Parallel Reservoir Simulations with Sparse Grid Techniques and Applications to Wormhole Propagation

Dissertation by

Yuanqing Wu

In Partial Fulfillment of the Requirements For the Degree of

Doctor of Philosophy

King Abdullah University of Science and Technology, Thuwal, Kingdom of Saudi Arabia

October 2015

Yuanqing Wu

All rights reserved
The dissertation of Yuanqing Wu is approved by the examination committee.

Committee Chairperson: Shuyu Sun
Committee Members: David Keyes, Omar Knio, Yu-Shu Wu, Xiao-Chuan Cai
ABSTRACT

Parallel Reservoir Simulations with Sparse Grid Techniques and Applications to Wormhole Propagation

Yuanqing Wu

In this work, two topics in reservoir simulations are discussed. The first topic is the two-phase compositional flow simulation in a hydrocarbon reservoir. The major obstacle that impedes the applicability of the simulation code is the long run time of the simulation procedure, and thus speeding up the simulation code is necessary. Two means are demonstrated to address the problem: parallelism in physical space and the application of sparse grids in parameter space. The parallel code can gain satisfactory scalability, and the sparse grids can remove the bottleneck of flash calculations. Instead of carrying out the flash calculation in each time step of the simulation, a sparse grid approximation of all possible results of the flash calculation is generated before the simulation. Then the constructed surrogate model is evaluated to approximate the flash calculation results during the simulation. The second topic is the matrix acidization simulation in carbonate reservoirs. In this work, different from the traditional simulation technique relying on the Darcy framework, we propose a new framework called Darcy-Brinkman-Forchheimer framework to simulate matrix acidization. Furthermore, to process the large quantity of cells in the simulation grid and shorten the long simulation time of the traditional serial code, standard domain-based parallelism is employed, using the Hypre multigrid library. In addition to that, a new technique called “experimenting field approach” to set coefficients in the model equations is introduced. In the dissolution experiments, different configurations of matrix acidization and a series of properties simulated by both frameworks are compared. The scalability of the parallel code is also evaluated, and good scalability can be achieved. Finally, a mixed finite element-based fully conservative method is proposed for the matrix acidization simulation based on the Darcy-Forchheimer framework. In this method, the mixed finite element methods are used not only for the Darcy-Forchheimer flow equations but also for the solute transport equation by introducing an auxiliary flux variable to guarantee full mass conservation. Numerical results are also given to verify theoretical analysis and effectiveness of the proposed scheme.
Firstly, I would like to thank King Abdullah to give me the precious chance to pursue my PhD degree in KAUST.

Then, I would like to express the deepest appreciation to my committee chair Professor Shuyu Sun, who has shown the attitude and the substance of a genius: he continually and persuasively conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his supervