_\delta v_\delta^{(1)}\sum_m \omega^m u_\alpha^m u_\beta^m u_\gamma^m u_\delta^m \\ &= c_s^2\rho^{(2)} + \rho^{(0)}(\boldsymbol{v}^{(1)}\otimes\boldsymbol{v}^{(1)})_{\alpha\beta} \\ &- \frac{\rho^{(0)}}{\lambda_\nu c_s^2}\sum_\gamma \partial_{x_\gamma}\sum_\delta v_\delta^{(1)}(\kappa c_s^4(\delta_{\alpha,\beta}\delta_{\gamma,\delta}+\delta_{\alpha,\gamma}\delta_{\beta,\delta}+\delta_{\alpha,\delta}\delta_{\beta,\gamma}))\end{aligned} \tag{A.31}$$

Using the property of Kronecker delta i.e $\delta_{ij} = 1$ only when $i = j$ otherwise zero.

$$\begin{aligned}\langle 1, u_\alpha u_\beta f^{(2)}\rangle &= c_s^2\rho^{(2)} + \rho^{(0)}(\boldsymbol{v}^{(1)}\otimes\boldsymbol{v}^{(1)})_{\alpha\beta} \\ &- \frac{\rho^{(0)}\kappa c_s^2}{\lambda_\nu}(\partial_{x_\alpha}v_\alpha^{(1)} + \partial_{x_\alpha}v_\beta^{(1)} + \partial_{x_\beta}v_\alpha^{(1)})\end{aligned} \tag{A.32}$$

Now, lets generalize above equation for second order tensor as

$$\begin{aligned}\langle 1, \boldsymbol{u}\otimes\boldsymbol{u}f^{(2)}\rangle &= c_s^2\rho^{(2)}\boldsymbol{I} + \rho^{(0)}(\boldsymbol{v}^{(1)}\otimes\boldsymbol{v}^{(1)}) \\ &- \frac{\rho^{(0)}\kappa c_s^2}{\lambda_\nu}(\nabla\cdot\boldsymbol{v}^{(1)}\boldsymbol{I} + \nabla(\boldsymbol{v}^{(1)})^T + \nabla(\boldsymbol{v}^{(1)}))\end{aligned} \tag{A.33}$$

where $\boldsymbol{I}$ is the identity matrix.Due to continuity condition $\nabla\cdot\boldsymbol{v}^{(1)} = 0$. Substituting Eq. A.33 in Eq. A.21 and replacing $\mathbf{p}^{(1)} = \rho^{(0)}\boldsymbol{v}^{(1)}$ results,

$$\begin{aligned}\partial_t(\rho^{(0)}\boldsymbol{v}^{(1)}) + \nabla\cdot\Big(c_s^2\rho^{(2)}\boldsymbol{I} + \rho^{(0)}(\boldsymbol{v}^{(1)}\otimes\boldsymbol{v}^{(1)}) \\ - \frac{\rho^{(0)}\kappa c_s^2}{\lambda_\nu}(\nabla(\boldsymbol{v}^{(1)})^T + \nabla(\boldsymbol{v}^{(1)}))\Big) = \boldsymbol{h}^{(3)}.\end{aligned} \tag{A.34}$$

Divide above equation with $\rho^{(0)}$. Also, setting pressure $P^{(2)} = c_s^2\rho^{(2)}$ and kinematic viscosity

$$\nu = \frac{\kappa c_s^2}{\lambda^\nu} \tag{A.35}$$

and the force term

$$\boldsymbol{h}^{(3)} = \sum_m \omega^m \boldsymbol{u}^m \boldsymbol{u}^m \cdot \boldsymbol{G}^3 = c_s^2 \boldsymbol{G}^{(3)} \tag{A.36}$$

to recover incompressible momentum transport equation of the form,

$$\partial_t \boldsymbol{v}^{(1)} + \nabla \cdot (\boldsymbol{v}^{(1)} \otimes \boldsymbol{v}^{(1)}) = -\frac{\nabla P^{(2)}}{\rho^{(0)}} + \nu \nabla \cdot (\nabla (\boldsymbol{v}^{(1)})^T + \nabla (\boldsymbol{v}^{(1)})) + \frac{c_s^2}{\rho^{(0)}} \boldsymbol{G}^{(3)}. \tag{A.37}$$

Comparing this with the external body force per unit volume in Eq. 2.56

$$\boldsymbol{F} = \boldsymbol{g} + \frac{\rho^e}{\rho^{(0)}} \boldsymbol{E} \tag{A.38}$$

gives

$$\boldsymbol{G}^{(3)} = \frac{\rho^{(0)}}{c_s^2} \boldsymbol{F}^{(3)} \tag{A.39}$$

Thus, the external body force term $F^m$ in Boltzmann equation Eq. A.1 is

$$F^{m,(3)} = \frac{1}{c_s^2} \omega^m \boldsymbol{u}^m \cdot (\rho^{(0)} \boldsymbol{F}^{(3)}). \tag{A.40}$$

This relation is similar to one proposed in [41]. However, there are other methods proposed to add forcing term to the Boltzmann equation [20, 28]. Taking moments of this body force term $F^m$ gives [20]

$$\sum_m F^m = 0, \quad \sum_m \boldsymbol{u}^m F^m = \rho^{(0)} \boldsymbol{F}, \quad \sum_m \boldsymbol{u}^m \boldsymbol{u}^m F^m = \rho^{(0)} (\boldsymbol{F}\boldsymbol{v} + \boldsymbol{v}\boldsymbol{F}) \tag{A.41}$$

Upper analysis satisfies the incompressible Navier-Stokes equations with velocity described by first order $\epsilon$ and pressure in the second order $\epsilon^2$ of density. From Eq. A.24, the velocity can be written in terms of the odd components

$$\boldsymbol{v} = \epsilon \boldsymbol{v}^{(1)} + \epsilon^3 \boldsymbol{v}^{(3)} + \epsilon^5 \boldsymbol{v}^{(5)} + \dots \tag{A.42a}$$

and density can be written in terms of the even components

$$\rho = \rho^{(0)} + \epsilon^2 \rho^{(2)} + \epsilon^4 \rho^{(4)} + \dots \tag{A.42b}$$

Thus, the velocity $\boldsymbol{v}^{(1)}$ and the pressure $P^{(2)}$ can be expressed in terms of $\epsilon$ as

$$\boldsymbol{v}^{(1)} = \frac{1}{\epsilon}\boldsymbol{v} + \mathcal{O}(\epsilon^2) \tag{A.43}$$

$$P^{(2)} = \frac{c_s^2}{\epsilon^2}(\rho - \rho^{(0)}) + \mathcal{O}(\epsilon^2) \tag{A.44}$$

which implies that this velocity and pressure yields at least second order accurate approximation of the incompressible Navier-Stokes equations.

#### A.1.1.1 Time discretization

As mentioned earlier, the semi-discrete Boltzmann equation Eq. A.1 in App. A.1 was discrete in velocity space but still continuous in $\boldsymbol{x}$ and $t$. Therefore, to obtain the fully discrete Boltzmann equation i.e. the lattice Boltzmann equation, Eq. A.1 must be approximated in $\boldsymbol{x}$ and $t$. Integrating Eq. A.1 along a characteristic for a time interval from 0 to $\delta t$ [21, 34] results in

$$\begin{aligned}&\int_0^{\delta t} \partial_t f^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) + \boldsymbol{u}^m \cdot \nabla f^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) ds \\ &= \int_0^{\delta t} \Omega^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) ds + \int_0^{\delta t} F^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) ds.\end{aligned} \tag{A.45}$$

The integration of left hand side gives exactly the difference of the function values at either end of the characteristics

$$\begin{aligned}&\int_0^{\delta t} \partial_t f^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) + \boldsymbol{u}^m \cdot \nabla f^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) ds \\ &\qquad = \int_0^{\delta t} \partial_s f^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) ds \\ &\qquad = f(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t + \delta t) - f(\boldsymbol{x}, t).\end{aligned} \tag{A.46}$$

The first integrand term on the right hand side of Eq. A.45 represents the collision term. Approximating this integral by the forward Euler would result in first order accuracy in $\delta t$ which causes numerical instability [20]. Therefore, to improve the stability and accuracy of the scheme, the collision term is integrated by trapezium rule

$$\int_0^{\delta t} \Omega^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) ds \approx \frac{\delta t}{2}\Big(\Omega(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t + \delta t) + \Omega(\boldsymbol{x}, t)\Big) \tag{A.47}$$

which gives second order accuracy in $\delta t$. To be consistent with collision integral, the second integrand term in Eq. A.45 representing the body force term $F^m$ is also approximated by trapezium rule. Thus, Eq. A.45 with MRT collision operator can be written as

$$\begin{aligned} & f^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) - f^m(\boldsymbol{x},t) \\ & = \frac{\delta t \boldsymbol{A}}{2}\Big[(f^{eq,m}(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) - f^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t)) \\ & + (f^{eq,m}(\boldsymbol{x},t) - f^m(\boldsymbol{x},t))\Big] \\ & + \frac{\delta t}{2}\Big[F^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) + F^m(\boldsymbol{x},t)\Big]. \end{aligned} \tag{A.48}$$

Clearly this is an implicit equation, since it involves the equilibrium function at a later time point $t+\delta t$ (on the right hand side of the equation). As the dependence of $f^{eq,m}$ is typically nonlinear with respect to $f^m$ this equation system is not easy to solve. To circumvent this problem, a variable transformation technique is proposed [34] to convert this into explicit equation by introducing a new variables $\bar{f}^m$ as

$$\bar{f}^m(\boldsymbol{x},t) = f^m(\boldsymbol{x},t) + \frac{\delta t \boldsymbol{A}}{2}\Big(f^m(\boldsymbol{x},t) - f^{eq,m}(\boldsymbol{x},t)\Big) - \frac{\delta t}{2}F^m(\boldsymbol{x},t) \tag{A.49}$$

Thus, the second-order fully discrete LBE with a body force in $\bar{f}^m$ is

$$\begin{aligned} & \bar{f}^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) - \bar{f}^m(\boldsymbol{x},t) \\ & = \delta t \boldsymbol{A}\left(\boldsymbol{I} + \frac{\delta t \boldsymbol{A}}{2}\right)^{-1}\Big(f^{eq,m}(\boldsymbol{x},t) - \bar{f}(\boldsymbol{x},t)\Big) \\ & + \delta t\left(\boldsymbol{I} + \frac{\delta t \boldsymbol{A}}{2}\right)^{-1} F^m(\boldsymbol{x},t). \end{aligned} \tag{A.50}$$

Above equation can be rewritten as

$$\begin{aligned} \bar{f}(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) = \bar{f}(\boldsymbol{x},t) + \bar{\boldsymbol{A}}(f^{eq,m}(\boldsymbol{x},t) - \bar{f}(\boldsymbol{x},t)) \\ + \delta t\left(\boldsymbol{I} - \frac{\bar{\boldsymbol{A}}}{2}\right) F^m(\boldsymbol{x},t) \end{aligned} \tag{A.51}$$

with new linear collision operator,

$$\bar{\boldsymbol{A}} = \delta t \boldsymbol{A}\left(1 + \frac{\delta t \boldsymbol{A}}{2}\right)^{-1} \tag{A.52}$$

Above new linear collision operator $\bar{\boldsymbol{A}}$ can be simplified into

$$\bar{\boldsymbol{A}} = \boldsymbol{M}^{-1}\bar{\boldsymbol{\Lambda}}\boldsymbol{M}. \tag{A.53}$$

by modifying the relaxation parameters in the new relaxation matrix $\bar{\boldsymbol{\Lambda}}$ as

$$\bar{\lambda} = \frac{\delta t\lambda}{\left(1 + \frac{\delta t\lambda}{2}\right)}. \tag{A.54}$$

Recalling $\lambda^{\nu} = \frac{c_s^2}{\nu}$, the above equation becomes

$$\bar{\lambda}_{\nu} = \frac{1}{\frac{\nu}{\delta t c_s^2} + \frac{1}{2}} \implies \nu = c_s^2 \delta t \left(\frac{1}{\bar{\lambda}_{\nu}} - \frac{1}{2}\right) \tag{A.55}$$

which is same as Eq. 3.22. Similarly, the relaxation parameter with bulk viscosity can written as

$$\zeta = \frac{(5 - 9c_s^2)\delta t}{9}\left(\frac{1}{\bar{\lambda}_{\zeta}} - \frac{1}{2}\right) \tag{A.56}$$

Thus, as long as relaxation parameter $\bar{\lambda}_{\nu}$ and $\bar{\lambda}_{\zeta}$ are chosen according to Eq. A.55 and Eq. A.56, the LBM scheme is second order in time. With BGK collision model, the second order LBE can be written as

$$\begin{aligned}\bar{f}(\boldsymbol{x} + \boldsymbol{u}^m\delta t, t + \delta t) = \bar{f}(\boldsymbol{x}, t) + \bar{\lambda}_{\nu}(f^{eq,m}(\boldsymbol{x}, t) - \bar{f}(\boldsymbol{x}, t)) \\ + \delta t\left(1 - \frac{\bar{\lambda}_{\nu}}{2}\right) F^m(\boldsymbol{x}, t)\end{aligned} \tag{A.57}$$

Notice that now the lattice Boltzmann algorithm is not solved for $f^m$ but on transformed variables $\bar{f}^m$ defined by Eq. A.49. Additionally, the macroscopic quantities density $\rho$ and velocity $\boldsymbol{v}$ in equilibrium function $f^{eq,m}$ can be computed directly from $\bar{f}^m$ assuming that the mass and momentum are conserved in collision term [20]. Taking zeroth moments of $\bar{f}$ Eq. A.49 gives the macroscopic density

$$\rho = \sum_m \bar{f}^m = \sum_m f^m \tag{A.58}$$

and first moments gives the momentum [34]

$$\mathbf{p} = \rho^0 \boldsymbol{v} = \sum_m \boldsymbol{u}^m f = \sum_m \boldsymbol{u}^m \bar{f} + \frac{\delta t}{2}\rho^{(0)}\boldsymbol{F}. \tag{A.59}$$

The other moments which are not conserved in collision term are affected by the replacement of $f^m$ by $\bar{f}^m$. For example, the deviatoric (non-equilibrium) stress $\mathbf{\Pi}^{(1)}$ is [34]

$$\begin{aligned}\mathbf{\Pi}^{(1)} = \mathbf{\Pi} - \mathbf{\Pi}^{(0)} &= \frac{\bar{\mathbf{\Pi}} - \mathbf{\Pi}^{(0)} + \frac{\delta t}{2}\rho^{(0)}[\boldsymbol{F}\boldsymbol{u} + \boldsymbol{u}\boldsymbol{F}]}{1 + \frac{\delta t \lambda}{2}} \\ &= \left(1 - \frac{\bar{\lambda}}{2}\right)\left(\bar{\mathbf{\Pi}} - \mathbf{\Pi}^{(0)} + \frac{\delta t}{2}\rho^{(0)}[\boldsymbol{F}\boldsymbol{u} + \boldsymbol{u}\boldsymbol{F}]\right) \quad \text{(A.60)}\end{aligned}$$

where

$$\bar{\mathbf{\Pi}} = \sum_m \boldsymbol{u}^m \boldsymbol{u}^m \bar{f}^m. \quad \text{(A.61)}$$

Note that addition of body force in momentum and momentum flux calculation is due to the fact that integrand of body force term is approximated by trapezoidal rule. However, for time independent force $\boldsymbol{F}$ or for small $\delta t$, this addition of force term in Eq. A.59 and Eq. A.60 can be avoided if integrand of body force term is approximated by forward Euler. Thus reduces the global scheme order to first order in time when body forces are considered in LBE. Thus, the transformed variables $\bar{f}$ will be devoid of body force term as

$$\bar{f}^m(\boldsymbol{x}, t) = f^m(\boldsymbol{x}, t) + \frac{\delta t \lambda_\nu}{2}\left(f^m(\boldsymbol{x}, t) - f^{eq,m}(\boldsymbol{x}, t)\right) \quad \text{(A.62)}$$

and the fully discrete LBE with a body force term can be written as

$$\begin{aligned}\bar{f}(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t + \delta t) = \bar{f}(\boldsymbol{x}, t) + \bar{\lambda}(f^{eq,m}(\boldsymbol{x}, t) - \bar{f}(\boldsymbol{x}, t)) \\ + \delta t F^m(\boldsymbol{x}, t). \quad \text{(A.63)}\end{aligned}$$

Thus, the macroscopic moments density, momentum and deviatoric stress are

$$\rho = \sum_m \bar{f}^m \quad \text{(A.64)}$$

$$\mathbf{p} = \rho^0 \boldsymbol{v} = \sum_m \boldsymbol{u}^m \bar{f} \quad \text{(A.65)}$$

$$\mathbf{\Pi}^{(1)} = \mathbf{\Pi} - \mathbf{\Pi}^{(0)} = \left(1 - \frac{\bar{\lambda}}{2}\right)\left(\bar{\mathbf{\Pi}} - \mathbf{\Pi}^{(0)}\right) \quad \text{(A.66)}$$

The forward Euler is used to approximate the body force term because of the limitation of the LBM algorithm implementation in Musubi. The current implementation adds the source term after the collision step resulting

in compute kernel being independent of the source term. Therefore, the contribution of force term by the second-order approximation of the force term in the momentum calculation is not accessible in the compute kernel.

### A.1.2 Multicomponent Lattice Boltzmann Method for Nonideal Liquid Mixtures

In this section, the semi-discrete Boltzmann Eq. 3.39 for species is analyzed by means of the diffusive asymptotic analysis. Like App. A.1.1, the macroscopic equations: the incompressible Navier-Stokes for mixture, the mass conservation of species and the Maxwell-Stefan equation for nonideal mixture for species diffusion transport presented in Section 2.2 are recovered. In the limit of small Knudsen and Mach number the desired macroscopic equations are solved with second order accuracy in space and first order accuracy in time.

Lets recall the diffusive scaling for time step and grid width

$$\widetilde{\delta x} = \epsilon \delta x \tag{A.67a}$$

$$\widetilde{\delta t} = \epsilon^2 \delta t, \tag{A.67b}$$

where $\epsilon$ may be considered as the Knudsen number $Kn$ or the Mach number $M$. Inserting this scaling in equation Eq. 3.39 with external force and replacing the partial derivatives by its asymptotic counterparts, the following asymptotic relation for the FDVM can be derived

$$\epsilon^2 \partial_t f_k^m + \epsilon \boldsymbol{u}^m \cdot \nabla f_k^m = \lambda_k \left( f_k^{eq,m} - f_k^m \right) + d_k^m. \tag{A.68}$$

Similar to the single component case, each species probability density function $f_k^m$, external force $d_k^m$, macroscopic variables like density $\rho_k$, momentum $\mathbf{p}_k$ and velocity $\mathbf{v}_k$ are expanded as an asymptotic power series in terms of $\epsilon$

$$f_k^m = \sum_{i=1}^{\infty} \epsilon^i f_k^{m,(i)} \tag{A.69a}$$

$$d_k^m = \sum_{i=1}^{\infty} \epsilon^i d_k^{m,(i)} \tag{A.69b}$$

$$\rho_k = \sum_{i=1}^{\infty} \epsilon^i \rho_k^{(i)} \tag{A.69c}$$

$$\mathbf{p}_k = \sum_{i=1}^{\infty} \epsilon^i \mathbf{p}_k^{(i)} \tag{A.69d}$$

$$\mathbf{v}_k = \sum_{i=1}^{\infty} \epsilon^i \mathbf{v}_k^{(i)}. \tag{A.69e}$$

Species momentum $\mathbf{p}_k$ can be rewritten as a convolution sum of $\rho_k^{(i)}$ and $\mathbf{v}_k^{(i)}$ by inserting expansion for species density $\rho_k$ and species velocity $\mathbf{v}_k$

$$\mathbf{p}_k = \rho_k \mathbf{v}_k = \sum_{i=1}^{\infty} \epsilon^i \sum_{\substack{j,l \in \mathbb{N}_{\geq 0} \\ j+l=i}} \rho_k^{(j)} \mathbf{v}_k^{(l)}. \tag{A.69f}$$

For convenience, lets split the equilibrium probability density function Eq. 3.42 into linear, bi-linear and quadratic parts centered around species equilibrium velocity $\mathbf{v}_k^{eq}$ and mixture velocity $\boldsymbol{v}$, as

$$\begin{aligned} &f_k^{eq,m}(\rho_k, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_N) \\ &= f_k^{eq,m,L}(\rho_k) + f_k^{eq,m,BL}(\rho_k, \mathbf{v}_k^{eq}(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N)) \\ &+ f_k^{eq,m,Q}(\rho_k, \boldsymbol{v}(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N), \boldsymbol{v}(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N)), \end{aligned} \tag{A.70}$$

where,

$$f_k^{eq,m,L}(\rho_k) = \omega^m \rho_k s_m^k \tag{A.71a}$$

$$f_k^{eq,m,BL}(\rho_k, \mathbf{v}_k^{eq}) = \omega^m \rho_k \frac{1}{c_s^2} \left(\boldsymbol{u}^m \cdot \mathbf{v}_k^{eq}\right) \tag{A.71b}$$

$$f_k^{eq,m,Q}(\rho_k, \boldsymbol{v}, \boldsymbol{v}) = \omega^m \rho_k \left( \frac{1}{2c_s^4} \left(\boldsymbol{u}^m \cdot \boldsymbol{v}\right) \left(\boldsymbol{u}^m \cdot \boldsymbol{v}\right) - \frac{1}{2c_s^2} \left(\boldsymbol{v} \cdot \boldsymbol{v}\right) \right). \tag{A.71c}$$

This splitting allows us to modify each part of the equilibrium separately. Taking advantage of isotropic condition Eq. 3.13, the following for linear term $f_k^{eq,m,L}$ can be obtained

$$\begin{aligned} \langle 1, f_k^{eq,m,L} \rangle &= \rho_k \\ \langle 1, \boldsymbol{u}^m f_k^{eq,m,L} \rangle &= 0 \\ \langle 1, u_\alpha^m u_\beta^m f_k^{eq,m,L} \rangle &= \rho_k \sum_m \omega^m s_m^k u_\alpha^m u_\beta^m \\ &= \rho_k \sum_{m \neq 0} \omega^m \phi_k u_\alpha^m u_\beta^m = \phi_k c_s^2 \delta_{\alpha,\beta} \rho_k \end{aligned} \tag{A.72}$$

Similarly, the first three moments of the bi-linear equilibrium part $f_k^{eq,m,BL}$ and the quadratic equilibrium part $f_k^{eq,m,Q}$ can be derived as

$$\begin{aligned}
\langle 1, f_k^{eq,m,BL}\rangle &= 0 \\
\langle 1, \boldsymbol{u}^m f_k^{eq,m,BL}\rangle &= \rho_k \mathbf{v}_k^{eq} \\
\langle 1, u_\alpha^m u_\beta^m f_k^{eq,m,BL}\rangle &= 0 \qquad \text{(A.73)} \\
\langle 1, f_k^{eq,m,Q}\rangle &= 0 \\
\langle 1, \boldsymbol{u}^m f_k^{eq,m,Q}\rangle &= 0 \\
\langle 1, u_\alpha^m u_\beta^m f_k^{eq,m,Q}\rangle &= (\rho_k \boldsymbol{v}\otimes\boldsymbol{v})_{\alpha,\beta}\,. \qquad \text{(A.74)}
\end{aligned}$$

Applying the asymptotic expansions for $f_k^m$ Eq. A.69a in the BGK semi-discrete Boltzmann equation Eq. A.68 in the diffusive limit and separating the scales in $\epsilon$, the following expression is obtained in the order $\epsilon^{i+2}$ for $i \geq -2$:

$$\begin{aligned}
\partial_t f_k^{m,(i)} + \boldsymbol{u}^m \cdot \nabla f_k^{m,(i+1)} &= \lambda_k f_k^{eq,m,L}(\rho_k^{(i+2)}) \\
&+ \lambda_k \sum_{\substack{j,l\geq 0 \\ j+l=i+2}} f_k^{eq,m,BL}(\rho_k^{(j)}, \mathbf{v}_k^{eq}(\boldsymbol{v}_1^{(l)}, \cdots \boldsymbol{v}_N^{(l)})) \\
&+ \lambda_k \sum_{\substack{j,l,p\geq 0 \\ j+l+p=i+2}} f_k^{eq,m,Q}(\rho_k^{(j)}, \boldsymbol{v}(\boldsymbol{v}_1^{(l)}, \cdots \boldsymbol{v}_N^{(l)}), \boldsymbol{v}(\boldsymbol{v}_1^{(p)}, \cdots, \boldsymbol{v}_N^{(p)})) \\
&- \lambda_k f_k^{m,(i+2)} + d_k^{m,(i+2)}. \qquad \text{(A.75)}
\end{aligned}$$

The previous equation holds true for all $i \in \mathbb{Z}$ by setting

$$f_k^{m,i} = 0 \qquad \text{(A.76)}$$

for $i < 0$. Notice that Eq. A.75 is the multicomponent analogue of single component Eq. A.13. Similar to the single component density Eq. A.14 and momentum Eq. A.16, the density and the momentum for species $k$ can be obtained from Eq. A.75 by applying $\rho_k = \sum_m f_k^m$ and $\mathbf{p}_k = \sum_m \boldsymbol{u}^m f_k^m$:

$$\partial_t \rho_k^{(i)} + \nabla \cdot \mathbf{p}_k^{i+1} = \partial_t \rho_k^{(i)} + \nabla \cdot \sum_{\substack{j,l\geq 0 \\ j+l=i+1}} \rho_k^{(j)} \mathbf{v}_k^{(l)} = g_k^{(i+2)} \qquad \text{(A.77)}$$

and

$$\begin{aligned}
\partial_t \mathbf{p}_k^{(i)} + \nabla \cdot \langle 1, \boldsymbol{u}^m \otimes \boldsymbol{u}^m f_k^{m,(i+1)}\rangle &= \lambda_k \sum_{\substack{j,l\geq 0 \\ j+l=i+2}} \rho_k^{(j)} \left(\mathbf{v}_k^{eq,(l)} - \mathbf{v}_k^{(l)}\right) \\
&+ \boldsymbol{h}_k^{(i+2)}. \qquad \text{(A.78)}
\end{aligned}$$

respectively, where

$$g_k^{(i)} = \sum_m d_k^{m,(i)} \tag{A.79a}$$

$$\boldsymbol{h}_k^{(i)} = \sum_m \boldsymbol{u}^m d_k^{m,(i)}. \tag{A.79b}$$

From [41], following relation holds true

$$d_k^{m,(i)} = \begin{cases} \omega^m \boldsymbol{u}^m \cdot \boldsymbol{G}_k^{(i)} & \text{, if } i = 1,3 \\ 0 & \text{, else} \end{cases}. \tag{A.80}$$

i.e. the external force $d_k^m$ is an odd function. Furthermore, lets make use of the following notation for the equilibrium center velocity built with the $l$-th part of our macroscopic asymptotic expansion Eq. 3.40 for nonideal liquid mixture $\forall j, l \geq 0,\ j + l = i + 2$

$$\begin{aligned} \mathbf{v}_k^{eq,(l)} &= \mathbf{v}_k^{eq}(\boldsymbol{v}_1^{(l)}, \cdots, \boldsymbol{v}_N^{(l)}) \\ &= \mathbf{v}_k^{(l)} + \frac{1}{\rho_k^{(j)}} \sum_\eta \Gamma_{k,\eta}^{-1} \rho_\eta^{(j)} \phi_\eta \sum_\zeta \chi_\zeta^{(j)} \frac{\mathcal{D}}{\mathcal{D}_{\zeta,\eta}} \left( \boldsymbol{v}_\zeta^{(l)} - \boldsymbol{v}_\eta^{(l)} \right). \end{aligned} \tag{A.81}$$

#### A.1.2.1 Species mass transport

The macroscopic species transport Eq. 2.14 can be obtained by setting $i = 0$ in density evolution Eq. A.77 and using Eq. A.24

$$\partial_t \rho_k^{(0)} + \nabla \cdot \left( \rho_k^{(0)} \mathbf{v}_k^{(1)} \right) = 0. \tag{A.82}$$

Note that the finite discrete velocity method conserves species mass as the relation

$$g_k^{(i)} = 0 \qquad \forall i \in \{0, \ldots, \infty\} \tag{A.83}$$

holds. As presented in Section 2.2, the momentum or mass flux density $\mathbf{p}_k$ is related to mass diffusion flux by $\mathbf{p}_k = \boldsymbol{j}_k + \rho_k \mathbf{v}$. Additionally, the phenomenological relation for the diffusion flux and the species concentration is required to complete the species transport and it is given by the Maxwell-Stefan diffusion equations. The Maxwell-Stefan equation with external forces for nonideal liquid mixture is given as

$$\sum_{j=1}^{n} \Gamma_{k,j} \nabla \chi_j - \boldsymbol{F}_k = \sum_{l=1}^{n} \frac{(\chi_k \boldsymbol{J}_l - \chi_l \boldsymbol{J}_k)}{c_t \mathcal{D}_{k,l}} \tag{A.84}$$

and it can be recovered by setting $i = -1$ Eq. A.78 which results

$$\nabla \cdot \langle 1, \boldsymbol{u}^m \otimes \boldsymbol{u}^m f_k^{m,(0)} \rangle = \lambda_k \left( \sum_{\substack{j,l \geq 0 \\ j+l=1}} \rho_k^{(j)} \left( \mathbf{v}_k^{eq,()} - \mathbf{v}_k^{(l)} \right) \right) + \boldsymbol{h}_k^{(1)}. \tag{A.85}$$

$f_k^{m,(0)}$ can be obtained by setting $i = -2$ in Eq. A.75 and setting macroscopic moments with negative index to zero gives

$$f_k^{m,(0)} = f_k^{eq,m,L}(\rho_k^{(0)}) = \omega^m s_k^m \rho_k^{(0)} \tag{A.86}$$

$$\Rightarrow \mathbf{v}_k^{(0)} = 0 \tag{A.87}$$

$$\Rightarrow \boldsymbol{v}^{(0)} = 0 \tag{A.88}$$

Combining Eq. A.86, Eq. A.85 and Eq. A.72 gives

$$\begin{aligned} c_s^2 \nabla \phi_k \rho_k^{(0)} &= \lambda_k \left( \rho_k^{(0)} \left( \mathbf{v}_k^{eq,(1)} - \mathbf{v}_k^{(1)} \right) \right) + \boldsymbol{h}_k^{(1)} \\ \implies \nabla \phi_k \rho_k^{(0)} &= \frac{\lambda_k}{c_s^2} \left( \rho_k^{(0)} \left( \mathbf{v}_k^{eq,(1)} - \mathbf{v}_k^{(1)} \right) \right) + \frac{1}{c_s^2} \boldsymbol{h}_k^{(1)}. \end{aligned} \tag{A.89}$$

Now, lets substitute the definition of relaxation parameter Eq. 3.47a i.e $\lambda_k = \frac{K}{\rho \mathcal{D}}$ and the speed of sound for liquid mixture Eq. 3.48 i.e $c_s^2 = \frac{K}{\rho}$ and the equilibrium velocity Eq. A.81 with $j = 0$ and $l = 1$ to obtain

$$\nabla \phi_k \rho_k^{(0)} = \sum_\eta \Gamma_{k,\eta}^{-1} \rho_\eta^{(0)} \phi_\eta \sum_\zeta \chi_\zeta^{(0)} \frac{1}{\mathcal{D}_{\eta,\zeta}} \left( \boldsymbol{v}_\zeta^{(1)} - \boldsymbol{v}_\eta^{(1)} \right) + \frac{1}{c_s^2} \boldsymbol{h}_k^{(1)} \tag{A.90}$$

This equation shows that the free parameter $\mathcal{D}$ has no influence on the species mass transport. Now, taking advantage of choice of parameter $\phi_k = \frac{\min_\alpha M_\alpha}{M_k} \leq 1$ and the relation of molar density $c_k = \frac{\rho_k}{M_k}$ results in

$$\begin{aligned} min_\alpha M_\alpha \nabla c_k^{(0)} &= \sum_\eta \Gamma_{k,\eta}^{-1} c_\eta^{(0)} min_\alpha M_\alpha \sum_\zeta \chi_\zeta^{(0)} \frac{1}{\mathcal{D}_{\eta,\zeta}} \left( \boldsymbol{v}_\zeta^{(1)} - \boldsymbol{v}_\eta^{(1)} \right) + \frac{1}{c_s^2} \boldsymbol{h}_k^{(1)} \\ \implies \nabla c_k^{(0)} &= \sum_\eta \Gamma_{k,\eta}^{-1} c_\eta^{(0)} \sum_\zeta \chi_\zeta^{(0)} \frac{1}{\mathcal{D}_{\eta,\zeta}} \left( \boldsymbol{v}_\zeta^{(1)} - \boldsymbol{v}_\eta^{(1)} \right) \\ &\quad + \frac{1}{min_\alpha M_\alpha c_s^2} \boldsymbol{h}_k^{(1)}. \end{aligned} \tag{A.91}$$

Summing up over all species gives

$$\begin{aligned}\sum_k \nabla c_k^{(0)} &= \nabla c_t^{(0)} \\ &= \sum_k \sum_\eta \Gamma_{k,\eta}^{-1} c_\eta^{(0)} \sum_\zeta \chi_\zeta^{(0)} \frac{1}{\mathcal{D}_{\eta,\zeta}} \left( \boldsymbol{v}_\zeta^{(1)} - \boldsymbol{v}_\eta^{(1)} \right) \\ &+ \frac{1}{min_\alpha M_\alpha c_s^2} \sum_k \boldsymbol{h}_k^{(1)} = 0 \iff \sum_k \boldsymbol{h}_k = 0. \end{aligned} \tag{A.92}$$

This shows that the zeroth order total mole density $c_t^{(0)}$ is constant throughout the whole domain only in case of balanced external forces. Using the definition of molar diffusion flux of first order $\boldsymbol{J}_k^{(1)} = c_k^{(0)}(\mathbf{v}_k^{(1)} - \boldsymbol{w}^{(1)})$ in Eq. A.91 and dividing both sides by $c_t^{(0)}$ and replacing $\chi_k = \frac{c_k}{c_t}$ gives the Maxwell-Stefan equation for nonideal liquid mixture

$$\nabla \chi_k^{(0)} = \sum_\eta \Gamma_{k,\eta}^{-1} \sum_\zeta \frac{1}{c_t \mathcal{D}_{\eta,\zeta}} \left( \chi_\eta^{(0)} \boldsymbol{J}_\zeta^{(1)} - \chi_\zeta^{(0)} \boldsymbol{J}_\eta^{(1)} \right) + \frac{1}{min_\alpha M_\alpha c_t^{(0)} c_s^2} \boldsymbol{h}_k^{(1)}$$

$$\begin{aligned} \implies \sum_\eta \Gamma_{k,\eta} \nabla \chi_\eta^{(0)} &= \sum_\zeta \frac{1}{c_t \mathcal{D}_{k,\zeta}} \left( \chi_\eta^{(0)} \boldsymbol{J}_\zeta^{(1)} - \chi_\zeta^{(0)} \boldsymbol{J}_\eta^{(1)} \right) \\ &+ \frac{1}{min_\alpha M_\alpha c_t^{(0)} c_s^2} \sum_\eta \Gamma_{k,\eta} \boldsymbol{h}_\eta^{(1)} \end{aligned} \tag{A.93}$$

This equation is similar to the Maxwell-Stefan equation for nonideal liquid mixture Eq. A.84 except for the forcing term with $\boldsymbol{h}_\eta^{(1)}$. Comparing above equation with Eq. A.84 and using Eq. A.79b, Eq. A.80 yields the following identification of the forcing term

$$\begin{aligned} \frac{1}{min_\alpha M_\alpha c_t^{(0)} c_s^2} \sum_\eta \Gamma_{k,\eta} \boldsymbol{h}_\eta^{(1)} &= \frac{1}{min_\alpha M_\alpha c_t^{(0)} c_s^2} \sum_\eta \Gamma_{k,\eta} \sum_m \boldsymbol{u}^m d_\eta^{(1)} \\ &= \frac{1}{min_\alpha M_\alpha c_t^{(0)} c_s^2} \sum_\eta \Gamma_{k,\eta} \sum_m \omega^m \boldsymbol{u}^m \left( \boldsymbol{u}^m \cdot \boldsymbol{G}_\eta^{(1)} \right) \\ &= \frac{1}{min_\alpha M_\alpha c_t^{(0)}} \sum_\eta \Gamma_{k,\eta} \boldsymbol{G}_\eta^{(1)} = \boldsymbol{F}_k \\ \implies \boldsymbol{G}_k^{(1)} &= min_\alpha M_\alpha \sum_\eta \Gamma_{k,\eta}^{-1} c_t^{(0)} \boldsymbol{F}_\eta \end{aligned} \tag{A.94}$$

Thus, the first order external force term $d_k^{m,(1)}$ corresponds to the external diffusive force $\boldsymbol{F}_k$ of the Maxwell Stefan equation for nonideal liquid mixture as

$$d_k^{m,(1)} = min_\alpha M_\alpha \omega^m \boldsymbol{u}^m \cdot \left( \sum_\eta \Gamma_{k,\eta}^{-1} c_t^{(0)} \boldsymbol{F}_\eta \right). \tag{A.95}$$

Note that the balance of $\boldsymbol{F}_k$ ensures the balance of $h_k^{(1)}$ after summation over $k$.

### A.1.2.2 Species momentum transport with external forces

So far the mass transport of species and the Maxwell-Stefan diffusion equation with external force for nonideal liquid mixture has been recovered. Now, to recover momentum equation i.e the Navier Stokes equations for each species, lets start by setting $i = 1$ in Eq. A.78 to get

$$\begin{aligned}
&\partial_t \mathbf{p}_k^{(1)} + \nabla \cdot \langle 1, \boldsymbol{u}^m \otimes \boldsymbol{u}^m f_k^{m,(2)} \rangle \\
&= \lambda_k \sum_{\substack{j,l \geq 0 \\ j+l=3}} \langle 1, \boldsymbol{u}^m f_k^{eq,m}(\rho_k^{(j)}, \boldsymbol{v}_1^{(l)}, \cdots, \boldsymbol{v}_N^{(l)}) - \mathbf{p}_k^{(3)} \rangle \\
&= \lambda_k \sum_{\substack{j,l \geq 0 \\ j+l=3}} \rho_k^{(j)} \left( \mathbf{v}_k^{eq,(l)} - \mathbf{v}_k^{(l)} \right) + \boldsymbol{h}_k^{(3)}.
\end{aligned} \tag{A.96}$$

$f_k^{m,(2)}$ is required to derive macroscopic species momentum equations with external force and it is derived by setting $i = 0$ in Eq. A.75

$$\begin{aligned}
&\partial_t f_k^{m,(0)} + \boldsymbol{u}^m \cdot \nabla f_k^{m,(1)} \\
&= \lambda_k f_k^{eq,m,L}(\rho_k^{(2)}) + \lambda_k \left( f_k^{eq,m,Q}(\rho_k^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(1)}) \right) \\
&- \lambda_k f_k^{m,(2)} + \underbrace{d_k^{m,(2)}}_{\overset{Eq.\ A.80}{=} 0}
\end{aligned} \tag{A.97}$$

Above equation can be written explicit for $f_k^{m,(2)}$ with its leading order coefficients $f_k^{m,(0)} = \omega^m s_k^m \rho_k^{(0)}$ and $f_k^{m,(1)}$ as

$$\begin{aligned}
f_k^{m,(2)} &= f_k^{eq,m,L}(\rho_k^{(2)}) + \left( f_k^{eq,m,Q}(\rho_k^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(1)}) \right) \\
&- \frac{1}{\lambda_k} \partial_t \omega^m s_k^m \rho_k^{(0)} - \frac{1}{\lambda_k} \boldsymbol{u}^m \cdot \nabla f_k^{m,(1)}.
\end{aligned} \tag{A.98}$$

From species mass conservation Eq. A.82, above equation can be rewritten as

$$f_k^{m,(2)} = f_k^{eq,m,L}(\rho_k^{(2)}) + \left(f_k^{eq,m,Q}(\rho_k^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(1)})\right) + \frac{\omega_m s_k^m}{\lambda_k}\nabla\cdot(\rho_k^{(0)}\mathbf{v}_k^{(1)}) - \frac{1}{\lambda_k}\boldsymbol{u}^m\cdot\nabla f_k^{m,(1)}. \tag{A.99}$$

Only unknown here is the first order coefficient $f_k^{m,(1)}$ and it can be obtained by setting $i=-1$ in Eq. A.75

$$\boldsymbol{u}^m\cdot\nabla f_k^{m,(0)} = \lambda_k f_k^{eq,m,BL}(\rho_k^0, \mathbf{v}_k^{eq}(\boldsymbol{v}_1^{(1)},\cdots,\boldsymbol{v}_N^{(1)})) - \lambda_k f_k^{m,(1)} + d_k^{m,(1)} \tag{A.100}$$

and rewriting for the first order part gives,

$$f_k^{m,(1)} = f_k^{eq,m,BL}(\rho_k^{(0)}, \mathbf{v}_k^{eq}(\boldsymbol{v}_1^{(1)},\cdots,\boldsymbol{v}_N^{(1)})) - \frac{1}{\lambda_k}\boldsymbol{u}^m\cdot\nabla f_k^{m,(0)} + \frac{1}{\lambda_k}d_k^{m,(1)}. \tag{A.101}$$

Substituting $f_k^{m,(0)}$ Eq. A.86 and equilibrium definitions of linear Eq. A.71a and bi-linear Eq. A.71b part gives

$$f_k^{m,(1)} = \omega^m\rho_k^{(0)}\frac{1}{c_s^2}\left(\boldsymbol{u}^m\cdot\mathbf{v}_k^{eq,(1)}\right) - \frac{1}{\lambda_k}\omega^m\boldsymbol{u}^m\cdot\nabla s_m^k\rho_k^{(0)} + \frac{1}{\lambda_k}d_k^{m,(1)} \tag{A.102}$$

Since the lattice velocity at rest is zero, $s_k$ can be replaced by $\phi_k$ for $m \neq center$ and using Eq. A.89 results in

$$\begin{aligned} f_k^{m,(1)} &= \omega^m\rho_k^{(0)}\frac{1}{c_s^2}\left(\boldsymbol{u}^m\cdot\mathbf{v}_k^{eq,(1)}\right) \\ &\quad - \omega^m\boldsymbol{u}^m\cdot\left(\frac{1}{c_s^2}\rho^{(0)}\left(\boldsymbol{v}_k^{eq,(1)} - \boldsymbol{v}_k^{(1)}\right)\right. \\ &\quad \left. + \frac{1}{\lambda_k c_s^2}\boldsymbol{h}_k^{(1)}\right) + \frac{1}{\lambda_k}d_k^{m,(1)} \\ &= \frac{\omega^m}{c_s^2}\rho_k^{(0)}\boldsymbol{u}^m\cdot\boldsymbol{v}_k^{(1)} - \frac{\omega^m}{\lambda_k c_s^2}\boldsymbol{u}^m\cdot\boldsymbol{h}_k^{(1)} + \frac{1}{\lambda_k}d_k^{m,(1)}. \end{aligned} \tag{A.103}$$

Lets substitute $f_k^{m,(1)}$ in Eq. A.99

$$
\begin{aligned}
f_k^{m,(2)} &= f_k^{eq,m,L}(\rho_k^{(2)}) + \left(f_k^{eq,m,Q}(\rho_k^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(1)})\right) \\
&+ \frac{\omega_m s_k^m}{\lambda_k} \nabla \cdot (\rho_k^{(0)} \mathbf{v}_k^{(1)}) \\
&- \frac{1}{\lambda_k} \boldsymbol{u}^m \cdot \nabla \left( \frac{\omega^m}{c_s^2} \rho_k^{(0)} \boldsymbol{u}^m \cdot \boldsymbol{v}_k^{(1)} - \frac{\omega^m}{\lambda_k c_s^2} \boldsymbol{u}^m \cdot \boldsymbol{h}_k^{(1)} + \frac{1}{\lambda_k} d_k^{m,(1)} \right).
\end{aligned} \tag{A.104}
$$

Lets derive the second order tensor $\langle 1, \boldsymbol{u}^m \otimes \boldsymbol{u}^m f_k^{m,(2)} \rangle$ which is required recover the momentum transport equation similar to single component case Eq. A.33 and taking advantage of Eq. A.72 and Eq. A.74 for linear and quadratic equilibrium term gives

$$
\begin{aligned}
&\langle 1, \boldsymbol{u}^m \otimes \boldsymbol{u}^m f_k^{m,(2)} \rangle \\
&= \phi_k c_s^2 \rho_k^{(2)} \boldsymbol{I} + (\rho_k^{(0)} \boldsymbol{v}^{(1)} \otimes \boldsymbol{v}^{(1)}) \\
&+ \frac{1}{\lambda_k} \underbrace{\langle 1, \boldsymbol{u}^m \otimes \boldsymbol{u}^m (\omega_m s_k^m) \rangle}_{\overset{Eq.3.13}{=} \phi_k c_s^2} \nabla \cdot (\rho_k^{(0)} \mathbf{v}_k^{(1)}) \boldsymbol{I} \\
&- \frac{\kappa c_s^2}{\lambda_k} \left( \nabla \cdot (\rho_k^{(0)} \mathbf{v}_k^{(1)}) \boldsymbol{I} + \nabla (\rho_k^{(0)} \mathbf{v}_k^{(1)})^T + \nabla (\rho_k^{(0)} \mathbf{v}_k^{(1)}) \right) \\
&+ \frac{1}{\lambda_k} \langle 1, \boldsymbol{u}^m \otimes \boldsymbol{u}^m \left( \boldsymbol{u}^m \cdot \nabla \left( \frac{\omega^m}{\lambda_k c_s^2} \boldsymbol{u}^m \cdot \boldsymbol{h}_k^{(1)} - \frac{1}{\lambda_k} d_k^{m,(1)} \right) \right) \rangle \\
&= \phi_k c_s^2 \rho_k^{(2)} \boldsymbol{I} + (\rho_k^{(0)} \boldsymbol{v}^{(1)} \otimes \boldsymbol{v}^{(1)}) + \frac{(\phi_k - \kappa) c_s^2}{\lambda_k} \nabla \cdot (\rho_k^{(0)} \mathbf{v}_k^{(1)}) \boldsymbol{I} \\
&- \frac{\kappa c_s^2}{\lambda_k} \left( \nabla (\rho_k^{(0)} \mathbf{v}_k^{(1)})^T + \nabla (\rho_k^{(0)} \mathbf{v}_k^{(1)}) \right) + \boldsymbol{L}_k^{(1)}
\end{aligned} \tag{A.105}
$$

where $\boldsymbol{L}_k$ expresses the second order moments of the forcing term given as

$$
\boldsymbol{L}_k^{(1)} = \frac{1}{\lambda_k} \langle 1, \boldsymbol{u}^m \otimes \boldsymbol{u}^m \left( \boldsymbol{u}^m \cdot \nabla \left( \frac{\omega^m}{\lambda_k c_s^2} \boldsymbol{u}^m \cdot \boldsymbol{h}_k^{(1)} - \frac{1}{\lambda_k} d_k^{m,(1)} \right) \right) \rangle.
$$

Using Eq. A.80, Eq. 3.13 and using same algebra similar to single component second order tensor of PDF i.e Eq. A.31 for forcing term results

$$
\begin{aligned}
\boldsymbol{L}_k^{(1)} &= \frac{1}{\lambda_k^2}\langle 1, \boldsymbol{u}^m \otimes \boldsymbol{u}^m \left( \boldsymbol{u}^m \cdot \nabla \left( \frac{\omega^m}{c_s^2} \boldsymbol{u}^m \cdot \boldsymbol{h}_k^{(1)} - \omega_m \boldsymbol{u}^m \cdot \boldsymbol{G}_k^{(1)} \right) \right) \rangle \\
&= \frac{1}{\lambda_k^2} \Bigg( \kappa c_s^2 \Big( (\nabla \cdot \boldsymbol{h}_k^{(1)}) \boldsymbol{I} + \nabla (\boldsymbol{h}_k^{(1)})^T + \nabla \boldsymbol{h}_k^{(1)} \Big) \\
&- \kappa c_s^4 \Big( (\nabla \cdot \boldsymbol{G}_k^{(1)}) \boldsymbol{I} + \nabla (\boldsymbol{G}_k^{(1)})^T + \nabla (\boldsymbol{G}_k^{(1)}) \Big) \Bigg) \\
&= \frac{1}{\lambda_k^2} \Bigg( \kappa c_s^2 \Bigg( \nabla \cdot \left( \sum_m \omega^m \boldsymbol{u}^m \boldsymbol{u}^m \boldsymbol{u}^m \cdot \boldsymbol{G}_k^{(1)} \right) \boldsymbol{I} + \nabla \left( \sum_m \omega^m \boldsymbol{u}^m \boldsymbol{u}^m \boldsymbol{u}^m \cdot \boldsymbol{G}_k^{(1)} \right)^T \\
&+ \nabla \left( \sum_m \omega^m \boldsymbol{u}^m \boldsymbol{u}^m \boldsymbol{u}^m \cdot \boldsymbol{G}_k^{(1)} \right) \Bigg) \\
&- \kappa c_s^4 \Big( \nabla \cdot (\boldsymbol{G}_k^{(1)}) \boldsymbol{I} + \nabla (\boldsymbol{G}_k^{(1)})^T + \nabla (\boldsymbol{G}_k^{(1)}) \Big) \Bigg) = 0. \qquad \text{(A.106)}
\end{aligned}
$$

Finally, the species momentum equation up to second order in $\epsilon$ can be derived by substituting the second order tensor $\langle 1, \boldsymbol{u}^m \otimes \boldsymbol{u}^m f_k^{m,(2)} \rangle$ Eq. A.105 and setting $\mathbf{p}_k^{(1)} = \rho_k^{(0)} \boldsymbol{v}_k^{(1)}$ and $\kappa = 1$

$$
\begin{aligned}
\partial_t (\rho_k^{(0)} \boldsymbol{v}_k^{(1)}) + \nabla \cdot \Bigg[ & (\rho_k^{(0)} \boldsymbol{v}^{(1)} \otimes \boldsymbol{v}^{(1)}) + \Bigg( \phi_k c_s^2 \rho_k^{(2)} \\
& + \frac{(\phi_k - 1) c_s^2}{\lambda_k} \nabla \cdot (\rho_k^{(0)} \mathbf{v}_k^{(1)}) \Bigg) \boldsymbol{I} \\
& - \frac{c_s^2}{\lambda_k} \Big( \nabla (\rho_k^{(0)} \mathbf{v}_k^{(1)})^T + \nabla (\rho_k^{(0)} \mathbf{v}_k^{(1)}) \Big) \Bigg] \\
& = \lambda_k \sum_{\substack{j,l \geq 0 \\ j+l=3}} \rho_k^{(j)} \left( \mathbf{v}_k^{eq,(l)} - \mathbf{v}_k^{(l)} \right) + \boldsymbol{h}_k^{(3)}
\end{aligned}
$$

$$\begin{aligned}\implies \partial_t(\rho_k^{(0)}\boldsymbol{v}_k^{(1)}) + \nabla\cdot(\rho_k^{(0)}\boldsymbol{v}^{(1)}\otimes\boldsymbol{v}^{(1)}) &= -\phi_k c_s^2\nabla\rho_k^{(2)} \\ &- \frac{(\phi_k-1)c_s^2}{\lambda_k}\nabla\cdot\nabla\cdot(\rho_k^{(0)}\mathbf{v}_k^{(1)}) \\ &+ \frac{c_s^2}{\lambda_k}\nabla\cdot\left(\nabla(\rho_k^{(0)}\mathbf{v}_k^{(1)})^T + \nabla(\rho_k^{(0)}\mathbf{v}_k^{(1)})\right) \\ &+ \lambda_k\sum_{\substack{j,l\geq 0\\ j+l=3}}\rho_k^{(j)}\left(\mathbf{v}_k^{eq,(l)} - \mathbf{v}_k^{(l)}\right) + c_s^2\boldsymbol{G}_k^{(3)}. \end{aligned} \tag{A.107}$$

#### A.1.2.3 Mixture transport with external forces

Now, lets derive the incompressible Navier Stokes equation for mixture i.e mass and momentum conservation equations. The mass conservation of mixture can be derived by summing over species $k$ of species mass conservation Eq. A.82

$$\partial_t\rho^{(0)} + \nabla\cdot\left(\rho^{(0)}\boldsymbol{v}^{(1)}\right) = 0. \tag{A.108}$$

Like $c_t^{(0)}$, the zeroth order mass density $\rho^{(0)}$ is fixed at initial time and remains constant throughout the domain over time. With this assumption, above equation reduces to incompressible mass conservation equation for mixture

$$\nabla\cdot\boldsymbol{v}^{(1)} = 0. \tag{A.109}$$

By summing over species $k$ of the species momentum transport Eq. A.107 yields

$$\begin{aligned}\partial_t(\rho^{(0)}\boldsymbol{v}^{(1)}) + \nabla\cdot(\rho^{(0)}\boldsymbol{v}^{(1)}\otimes\boldsymbol{v}^{(1)}) &= -\nabla\sum_k\phi_k c_s^2\rho_k^{(2)} \\ &+ \frac{\rho^{(0)}c_s^2}{\lambda_k}\nabla\cdot\left(\nabla(\boldsymbol{v}^{(1)})^T + \nabla(\boldsymbol{v}^{(1)})\right) \\ &+ c_s^2\sum_k\boldsymbol{G}_k^{(3)}\end{aligned} \tag{A.110}$$

Thus, the incompressible momentum transport equation for mixture can be obtained by dividing the equation with $\rho^{(0)}$ and setting the kinematic pressure

$$\tilde{P}^{(2)} = c_s^2\frac{\sum_k\phi_k\rho_k^{(2)}}{\rho^{(0)}}, \tag{A.111}$$

the kinematic viscosity

$$\nu = \frac{c_s^2}{\lambda_k}, \tag{A.112}$$

and the force term

$$\boldsymbol{F} = c_s^2 \sum_k G_k^{(3)} \tag{A.113}$$

gives

$$\partial_t \boldsymbol{v}^{(1)} + \nabla \cdot (\boldsymbol{v}^{(1)} \otimes \boldsymbol{v}^{(1)}) = -\nabla \tilde{P} + \frac{1}{\nu} \nabla \cdot \left( \nabla (\boldsymbol{v}^{(1)})^T + \nabla (\boldsymbol{v}^{(1)}) \right) + \boldsymbol{F}. \tag{A.114}$$

The total mixture force term per unit volume $\boldsymbol{F}$ can be identified by comparing above equation with the incompressible Navier-Stokes Eq. 2.56

$$\boldsymbol{F} = \boldsymbol{g} + \frac{\rho^{e,(0)}}{\rho^{(0)}} \hat{\boldsymbol{E}} \tag{A.115}$$

where $\boldsymbol{g}$ and $\hat{\boldsymbol{E}}$ are acceleration due to gravitational field and external electrical field acting on all the components in the same way. Thus, the forcing term $\boldsymbol{G}_k^{(3)}$ in species momentum Eq. A.107 can be written as

$$\boldsymbol{G}_k^{(3)} = \frac{1}{c_s^2} y_k \left( \rho^{(0)} \boldsymbol{F} \right) = \frac{1}{c_s^2} y_k \left( \rho^{(0)} \boldsymbol{g} + \rho^{e,(0)} \hat{\boldsymbol{E}} \right). \tag{A.116}$$

This concludes that the multicomponent LBE mixture forces $d_k^{(m,3)}$ on species $k$ as

$$d_k^{m,(3)} = \omega^m \boldsymbol{u}^m \cdot \boldsymbol{G}_k^{(3)} \qquad = \frac{\omega^m}{c_s^2} \boldsymbol{u}^m \cdot y_k \left( \rho^{(0)} \boldsymbol{F} \right). \tag{A.117}$$

Thus, total diffusive driving force on multicomponent LBE for each species $k$ can be written as

$$\begin{aligned} d_k^m &= d_k^{m,(1)} + d_k^{m,(3)} \\ &= \underbrace{min_\alpha M_\alpha \omega^m \boldsymbol{u}^m \cdot \left( \sum_\eta \Gamma_{k,\eta}^{-1} c_t^{(0)} \boldsymbol{F}_\eta \right)}_{\text{Diffusive driving force on species } k} \\ &+ \underbrace{\frac{\omega^m}{c_s^2} \boldsymbol{u}^m \cdot y_k \left( \rho^{(0)} \boldsymbol{F} \right)}_{\text{Mixture fraction force on species } k}. \end{aligned} \tag{A.118}$$

It's worth mentioning that the diffusive driving force term $d_k^{(m,1)}$ Eq. A.95 is consistent with the incompressible limit due to the fact that the mixture force term is of order $\mathcal{O}(\epsilon^3)$ and is small compared to the mixture velocity

$\boldsymbol{v}^{(1)}$. Similar to single component case, the mixture velocity $\boldsymbol{v}^{(1)}$ and the kinematic pressure term $\tilde{P}^{(2)}$ can be expressed in terms of $\epsilon$ as

$$\boldsymbol{v}^{(1)} = \frac{1}{\epsilon}\boldsymbol{v} + \mathcal{O}(\epsilon^2) \tag{A.119}$$

$$\begin{aligned}\tilde{P}^{(2)} &= \frac{c_s^2}{\epsilon^2}\frac{\sum_k \phi_k \rho_k^{(2)} - \rho^{(0)}}{\rho^{(0)}} + \mathcal{O}(\epsilon^2) \\ &= \frac{c_s^2}{\epsilon^2}\frac{\sum_k \phi_k \rho_k^{(2)} - min_\alpha M_\alpha c_t^{(0)}}{\rho^{(0)}} + \mathcal{O}(\epsilon^2)\end{aligned} \tag{A.120}$$

which implies that this velocity and pressure yields at least second order accurate approximation of the incompressible Navier-Stokes equations.

### A.1.2.4 Time discretization

Here, the semi-discrete MRT-Boltzmann equation Eq. 3.39 is discretized in time by integrating along its characteristics $\boldsymbol{u}^m$ same as in App. A.1.1.1

$$\begin{aligned}\int_0^{\delta t} \partial_t f_k^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) + \boldsymbol{u}^m \cdot \nabla f_k^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) ds \\ = \int_0^{\delta t} \underbrace{\Omega_k^m}_{:=\boldsymbol{A}_k(f_k^{eq,m} - f_k^m)} (\boldsymbol{x} + \boldsymbol{u}^m s, t+s) ds \\ + \int_0^{\delta t} d_k^m(\boldsymbol{x} + \boldsymbol{u}^m s, t+s) ds.\end{aligned} \tag{A.121}$$

Using, forward Euler to approximate the collision integral would lead to lack of mass conservation [7]. So, lets approximate the integrant of both collision and force term by trapezium rule and the resultant fully discrete MRT-LBE with force term can be written as

$$\begin{aligned}&f_k^m(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t+\delta t) - f_k^m(\boldsymbol{x}, t) \\ &= \frac{\delta t \boldsymbol{A}_k}{2}\Big[(f_k^{eq,m}(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t+\delta t) - f_k^m(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t+\delta t)) \\ &+ (f_k^{eq,m}(\boldsymbol{x}, t) - f_k^m(\boldsymbol{x}, t))\Big] \\ &+ \frac{\delta t}{2}\Big[d_k^m(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t+\delta t) + d_k^m(\boldsymbol{x}, t)\Big].\end{aligned} \tag{A.122}$$

This is an implicit equation similar to that of single component case so variable transformation [6] is used to solve the equation explicitly by

introducing a new variables $\bar{f}_k^m$

$$\bar{f}_k^m(\boldsymbol{x},t) = f_k^m(\boldsymbol{x},t) + \frac{\delta t \boldsymbol{A}_k}{2}\left(f_k^m(\boldsymbol{x},t) - f_k^{eq,m}(\boldsymbol{x},t)\right) - \frac{\delta t}{2} d_k^m(\boldsymbol{x},t) \tag{A.123}$$

Using this in Eq. A.122 gives second order fully discrete multicomponent LBE with force term for species $k$

$$\begin{aligned}
&\bar{f}_k^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) - \bar{f}_k^m(\boldsymbol{x},t) \\
&= (\delta t \boldsymbol{A}_k)\left(\boldsymbol{I} + \frac{\delta t \boldsymbol{A}_k}{2}\right)^{-1}\left(f_k^{eq,m}(\boldsymbol{x},t) - \bar{f}_k^m(\boldsymbol{x},t)\right) \\
&+ \delta t\left(\boldsymbol{I} + \frac{\delta t \boldsymbol{A}_k}{2}\right)^{-1} d_k^m(\boldsymbol{x},t).
\end{aligned} \tag{A.124}$$

Since the species density $\rho_k$ moments are conserved in collision and $\sum_m d_k^m = 0$, it is obtained directly from transformed variable.

$$\rho_k = \sum_m \bar{f}_k^m = \sum_m f_k^m. \tag{A.125}$$

But due to modified species velocity $\mathbf{v}_k^{eq}$ in equilibrium function, the species momentums $\mathbf{p}_k$ are not conserved in collision. Therefore, a linear equation system needs to be solved to get momentum $\mathbf{p}_k$ of untransformed variable $f_k^m$ by applying first order moment on Eq. A.123

$$\begin{aligned}
\bar{\mathbf{p}}_k = \langle 1, \boldsymbol{u}^m \bar{f}_k^m\rangle &= \mathbf{p}_k + \frac{\delta t \lambda_k^{\mathcal{D}}}{2}\left(\mathbf{p}_k - \langle 1, \boldsymbol{u}^m f_k^{eq,m}\rangle\right) - \sum_m \boldsymbol{u}^m d_k^m \\
&= \mathbf{p}_k + \frac{\delta t \lambda_k^{\mathcal{D}}}{2}\sum_l \mathbf{p}_l \phi_l \sum_\zeta \Gamma_{k,l}^{-1}\frac{\mathcal{D}}{\mathcal{D}_{l,\zeta}}\chi_\zeta \\
&- \frac{\delta t \lambda_k^{\mathcal{D}}}{2}\sum_\zeta \mathbf{p}_\zeta \phi_\zeta \sum_l \Gamma_{k,l}^{-1}\frac{\mathcal{D}}{\mathcal{D}_{l,\zeta}}\chi_l \\
&- \frac{\delta t}{2} c_s^2 \left(min_\alpha(M_\alpha)\sum_l \Gamma_{k,l}^{-1} c_t^0 \boldsymbol{F}_l + \frac{1}{c_s^2} y_k\left(\rho^{(0)}\boldsymbol{F}\right)\right).
\end{aligned} \tag{A.126}$$

For ideal mixture $\mathbf{\Gamma} = \boldsymbol{I}$, above equation can be written as

$$\bar{\mathbf{p}}_k = \left(1 + \frac{\delta t \lambda_k^{\mathcal{D}}}{2}\phi_k \sum_l \frac{\mathcal{D}}{\mathcal{D}_{k,l}}\chi_l\right)\mathbf{p}_k - \frac{\delta t \lambda^{\mathcal{D}_k}}{2}\chi_k \sum_l \frac{\mathcal{D}}{\mathcal{D}_{k,l}}\phi_l \mathbf{p}_l - \frac{\delta t}{2} c_s^2 \left(min_\alpha(M_\alpha) c_t^0 \boldsymbol{F}_k + \frac{1}{c_s^2} y_k \left(\rho^{(0)} \boldsymbol{F}\right)\right). \tag{A.127}$$

This linear equation system must be solved at evert time step for every lattice cell to obtain the species momentums $\mathbf{p}_k$. it is worth mentioning that summing up previous equation results that the total mixture momentum $\mathbf{p}$ which is conserved, therefore it is possible to compute it directly from $\bar{f}_k^m$ as

$$\mathbf{p} = \rho \boldsymbol{v} = \sum_k \sum_m \boldsymbol{u}^m \bar{f}_k^m. \tag{A.128}$$

After obtaining species momentums, the collide and the streaming steps are carried out to complete the time step. In contrast to the implementation described in [7], the presented model considers variable diffusivities and thermodynamic factors and hence the element-wise variable system changes in each iteration. A detailed discussion on the solvability of this linear equation system is described in [103].

Notice that in Eq. A.126 and Eq. A.127, the external force term contributes to the calculation of species momentum $\mathbf{p}_k$ which is similar to single component LBM. As mentioned before, source terms in Musubi are added outside collision kernel so the force term in species momentum can be avoided if the forward Euler is used to approximate the integrand of the external force term. As the external forces are usually small (to be consistent with the low Mach regime) the limitations with respect to stability are moderate. Additionally, the stability of the scheme can be improved by choosing small $\delta t$. With first order approximation of the external force term, the fully discrete LBE Eq. A.122 for species $k$ is

$$\begin{aligned} &\bar{f}_k^m(\boldsymbol{x} + \boldsymbol{u}^m \delta t, t + \delta t) - \bar{f}_k^m(\boldsymbol{x}, t) \\ &= (\delta t \boldsymbol{A}_k)\left(\boldsymbol{I} + \frac{\delta t \boldsymbol{A}_k}{2}\right)^{-1}\left(f_k^{eq,m}(\boldsymbol{x}, t) - \bar{f}_k^m(\boldsymbol{x}, t)\right) \\ &+ \delta t d_k^m(\boldsymbol{x}, t). \end{aligned} \tag{A.129}$$

Thus, the linear equation system for nonideal and ideal mixture Eq. A.126

and Eq. A.127 to compute species momentum $\mathbf{p}_k$ are simplified to

$$\bar{\mathbf{p}}_k = \mathbf{p}_k + \frac{\delta t \lambda_k^{\mathcal{D}}}{2} \sum_l \mathbf{p}_l \phi_l \sum_\zeta \Gamma_{k,l}^{-1} \frac{\mathcal{D}}{\mathcal{D}_{l,\zeta}} \chi_\zeta - \frac{\delta t \lambda_k^{\mathcal{D}}}{2} \sum_\zeta \mathbf{p}_\zeta \phi_\zeta \sum_l \Gamma_{k,l}^{-1} \frac{\mathcal{D}}{\mathcal{D}_{l,\zeta}} \chi_l. \tag{A.130}$$

and

$$\bar{\mathbf{p}}_k = \left(1 + \frac{\delta t \lambda_k^{\mathcal{D}}}{2} \phi_k \sum_l \frac{\mathcal{D}}{\mathcal{D}_{k,l}} \chi_l\right) \mathbf{p}_k - \frac{\delta t \lambda_k^{\mathcal{D}}}{2} \chi_k \sum_l \frac{\mathcal{D}}{\mathcal{D}_{k,l}} \phi_l \mathbf{p}_l. \tag{A.131}$$

It's worth mentioning that the solvability conditions of the linear equations systems are not affected by the external forcing term and also the computational cost of solving the linear equation system is same for BGK and MRT collision models.

#### A.1.2.5 Effect of different velocities in equilibria

In previous analysis, the mixture velocity is used for the quadratic part of equilibrium function instead of species equilibrium velocity to recover correct nonlinear term in the incompressible Navier-Stokes equations for large diffusion velocities. This claim can be verified by studying the effect of different velocity in equilibrium distribution function with the help of well-known Taylor green test case for small and large diffusion velocities. For this, let us consider ternary mixture $H_2O$, $Na$ and $Cl$ without external forcing terms [94] to analyze the exactness of the proposed multicomponent LBM model for liquid mixture. The Taylor-Green vortex is an analytical solution of the incompressible Navier-Stokes equation and is defined by (for a two-dimensional, fully periodic domain of length $L_x = L_y = L$ in each direction):

$$\boldsymbol{v}(x,y,t) = \begin{pmatrix} \sin\left(\frac{2\pi}{L}x\right)\cos\left(\frac{2\pi}{L}y\right)\exp\left(-2\frac{4\pi^2}{L^2}\nu t\right) \\ -\cos\left(\frac{2\pi}{L}x\right)\sin\left(\frac{2\pi}{L}y\right)\exp\left(-2\frac{4\pi^2}{L^2}\nu t\right) \end{pmatrix}$$

$$P(x,y,t) = \frac{\rho}{4}\left(\cos\left(2\frac{2\pi}{L}x\right) + \cos\left(2\frac{2\pi}{L}y\right)\right) \cdot \exp\left(-4\frac{4\pi^2}{L^2}\nu t\right)$$

Furthermore, we prescribe homogenous mole density concentration fields

$$c_t = c_t^{(0)}, c_1 = c_2 = c_3 = c_t/3$$

such that no net-diffusive mass flux will be observed. The velocity distributions are given by

$$\begin{aligned}
\boldsymbol{v}_1\left(\boldsymbol{x}, t=0\right) &= \boldsymbol{v}(\boldsymbol{x}+\boldsymbol{r}_1, t=0) \\
\boldsymbol{v}_2\left(\boldsymbol{x}, t=0\right) &= \boldsymbol{v}(\boldsymbol{x}+\boldsymbol{r}_2, t=0) \\
\boldsymbol{v}_3\left(\boldsymbol{x}, t=0\right) &= \frac{\rho}{\rho_3}\boldsymbol{v} - \left(\frac{\rho_1}{\rho_3}\boldsymbol{v}_1 + \frac{\rho_2}{\rho_3}\boldsymbol{v}_2\right).
\end{aligned}$$

By construction, the upper initial conditions for the mixture are the usual Taylor-Green initial conditions. The vectors $\boldsymbol{r}_k = (r_k, r_k)^T$ with $r_k \in [0; L]$ for $k = 1, 2$ determine the phase shifts of the species velocities $\mathbf{v}_k$ with respect to the density averaged, mixture velocity $\boldsymbol{v}$ and the phase shift between $\boldsymbol{v}_1, \boldsymbol{v}_2$ can be varied by changing the ratio $r_1/r_2$. With species equilibrium velocity in quadratic part of equilibrium function

$$f_k^{eq,m,Q}\left(\rho_k, \mathbf{v}_k^{eq}\right) = \omega^m \rho_k \left(\frac{1}{2c_s^4}\left(\boldsymbol{u}^m \cdot \mathbf{v}_k^{eq}\right)^2 - \frac{1}{2c_s^2}(\mathbf{v}_k^{eq})^2\right),$$

the nonlinear term of the incompressible Navier-Stokes equation for mixture can be recovered only when diffusion velocities are small [38]

$$\boldsymbol{v} - \mathbf{v}_k \in \mathcal{O}(\epsilon),$$

For this study, the quadratic equilibrium part is generalized by a convex combination of $\boldsymbol{v}$ and $\mathbf{v}_k^{eq}$, i.e.

$$\begin{aligned}
f_k^{eq,m} =& f_k^{eq,m,L}\left(\rho_k\right) + f_k^{eq,m,BL}\left(\rho_k, \mathbf{v}_k^{eq}\right) \\
&+ f_k^{eq,m,Q}\left(\rho_k, \theta\boldsymbol{v} + (1-\theta)\,\mathbf{v}_k^{eq}, \theta\boldsymbol{v} + (1-\theta)\,\mathbf{v}_k^{eq}\right)
\end{aligned} \tag{A.132}$$

where $\theta \in [0; 1]$.

Figure A.1 shows the relative error between the analytical solution for the mixture's velocity component (in x direction) and the numerical results over simulation time. It can be observed that the error for small diffusion velocities (i.e. $\boldsymbol{v} \approx \mathbf{v}_k$) is independent of $\theta$. However, the situation changes when $r_1$ increases. In these situations $\theta = 1$ delivers much better results and the relative error is below $10^{-2}$ whereas the error for $\theta = 0$ is above 1.0 ( for $r_1 = L/10^4$ and $r_1 = L/10^3$). It is worth emphasizing that the relative error for $\theta = 1$ is in the range of $10^{-2}$, independent of the choice of diffusion velocities. Overall the numerical results are in good agreement to the theoretical expectations, pointing out that the correct mixture Navier-Stokes equation can be recovered by equilibrium Eq. A.132 for $\theta = 1$.

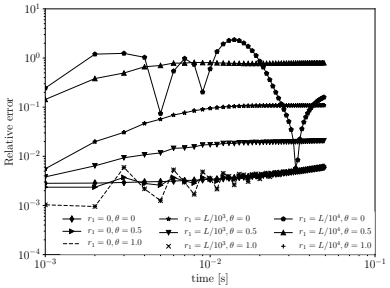



**Figure A.1** Relative mixture velocity error (x component) for $r_2 = 0$, different relative species velocity angles $r_1$ and different equilibrium velocities $\theta$. For $\theta = 1$ the correct nonlinearity of the incompressible Navier-Stokes equation is recovered.

### A.1.3 Lattice Boltzmann Method for Electric Potential

In this section, the BGK semi-discrete Boltzmann equation of electric potential is analyzed using asymptotic analysis under diffusive scaling (App. A.1.1) to recover the Poisson equation Eq. 2.64. The BGK semi-discrete Boltzmann equation for electric potential with source term can be written as

$$\partial_t f^m(\boldsymbol{x}, t) + \boldsymbol{u}^m \cdot \nabla f^m(\boldsymbol{x}, t) = \lambda_\gamma (f^{eq,m}(\boldsymbol{x}, t) - f^m(\boldsymbol{x}, t)) + \mathcal{S}^m. \quad \text{(A.133)}$$

Recalling the diffusive scaling used in App. A.1.1 i.e $\widetilde{\delta x} = \epsilon \delta x$ and $\widetilde{\delta t} = \epsilon^2 \delta t$ to above equation gives

$$\epsilon^2 \partial_t f^m + \epsilon \boldsymbol{u}^m \cdot \nabla f^m = \lambda^\gamma (f^{eq,m} - f^m) + \mathcal{S}^m \quad \text{(A.134)}$$

Using Eq. 3.13 in the equilibrium distribution function for electric potential Eq. 3.73 gives

$$\langle 1, f^{eq} \rangle = \psi \quad \text{(A.135a)}$$

$$\langle 1, u_\alpha f^{eq} \rangle = 0 \quad \text{(A.135b)}$$

$$\langle 1, u_\alpha u_\beta f^{eq} \rangle = c_s^2 \psi \delta_{\alpha,\beta}. \quad \text{(A.135c)}$$

Expanding the particle distribution function $f^m$, the source term $\mathcal{S}^m$ and the macroscopic electric potential $\psi$ asymptotically in the small parameter

$\epsilon$ same as Eq. A.3a:

$$f^m = \sum_{i=0}^{\infty} \epsilon^i f^{m,(i)} \tag{A.136a}$$

$$\mathcal{S}^m = \sum_{i=0}^{\infty} \epsilon^i \mathcal{S}^{m,(i)} \tag{A.136b}$$

$$\psi = \sum_{i=0}^{\infty} \epsilon^i \psi^{(i)} \tag{A.136c}$$

Applying the asymptotic expansion of $f^m$ Eq. A.136a in semi-discrete Boltzmann equation Eq. A.134 with $f^{m,(i)} = 0$ for $i < 0$. We get the following expression in the order $\epsilon^{i+2}$ for $i \geq -2$:

$$\partial_t f^{m,(i)} + \boldsymbol{u}^m \cdot \nabla f^{m,(i+1)} = \lambda^\gamma \left( f^{eq,m}(\psi^{(i+2)}) - f^{m,(i+2)} \right) + \mathcal{S}^{m,(i+2)} \tag{A.137}$$

The time dependent electric potential Eq. 3.71 is obtained by taking zeroth order moment of above equation i.e $\psi^{(i)} = \sum_m f^{m,(i)}$ and setting $i = 0$

$$\begin{aligned} \partial_t \psi^{(0)} + \nabla \cdot \langle 1, \boldsymbol{u} f^{(1)} \rangle &= \lambda^\gamma (\psi^{(2)} - \psi^{(2)}) + \sum_m \mathcal{S}^{m,(2)} \\ \implies \partial_t \psi^{(0)} + \nabla \cdot \langle 1, \boldsymbol{u} f^{(1)} \rangle &= \sum_m \mathcal{S}^{m,(2)} \end{aligned} \tag{A.138}$$

To determine the unknown variable $f^{m,(1)}$, the leading order coefficient $f^{m,(0)}$ must be determined. Setting $i = -2$ in Eq. A.137 and using $f^{m,(i)} = 0$ for $i < 0$ results in

$$f^{m,(0)} = f^{eq,m,(0)} + \frac{1}{\lambda^\gamma} \mathcal{S}^{m,(0)}. \tag{A.139}$$

However, taking the first moment of the equations gives $\mathcal{S}^{m,(0)} = 0$. Thus, above equation reduces to

$$f^{m,(0)} = \omega^m \psi^{(0)}. \tag{A.140}$$

Now, $f^{m,(1)}$ is determined by setting $i = -1$ in Eq. A.137

$$\begin{aligned} \boldsymbol{u}^m \cdot \nabla f^{m,(0)} &= \lambda^\gamma (f^{eq,m,(1)} - f^{m,(1)}) + \mathcal{S}^{m,(1)} \\ \boldsymbol{u}^m \cdot \nabla (\omega^m \psi^{(0)}) &= \lambda^\gamma (\omega^m \psi^{(1)} - f^{m,(1)}) + \mathcal{S}^{m,(1)}. \end{aligned} \tag{A.141}$$

Taking zeroth moment of above equation gives $\mathcal{S}^{m,(1)} = 0$. Thus, $f^{m,(1)}$ can be written as

$$f^{m,(1)} = \omega^m \psi^{(1)} - \frac{1}{\lambda^\gamma} \boldsymbol{u}^m \cdot \nabla(\omega^m \psi^{(0)}), \tag{A.142}$$

Taking first moment of the above equation gives

$$\nabla \psi^{(0)} = -\frac{\lambda^\gamma}{c_s^2} \sum_m \boldsymbol{u}^m f^{m,(1)} \tag{A.143}$$

since $\sum \boldsymbol{u}^m \omega^m \psi^{(1)} = 0$ from Eq. A.135. From the expansion of $f^m$ Eq. A.136a, $f^{m,(1)}$ can be approximated as $f^{m,(1)} = \frac{1}{\epsilon}(f^m - f^{m,(0)}) + O(\epsilon)$. Using this relation together with $\langle 1, u_\alpha f^{m,(0)} \rangle = 0$ and setting $\epsilon = \delta x$ from diffusive scaling in above equation gives

$$\nabla \psi^{(0)} = -\frac{\lambda^\gamma}{c_s^2 \delta x} \sum_m \boldsymbol{u}^m f^m. \tag{A.144}$$

Thus, the macroscopic electric field $\boldsymbol{E}^{(0)} = -\nabla \psi^{(0)}$ can be computed locally using

$$\boldsymbol{E}^{(0)} = \frac{\lambda^\gamma}{c_s^2 \delta x} \sum_m \boldsymbol{u}^m f^m. \tag{A.145}$$

Finally, lets substitute Eq. A.142 in Eq. A.138

$$\begin{aligned} &\partial_t \psi^{(0)} - \frac{1}{\lambda^\gamma} \nabla \cdot \sum_m \boldsymbol{u}^m \boldsymbol{u}^m \cdot \nabla(\omega^m \psi^{(0)}) = \sum_m \mathcal{S}^{m,(2)} \\ \implies &\partial_t \psi^{(0)} = \frac{c_s^2}{\lambda^\gamma} \nabla^2 \psi^{(0)} + \sum_m \mathcal{S}^{m,(2)}. \end{aligned} \tag{A.146}$$

Comparing this equation with Eq. 3.71 gives the relation between relaxation parameter $\lambda^\gamma$ and the potential diffusivity $\gamma$ as

$$\gamma = \frac{c_s^2}{\lambda^\gamma} \tag{A.147}$$

To satisfy the Poisson's equation Eq. 2.64 in a steady state situation, i.e. when $\partial_t \psi^{(0)} = 0$ the source term $\mathcal{S}^{m,(2)}$ must fulfill the following condition

$$\mathcal{S}^{m,(2)} = \omega^m \gamma \frac{\rho^e}{\epsilon_r \epsilon_0}. \tag{A.148}$$

### A.1.4 Time Discretization

To obtain the fully discrete LBE for electric potential, Eq. A.133 must be approximated in $\boldsymbol{x}$ and $t$ similar to App. A.1.1.1. Integrating Eq. A.133 along its characteristics for a time interval 0 to $\delta t$ gives

$$\begin{aligned} & f^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) - f^m(\boldsymbol{x},t) \\ & = \int_0^{\delta t} \left(\Omega^m(\boldsymbol{x}+\boldsymbol{u}^m s, t+s) + \mathcal{S}^m(\boldsymbol{x}+\boldsymbol{u}^m s, t+s)\right) ds \end{aligned} \tag{A.149}$$

where, $\Omega^m$ is the collision term with single relaxation parameter $\lambda^\gamma$ is

$$\Omega^m(\boldsymbol{x}+\boldsymbol{u}^m s, t+s) = \lambda^\gamma(f^{eq,m}(\boldsymbol{x}+\boldsymbol{u}^m s, t+s) - f^m(\boldsymbol{x}+\boldsymbol{u}^m s, t+s)). \tag{A.150}$$

Approximating the integrant of both collision and source term with trapezium rule results in

$$\begin{aligned} & f^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) - f^m(\boldsymbol{x},t) \\ & = \frac{\delta t\lambda^\gamma}{2}\Big[(f^{eq,m}(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) - f^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t)) \\ & + (f^{eq,m}(\boldsymbol{x},t) - f^m(\boldsymbol{x},t))\Big] \\ & + \frac{\delta t}{2}\Big[\mathcal{S}^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) + \mathcal{S}^m(\boldsymbol{x},t)\Big] \end{aligned} \tag{A.151}$$

This is an implicit equation since it requires equilibrium at new time step. So, lets introduce the new variables $\bar{f}^m$

$$\bar{f}^m(\boldsymbol{x},t) = f^m(\boldsymbol{x},t) + \frac{\lambda^\gamma\delta t}{2}\Big(f^m(\boldsymbol{x},t) - f^{eq,m}(\boldsymbol{x},t)\Big) - \frac{\delta t}{2}\mathcal{S}^m(\boldsymbol{x},t) \tag{A.152}$$

to solve Eq. A.151 explicitly [34] which is same as in App. A.1.1.1. Thus, the second-order fully discrete LBE for electric potential with source term in $\bar{f}^m$ is

$$\begin{aligned} \bar{f}^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) - \bar{f}^m(\boldsymbol{x},t) = & \frac{\delta t\lambda^\gamma}{\left(1+\frac{\delta t\lambda^\gamma}{2}\right)}\left(f^{eq,m}(\boldsymbol{x},t) - \bar{f}^m(\boldsymbol{x},t)\right) \\ & + \frac{\delta t}{\left(1+\frac{\delta t\lambda^\gamma}{2}\right)}\mathcal{S}^m(\boldsymbol{x},t) \end{aligned} \tag{A.153}$$

Above equation can be rewritten with modified relaxation parameter

$$\bar{\lambda}^\gamma = \frac{\delta t\lambda^\gamma}{\left(1+\frac{\delta t\lambda^\gamma}{2}\right)} \tag{A.154}$$

as

$$\bar{f}^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) - \bar{f}^m(\boldsymbol{x},t) = \bar{\lambda}^\gamma \left(f^{eq,m}(\boldsymbol{x},t) - \bar{f}^m(\boldsymbol{x},t)\right) + \delta t \left(1 - \frac{\bar{\lambda}^\gamma}{2}\right) \mathcal{S}^m(\boldsymbol{x},t). \quad \text{(A.155)}$$

Substituting $\lambda^\gamma = \frac{c_s^2}{\gamma}$ in $\bar{\lambda}^\gamma$ gives

$$\gamma = \delta t c_s^2 \left(\frac{1}{\bar{\lambda}^\gamma} - \frac{1}{2}\right) \quad \text{(A.156)}$$

which is same as Eq. 3.77. Thus, as long as relaxation parameter is chosen according to this equation, the LBE for electric potential is second order in time $\delta t$. However, taking zeroth moment of Eq. A.152 gives macroscopic electric potential $\bar{\phi}$ as

$$\bar{\phi} = \sum_m \bar{f}^m = \phi - \frac{\delta t}{2}\gamma \frac{\rho^e}{\epsilon_r \epsilon_0}. \quad \text{(A.157)}$$

Once again we have macroscopic quantity depending on source term which is not supported in current *Musubi* implementation. Therefore, the integrant of source term in Eq. A.149 is approximated with forward Euler. It would reduce the scheme to first order but the LBE for electric potential is solved at every time till convergence and time step $\delta t$ is an artificial time introduced to solve Poisson's equation. Therefore, $\delta t$ can be chosen small enough to maintain stability and accuracy of the scheme. The transformed variables $\bar{f}$ can be rewritten without source term as

$$\bar{f}^m(\boldsymbol{x},t) = f^m(\boldsymbol{x},t) + \frac{\lambda^\gamma \delta t}{2}\Big(f^m(\boldsymbol{x},t) - f^{eq,m}(\boldsymbol{x},t)\Big). \quad \text{(A.158)}$$

Thus, the fully discrete BGK-LBE for electrical potential with first-order source term can be written as

$$\bar{f}^m(\boldsymbol{x}+\boldsymbol{u}^m\delta t, t+\delta t) - \bar{f}^m(\boldsymbol{x},t) = \bar{\lambda}^\gamma \left(f^{eq,m}(\boldsymbol{x},t) - \bar{f}^m(\boldsymbol{x},t)\right) + \delta t \mathcal{S}^m(\boldsymbol{x},t). \quad \text{(A.159)}$$

Now, the macroscopic electric potential $\psi$ can be computed directly from transformed PDF $\bar{f}^m$

$$\psi = \sum_m \bar{f}^m = \sum_m f^m \quad \text{(A.160)}$$

since electric potential is conserved in collision. But the non-conserved macroscopic electric field $\boldsymbol{E}$ given in Eq. A.145 becomes

$$\begin{aligned}\boldsymbol{E} &= \frac{\lambda^{\gamma}}{c_s^2 \delta x} \sum_m \boldsymbol{u}^m f^m = \frac{\lambda^{\gamma}}{1 + \frac{\delta t \lambda^{\gamma}}{2}} \frac{1}{c_s^2 \delta x} \sum_m \boldsymbol{u}^m \bar{f}^m \\ &= \frac{\bar{\lambda}^{\gamma}}{c_s^2 \delta x \delta t} \sum_m \boldsymbol{u}^m \bar{f}^m \end{aligned} \tag{A.161}$$

## A.2 Multiple Relaxation Time

In this section, the velocity vectors, weights and transformation matrix for $D3Q19$ layout are given.

### A.2.1 D3Q19 model

For D3Q19 model, the discrete lattice velocity vector, $\boldsymbol{u}^m$ and weights, $\omega^m$ are given as:

$$\boldsymbol{u}^m = \begin{pmatrix} u_x^m \\ u_y^m \\ u_z^m \end{pmatrix}$$

$$= \begin{pmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.162}$$

$$\omega^m = \begin{cases} 1/18 & \text{for } m = 1 \cdots 8 \\ 1/36 & \text{for } m = 9 \cdots 18 \\ 1/3 & \text{for } m = 19 \end{cases} \tag{A.163}$$

The most diagonal entries of collision matrix $\Lambda$ are set to 1 i.e $\lambda_{i,i} = 1$ except

$$\begin{aligned} &\lambda_{2.2} = \lambda_{\zeta}, \\ &\lambda_{10,10} = \lambda_{12,12} = \lambda_{14,14} = \lambda_{16,16} = \lambda_{\nu}. \end{aligned} \tag{A.164}$$

$\lambda_{\nu}$ and $\lambda_{\zeta}$ are relaxation parameter related to kinematic and bulk viscosity respectively, The transformation matrix $M$ with combination of discrete

velocity vectors which are orthonormal to each other for $D3Q19$ model is written as

$$\boldsymbol{M} = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \tag{A.165}$$

The moment space which results from multiplying transformation matrix with distribution function for $D3Q19$ model is

$$\begin{aligned}
\boldsymbol{m} = (&m_0 \;\; m_x \;\; m_y \;\; m_z \;\; m_{xx} \;\; m_{yy} \;\; m_{zz} \;\; m_{xy} \;\; m_{yz} \;\; m_{xz} \\
&m_{xyy} \;\; m_{xxz} \;\; m_{yyx} \;\; m_{yyz} \;\; m_{zzx} \;\; m_{zzy} \\
&m_{xxyy} \;\; m_{yyzz} \;\; m_{zzxx})^T
\end{aligned} \tag{A.166}$$

Multiplying the equilibrium distribution function $f^{eq,m}$ Eq. 3.3 with

transformation matrix $\boldsymbol{M}$ results in following equilibrium moments

$$\boldsymbol{m}^{eq} = \begin{pmatrix} \rho \\ \rho_0 v_x \\ \rho_0 v_y \\ \rho_0 v_z \\ \frac{1}{3}\rho + \rho_0 v_x^2 \\ \frac{1}{3}\rho + \rho_0 v_y^2 \\ \frac{1}{3}\rho + \rho_0 v_z^2 \rho_0 v_x v_y \\ \rho_0 v_y v_z \\ \rho_0 v_x v_z \\ \frac{1}{3}\rho_0 v_y \\ \frac{1}{3}\rho_0 v_z \\ \frac{1}{3}\rho_0 v_x \\ \frac{1}{3}\rho_0 v_z \\ \frac{1}{3}\rho_0 v_x \\ \frac{1}{3}\rho_0 v_y \\ \frac{1}{3}(\frac{1}{3}\rho + \rho_0 v_x^2 + \rho_0 v_y^2 - \frac{1}{2}\rho_0 v_z^2) \\ \frac{1}{3}(\frac{1}{3}\rho - \frac{1}{2}\rho_0 v_x^2 + \rho_0 v_z^2 + \rho_0 v_y^2) \\ \frac{1}{3}(\frac{1}{3}\rho + \rho_0 v_x^2 - \frac{1}{2}\rho_0 v_y^2 + \rho_0 v_z^2) \end{pmatrix}. \tag{A.167}$$

Here only first ten moments have a direct physical interpretation for the incompressible Navier-Stokes equations as pressure, velocity and shear stress.

**Multi-component Lattice Boltzmann Method** The non-zero entires of relaxation matrix $\boldsymbol{\Lambda}_k$ for multi-component lattice Boltzmann equation for species $k$ is set to

$$\begin{aligned} &\lambda_{2,2} = \lambda_{3.3} = \lambda_{4.4} = \lambda_k^{\mathcal{D}}, \\ &\lambda_{5,5} = \lambda_{6,6} = \lambda_{7,7} = \frac{\lambda_k^{\zeta} + \lambda^{\nu}}{2}, \\ &\lambda_{5,6} = \lambda_{5,7} = \lambda_{6,5} = \lambda_{6,7} = \lambda_{7,5} = \lambda_{7,6} = \frac{\lambda_k^{\zeta} - \lambda^{\nu}}{2}, \\ &\lambda_{8,8} = \lambda_{9,9} = \lambda_{10,10} = \lambda_k^{\nu}, \\ &\lambda_{i,i} = 1 \quad \forall i = 11 \ldots 19. \end{aligned} \tag{A.168}$$

## A.3 Configurations

This section contains example configuration for *STfun*and *varSys*.

```
variable = {
  {
     name = 'velmag',
     ncomponents = 1,
     vartype = 'operation',
     operation = {
       kind = 'magnitude',
       input_varname = {'velocity'}
     }
  },
  {
     name = 'momentum_mag',
     ncomponents = 3,
     vartype = 'operation',
     operation = {
       kind = 'multiplication',
       input_varname = {'density','velmag'}
     }
  }
}
```

**Listing A.1** Operation variable definition in configuration file

```
variable = {
  {
    name = 'press_out',
    ncomponents = 1,
    vartype = 'st_fun',
    st_fun = 1.0
  },
  {
    name = 'vel_in',
    ncomponents = 3,
    vartype = 'st_fun',
    st_fun = {
        predefined = 'combined',
        temporal = {kind = 'linear', from_time=0, to_time=0.25,
                    min_factor=0, max_factor=1.0},
        spatial = {1.0, 0, 0}
     }
  }
}

boundary_condition={
   ...
  { label = 'outflow',
    kind = 'press_dirichlet',
    pressure = 'press_out'
  },
  { label = 'inflow',
     kind = 'vel_dirichlet',
     velocity = 'vel_in'
  },
   ...
}
```

**Listing A.2** Configuration of space-time function in variable table

```
   constant
```

```
ref_press = 1.0
u_mean = 1.0

boundary_condition={
  ...
  { label = 'outflow',
    kind = 'press_dirichlet',
    pressure = ref_press
  },
  { label = 'inflow',
     kind = 'vel_dirichlet',
     velocity = {
       predefined = 'combined',
       temporal = {kind = 'linear', from_time=0, to_time=0.25,
                   min_factor=0, max_factor=1.0},
       spatial = {u_mean, 0, 0}
     }
  }
  ...
}

   space time Lua function
function force_fun(x,y,z,t)
  return {2.0*x*t, y, 0}    returns a vector
end

source = {
  force  = force_fun,
}
```

**Listing A.3** Configuration of space-time function as *anonymous* variable

```
boundary_condition ={
\dots
  {
    label='inflow'
    kind='dirichlet',
    velocity = 'vel_d1',
    pressure = 'press_d1'
  }
\dots
}

variable = {
  {
    name = 'surface_coupling',
    ncomponents = 4,
    vartype = 'st_fun',
    st_fun = {
      predefined = 'apesmate',
      input_varname = {'velocity', 'pressure'},
      domain_from = 'dom_1'
    }
  },
  {
    name='vel_d1',
    ncomponents = 3,
    vartype = 'operation',
    operation = {
      kind = 'extract',
      input_varname = {'surface_coupling'},
      input_varindex = {1,2,3},
```

```
31      }
32    },
33    {
34      name='press_d1',
35      ncomponents = 1,
36      vartype = 'operation',
37      operation = {
38        kind = 'extract',
39        input_varname = {'surface_coupling'},
40        input_varindex = {4},
41      }
42    }
43  }
```

**Listing A.4** "combine" and "extract" operation variable definition in configuration file

```
source = {
  force = {
    fun = force_fun,
    shape = {
      kind = 'canoND',
      object = { origin = {0,0,0}, vec = {{1,0.,0}, {0,1,0}} }
    }
  }
}
```

**Listing A.5** Configuration of space-time function variable with shape

```
source = {
  force = 'electric_force'
}
variable = {
  {
    name = 'electric_force'
    ncomponents = 3,
    vartype = 'st_fun',
    st_fun = {
      predefined = 'apesmate',
      input_varname = {'electric_field'},
      domain_from = 'potential_dom'
    }
}
```

**Listing A.6** Configuration of space-time function variable for coupling

```
    simulation time control
sim_control = {
  time_control = {
     max = 1.0   maximum simulation time in second
     interval = 0.1      interval to check status of the simulation in second
  }
}

   Logic to distribute all domain on all process
   Default: false
share_domain = false
```

```
13     Logic to define nProc is fraction or integer
14     Default: true
15  nproc_is_frac = true
16
17  domain_object ={
18    {
19      label='dilute_mc',      domain name
20      solver='musubi',        solver  name
21      filename = 'multicomponent_dilute_musubi.lua',  domain configuration filename
22      nproc_frac = 2/6        fraction of number of process to use for this domain
23    },
24    {
25      label='conc_mc',
26      solver='musubi',
27      filename = 'multicomponent_conc_musubi.lua',
28      nproc_frac = 2/6
29    },
30    {
31      label='dilute_potential',
32      solver='musubi',
33      filename = 'potential_dilute_musubi.lua',
34      nproc_frac = 1/6
35    },
36    {
37      label='conc_potential',
38      solver='musubi',
39      filename = 'potential_conc_musubi.lua',
40      nproc_frac = 1/6
41    }
42  }
```

**Listing A.7** *APESmate* domains configuration

# Bibliography


[1] . 2011. URL: https://www.siemens.com/press/pi/IIS201106205e.

[2] Pierre Andries, Kazuo Aoki, and Benoit Perthame. "A Consistent BGK-Type Model for Gas Mixtures". English. In: *Journal of Statistical Physics* 106.5-6 (2002), pp. 993–1018. ISSN: 0022-4715. DOI: 10.1023/A:1014033703134. URL: http://dx.doi.org/10.1023/A%3A1014033703134.

[3] S. Arcidiacono et al. "Lattice Boltzmann model for the simulation of multicomponent mixtures". In: *Physical Review E* 76.4 (Oct. 2007), pp. 1–11. ISSN: 1539-3755. DOI: 10.1103/PhysRevE.76.046703. URL: http://link.aps.org/doi/10.1103/PhysRevE.76.046703.

[4] R. Aris. "On the Dispersion of a Solute in a Fluid Flowing through a Tube". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 235.1200 (1956), pp. 67–77. DOI: 10.1098/rspa.1956.0065. URL: http://rspa.royalsocietypublishing.org/content/235/1200/67.abstract.

[5] Pietro Asinari. "Multi-species Lattice Boltzmann Models and Practical Examples". In: *Methods* (2008).

[6] Pietro Asinari. "Multiple-relaxation-time lattice Boltzmann scheme for homogeneous mixture flows with external force". In: *Physical Review E* 77.5 (May 2008), pp. 1–13. ISSN: 1539-3755. DOI: 10.1103/PhysRevE.77.056706. URL: http://link.aps.org/doi/10.1103/PhysRevE.77.056706.

[7] Pietro Asinari. "Semi-implicit-linearized multiple-relaxation-time formulation of lattice Boltzmann schemes for mixture modeling". In: *Physical Review E* 73.5 (May 2006), pp. 1–24. ISSN: 1539-3755. DOI: 10.1103/PhysRevE.73.056705. URL: http://link.aps.org/doi/10.1103/PhysRevE.73.056705.

[8] V. Baltazar, G.B Harris, and C.W White. "The selective recovery and concentration of sulphuric acid by electrodialysis". In: *Hydrometallurgy* 30.1-3 (1992), pp. 463–481. ISSN: 0304386X. DOI: 10.1016/0304-386X(92)90100-E.

[9] Gordon S. Beavers and Daniel D. Joseph. "Boundary conditions at a naturally permeable wall". In: *Journal of Fluid Mechanics* 30.01 (1967), p. 197. ISSN: 0022-1120. DOI: 10.1017/S0022112067001375.

[10] Sam Bennett. "A Lattice Boltzmann Model for Diffusion of Binary Gas Mixtures". PhD thesis. University of Cambridge, 2010.

[11] Sam Bennett, Pietro Asinari, and Paul J. Dellar. "A lattice Boltzmann model for diffusion of binary gas mixtures that includes diffusion slip". In: *International Journal for Numerical Methods in Fluids* 69 (1 2012), pp. 171–189. ISSN: 1097-0363. DOI: 10.1002/fld.2549. URL: http://dx.doi.org/10.1002/fld.2549.

[12] J. Bernsdorf et al. "Lattice Boltzmann Simulation and Visualisation of Adsorption Processes in Complex Geometries". In: *Computational Science — ICCS 2003*. Ed. by PeterM.A. Sloot et al. Vol. 2657. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 1054–1061. ISBN: 978-3-540-40194-0. DOI: 10.1007/3-540-44860-8\textunderscore109. URL: http://dx.doi.org/10.1007/3-540-44860-8\textunderscore109.

[13] P. L. Bhatnagar, E. P. Gross, and M. Krook. "A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems". In: *Physical review* 94 (3 May 1954), pp. 511–525. URL: http://link.aps.org/doi/10.1103/PhysRev.94.511.

[14] R. B. Bird, W. E. Stewart, and E. N. Lightfoot. *Transport Phenomena*. John Whiley & Sons, Jan. 1976.

[15] Mohamed Bouzidi, Mouaouia Firdaouss, and Pierre Lallemand. "Momentum transfer of a Boltzmann-lattice fluid with boundaries". In: *Physics of Fluids* 13.11 (2001), p. 3452. ISSN: 10706631. DOI: 10.1063/1.1399290. URL: http://scitation.aip.org/content/aip/journal/pof2/13/11/10.1063/1.1399290.

[16] R. Carty and T. Schrodt. "Concentration Profiles in Ternary Gaseous Diffusion". In: *Industrial and Engineering Chemistry Fundamentals* 14.3 (1975), pp. 276–278. ISSN: 01964313. DOI: 10.1021/i160055a025.

[17] Cornelis Ronald Visser. "Electrodialytic Recovery of Acids and Bases: Multicomponent Mass Transfer Description". PhD thesis. Groningen: Rijksuniversiteit Groningen, 2001.

[18] A R Da Costa, A G Fane, and D E Wiley. "Spacer characterization and pressure drop modelling in spacer-filled channels for ultrafiltration". In: *Journal of Membrane Science* 87 (1994), pp. 79–98.

[19] Dominique D'Humières et al. "Multiple-relaxation-time lattice Boltzmann models in three dimensions." In: *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences* 360.1792 (Mar. 2002), pp. 437–51. ISSN: 1364-503X. DOI: 10.1098/rsta.2001.0955. URL: http://www.ncbi.nlm.nih.gov/pubmed/16214687.

[20] Paul J. Dellar. "An interpretation and derivation of the lattice Boltzmann method using Strang splitting". In: *Computers and Mathematics with Applications* 65.2 (2013), pp. 129–141. ISSN: 08981221. DOI: 10.1016/j.camwa.2011.08.047. URL: http://dx.doi.org/10.1016/j.camwa.2011.08.047.

[21] Paul Dellar. "Bulk and shear viscosities in lattice Boltzmann equations". In: *Physical Review E* 64.3 (2001), p. 031203. ISSN: 1063-651X. DOI: 10.1103/PhysRevE.64.031203. URL: http://link.aps.org/doi/10.1103/PhysRevE.64.031203.

[22] Encyclopedia of Desalination and Water Resources (DESWARE). *Energy Requirements of Desalination Processes*. http://www.desware.net/desa4.aspx. Accessed: 2016-08-29.

[23] Piotr Długołęcki et al. "On the resistances of membrane, diffusion boundary layer and double layer in ion exchange membrane transport". In: *Journal of Membrane Science* 349.1-2 (2010), pp. 369–379. ISSN: 03767388. DOI: 10.1016/j.memsci.2009.11.069.

[24] M. FIDALEO and M. MORESI. "Optimal strategy to model the electrodialytic recovery of a strong electrolyte". In: *Journal Of Membrane Science* 260.1-2 (2005), pp. 90–111. ISSN: 0376-7388. DOI: 10.1016/j.memsci.2005.01.048.

[25] Vera Sumberova Siernon Kuindersma Hans Wesselingh Gerrit Kraaijeveld. "Modelling electrodialysis using the Maxwell-Stefan description". In: *The Chemical Engineering Journal* 57 (1995), pp. 163–176.

[26] Dominik Geuß. "Investigation of Mixture Modeling for the Lattice Boltzmann Method". In: *Master Thesis* 10 (2008).

[27] Zhaoli Guo, Baochang Shi, and Chuguang Zheng. "An extrapolation method for boundary conditions in lattice Boltzmann method". In: *Physics of Fluids* 14.May (2002), pp. 10–14. DOI: 10.1063/1.1471914.

[28] Zhaoli Guo, Chuguang Zheng, and Baochang Shi. "Discrete lattice effects on the forcing term in the lattice Boltzmann method". In: *Physical Review E* 65.4 (Apr. 2002), pp. 1–6. ISSN: 1063-651X. DOI: 10.1103/PhysRevE.65.046308. URL: http://link.aps.org/doi/10.1103/PhysRevE.65.046308.

[29] Daniel F. Harlacher et al. "Dynamic Load Balancing for Unstructured Meshes on Space-Filling Curves". In: *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*. 2012, pp. 1661–1669.

[30] Daniel F. Harlacher et al. "Tree Based Voxelization of STL Data". In: *Journal of Computational Science* (2013), pp. 81–92. ISSN: 18777503. DOI: 10.1007/978-3-642-22244-3\textunderscore6.

[31] Manuel Hasert. "Multi-scale lattice Boltzmann Simulations on Distributed Octrees". PhD thesis. RWTH Aachen University, 2014.

[32] Manuel Hasert et al. "Complex fluid simulations with the parallel tree-based Lattice Boltzmann solver Musubi". In: *Journal of Computational Science* (2013). ISSN: 18777503. DOI: 10.1016/j.jocs.2013.11.001.

[33] X He. "Lattice Boltzmann simulation of electrochemical systems". In: *Computer Physics Communications* 129.1-3 (July 2000), pp. 158–166. ISSN: 00104655. DOI: 10.1016/S0010-4655(00)00103-X. URL: http://linkinghub.elsevier.com/retrieve/pii/S001046550000103X.

[34] Xiaoyi He, Shiyi Chen, and Gary D. Doolen. "A Novel Thermal Model for the Lattice Boltzmann Method in Incompressible Limit". In: *Journal of Computational Physics* 146.1 (1998), pp. 282–300. ISSN: 00219991. DOI: 10.1006/jcph.1998.6057.

[35] Xiaoyi He and Li-shi Luo. "Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation". In: *Physical Review E* 56.6 (1997), pp. 6811–6817. ISSN: 1063-651X. DOI: 10.1103/PhysRevE.56.6811.

[36] E Hückel and P Debye. "The theory of electrolytes: I. lowering of freezing point and related phenomena". In: *Phys. Z* 24 (1923), pp. 185–206.

[37] Roberto Ierusalimschy, Luiz Henrique De Figueiredo, and Waldemar Celes. *Lua 5.1 Reference Manual*. Roberto Ierusalimschy, Aug. 2006. ISBN: 8590379833.

[38] Pietro Asinari Jens Zudrop Sabine Roller. "Lattice boltzmann scheme for electrolytes by an extended maxwell-stefan approach". In: *Phys. Rev. E* (2014). URL: http://link.aps.org/doi/10.1103/PhysRevE.89.053310.

[39] Matthias Johannink et al. "Predictive pressure drop models for membrane channels with non-woven and woven spacers". In: *Desalination* 376 (2015), pp. 41 –54. ISSN: 0011-9164. DOI: https://doi.org/10.1016/j.desal.2015.07.024. URL: http://www.sciencedirect.com/science/article/pii/S0011916415300321.

[40] Abhijit S Joshi et al. "Lattice Boltzmann method for continuum, multi-component mass diffusion in complex 2D geometries". In: *Journal of Physics D: Applied Physics* 40.9 (May 2007), pp. 2961–2971. ISSN: 0022-3727. DOI: 10.1088/0022-3727/40/9/044. URL: http://stacks.iop.org/0022-3727/40/i=9/a=044?key=crossref.02c1b8c688f524a0f774a4bc0b9c25e8.

[41] M Junk, a Klar, and L Luo. "Asymptotic analysis of the lattice Boltzmann equation". In: *Journal of Computational Physics* 210.2 (Dec. 2005), pp. 676–704. ISSN: 00219991. DOI: 10.1016/j.jcp.2005.05.003. URL: http://linkinghub.elsevier.com/retrieve/pii/S0021999105002573.

[42] Michael Junk and Zhaoxia Yang. "Asymptotic Analysis of Lattice Boltzmann Boundary Conditions". In: *Journal of Statistical Physics* 121 (1-2 Oct. 2005), pp. 3–35. ISSN: 0022-4715. DOI: 10.1007/s10955-005-8321-2. URL: http://www.springerlink.com/index/10.1007/s10955-005-8321-2.

[43] Michael Junk and Zhaoxia Yang. "Asymptotic Analysis of Lattice Boltzmann Outflow Treatments." In: *Communications in Computational Physics* x.x (2011), pp. 1–11. URL: http://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:Asymptotic+Analysis+of+Lattice+Boltzmann+Outflow+Treatments\#1.

[44] Michael Junk and Zhaoxia Yang. "One-point boundary condition for the lattice Boltzmann method". In: *Physical Review E* 72.6 (Dec. 2005), pp. 1–8. ISSN: 1539-3755. DOI: 10.1103/PhysRevE.72.066701. URL: http://link.aps.org/doi/10.1103/PhysRevE.72.066701.

[45] Michael Junk and Zhaoxia Yang. “Pressure boundary condition for the lattice Boltzmann method”. In: *Computers & Mathematics with Applications* 58 (5 Sept. 2009), pp. 922–929. ISSN: 08981221. DOI: 10.1016/j.camwa.2009.02.006. URL: http://linkinghub.elsevier.com/retrieve/pii/S0898122109000972.

[46] Woo Sik Kim, Joong Kon Park, and Ho Nam Chang. “Mass transfer in a three-dimensional net-type turbulence promoter”. In: *International Journal of Heat and Mass Transfer* 30.6 (1987), pp. 1183 –1192. ISSN: 0017-9310. DOI: https://doi.org/10.1016/0017-9310(87)90047-0. URL: http://www.sciencedirect.com/science/article/pii/0017931087900470.

[47] Harald Klimach. *Advanced Options in Tables and Universal Scripting*. https://bitbucket.org/haraldkl/aotus/wiki/Home. [Online; accessed 14-September-2011]. 2011.

[48] Harald Klimach et al. “Distributed octree mesh infrastructure for flow simulations”. In: *ECCOMAS 2012 - European Congress on Computational Methods in Applied Sciences and Engineering, e-Book Full Papers*. Ed. by J Eberhardsteiner. Vienna, Austria, 2012.

[49] A. Knoll. *A Survey of Octree Volume Rendering Methods*. 2008.

[50] C P Koutsou, S G Yiantsios, and A J Karabelas. “A numerical and experimental study of mass transfer in spacer-filled channels : Effects of spacer geometrical characteristics and Schmidt number”. In: *Journal of Membrane Science* 326 (2009), pp. 234–251. DOI: 10.1016/j.memsci.2008.10.007.

[51] C. P. Koutsou, S. G. Yiantsios, and A. J. Karabelas. “Direct numerical simulation of flow in spacer-filled channels: Effect of spacer geometrical characteristics”. In: *Journal of Membrane Science* 291.1-2 (2007), pp. 53–69. ISSN: 03767388. DOI: 10.1016/j.memsci.2006.12.032.

[52] T. Krüger, F. Varnik, and D. Raabe. “Second-order convergence of the deviatoric stress tensor in the standard Bhatnagar-Gross-Krook lattice Boltzmann method”. In: *Physical Review E* 82.2 (2010), pp. 1–4. ISSN: 1539-3755. DOI: 10.1103/PhysRevE.82.025701. URL: http://link.aps.org/doi/10.1103/PhysRevE.82.025701.

[53] Verena Krupp et al. “Efficient Coupling of Fluid and Acoustic Interaction on Massive Parallel Systems”. In: *Sustained Simulation Performance 2016*. Ed. by Michael M. Resch et al. Springer International Publishing, 2016, pp. 61–81. DOI: 10.1007/978-3-3199-46735-1_6.

[54] L.D. LANDAU and E.M. LIFSHITZ. “CHAPTER II - VISCOUS FLUIDS”. In: *Fluid Mechanics (Second Edition)*. Ed. by L.D. LANDAU and E.M. LIFSHITZ. Second Edition. Pergamon, 1987, pp. 44 – 94. ISBN: 978-0-08-033933-7. DOI: https://doi.org/10.1016/B978-0-08-033933-7.50010-6. URL: http://www.sciencedirect.com/science/article/pii/B9780080339337500106.

[55] Anthony J. C. Ladd. “Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation”. In: *Journal of Fluid Mechanics* 271 (July 1994), pp. 285–309. ISSN: 1469-7645. DOI: 10.1017/S0022112094001771. URL: http://journals.cambridge.org/article_S0022112094001771.

[56] Pierre Lallemand and Li-Shi Luo. “Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability”. In: *Phys. Rev. E* 61 (6 2000), pp. 6546–6562. DOI: 10.1103/PhysRevE.61.6546. URL: http://link.aps.org/doi/10.1103/PhysRevE.61.6546.

[57] Hong-Joo Lee et al. “Designing of an electrodialysis desalination plant”. In: *Desalination* 142.3 (2002), pp. 267–286. ISSN: 0011-9164. DOI: 10.1016/S0011-9164(02)00208-4.

[58] Dongqing Li. “Electro-viscous effects on pressure-driven liquid flow in microchannels”. In: *Colloids and Surfaces A: Physicochemical and Engineering Aspects* 195 (2001), pp. 35 –57.

[59] Yu-ling Li and Kuo-lun Tung. “CFD simulation of fluid flow through spacer-filled membrane module: selecting suitable cell types for periodic boundary conditions”. In: *Desalination* 233.August 2007 (2008), pp. 351–358. DOI: 10.1016/j.desal.0000.00.000.

[60] Xin Liu, Thijs J.H. Vlugt, and André Bardow. “Maxwell-Stefan diffusivities in liquid mixtures: Using molecular dynamics for testing model predictions”. In: *Fluid Phase Equilibria* 301.1 (Feb. 2011), pp. 110–117. ISSN: 03783812. DOI: 10.1016/j.fluid.2010.11.019. URL: http://linkinghub.elsevier.com/retrieve/pii/S0378381210005856.

[61] Kang Luo et al. “A lattice Boltzmann method for electric field-space charge coupled problems”. In: *Electrostatics Joint Conference*. 2016, pp. 1–11.

[62] Kang Luo et al. “International Journal of Heat and Mass Transfer Lattice Boltzmann modelling of electro-thermo-convection in a planar layer of dielectric liquid subjected to unipolar injection and thermal gradient”. In: *International Journal of Heat and Mass Transfer* 103 (2016), pp. 832–846.

[63] Kannan Masilamani and Harald KIimach. *Seeder Mesh Generator*. `https://bitbucket.org/apesteam/seeder`. [Online; accessed 01-August-2013]. 2013.

[64] Kannan Masilamani, Harald Klimach, and Sabine Roller. “Highly Efficient Integrated Simulation of Electro-Membrane Processes for Desalination of Sea Water”. In: *High Performance Computing in Science and Engineering ’13*. Ed. by Wolfgang E. Nagel, Dietmar B. Kröner, and Michael M. Resch. Springer Cham Heidelberg New York Dordrecht London, 2013, pp. 493–508. ISBN: 978-3-319-02164-5. DOI: `http://dx.doi.org/10.1007/978-3-319-02165-2`.

[65] Kannan Masilamani et al. “Highly Efficient Integrated Simulation of Electro-Membrane Processes for Desalination of Sea Water”. In: *High Performance Computing in Science and Engineering ’14*. Ed. by Wolfgang E. Nagel, Dietmar B. Kröner, and Michael M. Resch. Springer Cham Heidelberg New York Dordrecht London, 2014.

[66] Michael McCracken and John Abraham. “Lattice Boltzmann methods for binary mixtures with different molecular weights”. In: *Physical Review E* 71.4 (Apr. 2005). ISSN: 1539-3755. DOI: `10.1103/PhysRevE.71.046704`. URL: `http://link.aps.org/doi/10.1103/PhysRevE.71.046704`.

[67] FrankJ. Millero, J. Ricco, and DonaldR. Schreiber. “PVT properties of concentrated aqueous electrolytes. II. Compressibilities and apparent molar compressibilities of aqueous NaCl, Na2SO4, MgCl2, and MgSO4 from dilute solution to saturation and from 0 to 50C”. In: *Journal of Solution Chemistry* 11.10 (1982), pp. 671–686. ISSN: 0095-9782. DOI: `10.1007/BF00645335`. URL: `http://dx.doi.org/10.1007/BF00645335`.

[68] Charles W. Monroe and John Newman. "Onsager Reciprocal Relations for Stefan−Maxwell Diffusion". In: *Industrial & Engineering Chemistry Research* 45.15 (2006), pp. 5361–5367. ISSN: 0888-5885. DOI: 10.1021/ie051061e. URL: http://pubs.acs.org/doi/pdf/10.1021/ie051061e.

[69] G. M. Morton. *A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing*. Tech. rep. Ottawa, Canada: IBM Ltd, 1966.

[70] K. Mosthaf et al. "A coupling concept for two-phase compositional porous-medium and single-phase compositional free flow". In: *Water Resources Research* 47.10 (2011), n/a. ISSN: 00431397. DOI: 10.1029/2011WR010685.

[71] C.-D. Munz et al. "Divergence Correction Techniques for Maxwell Solvers Based on a Hyperbolic Model". In: *Journal of Computational Physics* 161.2 (July 2000), pp. 484–511. ISSN: 00219991. DOI: 10.1006/jcph.2000.6507. URL: http://linkinghub.elsevier.com/retrieve/pii/S002199910096507 0http://www.sciencedirect.com/science/article/pii/S0021999100965070.

[72] John S. Newman and Karen E. Thomas-Alyea. *Electrochemical systems*. 3rd ed. Electrochemical Society series. Hoboken, NJ: Wiley-Interscience, 2004.

[73] C. Picioreanu, J.S. Vrouwenvelder, and M.C.M. van Loosdrecht. "Three-dimensional modeling of biofouling and fluid dynamics in feed spacer channels of membrane devices". In: *Journal of Membrane Science* 345.1-2 (2009), pp. 340–354. ISSN: 03767388. DOI: 10.1016/j.memsci.2009.09.024. URL: http://linkinghub.elsevier.com/retrieve/pii/S0376738809006802.

[74] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000. DOI: 10.1017/CBO9780511840531.

[75] Neda Ebrahimi Pour. "Investigating the effect of spacer geometry in electrodialysis process for seawater desalination". PhD thesis. Univeristy of Siegen, 2016.

[76] William H. Press et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. New York, NY, USA: Cambridge University Press, 2007. ISBN: 0521880688, 9780521880688.

[77] Process Systems Enterprise. *gPROMS, www.psenterprise.com/gproms, rev. 12.06.2014*. 1997-2009. URL: www.psenterprise.com/gproms.

[78] Steven Psaltis. “Multicomponent Charge Transport in Electrolyte Solutions”. PhD thesis. Queensland University of Technology, 2012.

[79] Jiaxing Qi. “Efficient Lattice Boltzmann Simulations on Large Scale High Performance Computing Systems”. PhD thesis. RWTH Aachen University, 2017.

[80] Sabine Roller et al. “An Adaptable Simulation Framework Based on a Linearized Octree”. In: *High Performance Computing on Vector Systems 2011*. Ed. by Michael RESCH et al. Springer Berlin Heidelberg, 2012, pp. 93–105. ISBN: 978-3-642-22243-6. DOI: 10.1007/978-3-642-22244-3\textunderscore7.

[81] Mohtada Sadrzadeh and Toraj Mohammadi. “Treatment of sea water using electrodialysis: Current efficiency evaluation”. In: *Desalination* 249.1 (2009), pp. 279–285. ISSN: 00119164. DOI: 10.1016/j.desal.2008.10.029. URL: http://dx.doi.org/10.1016/j.desal.2008.10.029.

[82] K Sandeep. “Flow visualization through spacer filled channels by computational fluid dynamics I . Pressure drop and shear rate calculations for flat sheet geometry”. In: *Journal of Membrane Science* 193.May (2001), pp. 69–84.

[83] Toshikatsu Sata. *Ion exchange membranes: Preparation, characterization, modification and application*. Cambridge: Royal Society of Chemistry, 2004. ISBN: 0854045902.

[84] J Schwinge, D E Wiley, and D F Fletcher. “A CFD study of unsteady flow in narrow spacer-filled channels for spiral-wound membrane modules”. In: *Desalination* 146.2002 (2002), pp. 195–201.

[85] M. Shakaib, S. M F Hasani, and M. Mahmood. “CFD modeling for flow and mass transfer in spacer-obstructed membrane feed channels”. In: *Journal of Membrane Science* 326.2 (2009), pp. 270–284. ISSN: 03767388. DOI: 10.1016/j.memsci.2008.09.052.

[86] Xiaowen Shan and Gary Doolen. “Multicomponent lattice-Boltzmann model with interparticle interaction”. In: *Journal of Statistical Physics* 81.1-2 (Oct. 1995), pp. 379–393. ISSN: 0022-4715. DOI: 10.1007/BF02179985. URL: http://www.springerlink.com/index/10.1007/BF02179985.

[87] Baochang Shi and Zhaoli Guo. “Lattice Boltzmann model for nonlinear convection-diffusion equations”. In: *Phys. Rev. E* 79 (1 2009), p. 016701. DOI: 10.1103/PhysRevE.79.016701. URL: https://link.aps.org/doi/10.1103/PhysRevE.79.016701.

[88] Baochang Shi et al. "A new scheme for source term in LBGK model for convectiondiffusion equation". In: *Computers & Mathematics with Applications* 55.7 (2008). Mesoscopic Methods in Engineering and Science, pp. 1568 –1575. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2007.08.016. URL: http://www.sciencedirect.com/science/article/pii/S0898122107006220.

[89] Spiegler, K.S., Laird, A.D.K. *Principles of Desalination*. 2nd edition. Academic Press, New York, 1980. ISBN: 978-0-12-395660-6.

[90] H. Strathmann. "Electrodialysis, a mature technology with a multitude of new applications". In: *Desalination* 264.3 (2010), pp. 268–288. ISSN: 0011-9164. DOI: 10.1016/j.desal.2010.04.069.

[91] Heiner Strathmann. "Assessment of Electrodialysis Water Desalination Process Costs". In: *Proceedings of the International Conference on Desalination Costing, Lemassol, Cyprus, December 6-8, 2004* (2004), pp. 32–54.

[92] Heiner Strathmann, Andrej Grabowski, and Gerhart Eigenberger. "Ion-Exchange Membranes in the Chemical Process Industry". In: *Industrial & Engineering Chemistry Research* 52.31 (2013), pp. 10364–10379. ISSN: 0888-5885. DOI: 10.1021/ie4002102. URL: http://pubs.acs.org/doi/abs/10.1021/ie4002102.

[93] Yoshinobu Tanaka. *Ion Exchange Membranes: Fundamentals and Applications*. 2nd ed. Membrane Science and Technology. Elsevier, 2015. ISBN: 9780444633194.

[94] G. I. Taylor and a. E. Green. "Mechanism of the Production of Small Eddies from Large Ones". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 158.895 (Feb. 1937), pp. 499–521. ISSN: 1364-5021. DOI: 10.1098/rspa.1937.0036. URL: http://rspa.royalsocietypublishing.org/cgi/doi/10.1098/rspa.1937.0036.

[95] G. Taylor. "Dispersion of Soluble Matter in Solvent Flowing Slowly through a Tube". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 219.1137 (1953), pp. 186–203. DOI: 10.1098/rspa.1953.0139. URL: http://rspa.royalsocietypublishing.org/content/219/1137/186.abstract.

[96] Ross Taylor and R. Krishna. *Multicomponent mass transfer*. New York: Wiley, 1993. ISBN: 9780471574170.

[97] Blender Documentation Team. *Blender - a 3D modeling and rendering package*. 2016. URL: https://docs.blender.org/manual/en/dev/.

[98] *VDI-Wärmeatlas*. 10th ed. Berlin [u.a.]: Springer, 2006. ISBN: 978-3-540-29646-1. URL: http://www.worldcat.org/oclc/180047763.

[99] Fernando Valero and Ramón Arbós. "Desalination of brackish river water using Electrodialysis Reversal (EDR): Control of the THMs formation in the Barcelona (NE Spain) area". In: *Desalination* 253.1-3 (2010), pp. 170–174. ISSN: 0011-9164. DOI: 10.1016/j.desal.2009.11.011. URL: http://www.sciencedirect.com/science/article/pii/S0011916409012971.

[100] Chapman Thomas Woodring. *The transport properties of concentrated electrolytic solutions*. University Microfilms, 1969.

[101] Xuguang Yang et al. "A Coupled Lattice Boltzmann Method to Solve Nernst Planck Model for Simulating Electro-osmotic Flows". In: *Journal of Scientific Computing* 61 (2014), pp. 222–238. DOI: 10.1007/s10915-014-9820-6.

[102] Thomas Zeiser, Georg Hager, and Gerhard Wellein. "Benchmark Analysis and Application Results for Lattice Boltzmann Simulations on NEC SX Vector and Intel Nehalem Systems." In: *Parallel Processing Letters* 19 (Dec. 2009), pp. 491–511.

[103] Jens Zudrop. "Efficient numerical methods for fluid- and electrodynamics on massively parallel systems". PhD thesis. RWTH Aachen University, 2015.

[104] Jens Zudrop et al. "A robust lattice Boltzmann method for parallel simulations of multicomponent flows in complex geometries". In: *Computers and Fluids* 153 (2017), pp. 20–33. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2017.04.021. URL: http://www.sciencedirect.com/science/article/pii/S0045793017301482.

# List of Figures

# List of Tables

# List of Algorithms

Simulation Techniques in Siegen
Vol 4

This thesis presents the development of a scalable coupled simulation framework to simulate the interactions of different physical phenomena.
It employs rigorous models and develops a numerical strategy for coupled simulations implemented in the APESmate tool.
This coupling strategy maintains good scalability on large, distributed computing systems and ensures accurate interpolation between meshes.
APESmate enables multiphysics simulations with any physics supported by solvers in the APES framework.
One specific multiphysics application this thesis discusses is the electrodialysis process. Here, ions in a multicomponent flow are driven apart by an electric field. To this end, the presented work implements a Maxwell-Stefan model in the Lattice-Boltzmann solver Musubi and investigates different geometrical designs for electrodialysis.

***Kannan Masilamani*** studied computational engineering at the Friedrich-Alexander-University of Erlangen-Nuremberg.
His master thesis was dedicated to particulate transport in Lattice-Boltzmann.
He worked for Siemens in the HISEEM project, funded by BMBF, to simulate electrodialysis processes for seawater desalination. This research is the central part of the present doctoral thesis and earned him a PRACE award.
After the HISEEM project, he worked in research and teaching at the chair for Simulation Techniques and Scientific Computing of the University Siegen. In July 2020, he assumed a research position at DLR in Dresden. Kannan Masilamani is a main developer of the open-source Lattice-Boltzmann solver Musubi.

The series ***Simulation Techniques in Siegen*** presents contributions to the field of scientific computing with a focus on the utilization of large-scale computing systems for highly resolved simulations. Applications, as well as numerical methods and their efficient implementation on modern supercomputers, are investigated and described.

ISBN: 978-3-96182-077-1

universi
UNIVERSITÄTSVERLAG SIEGEN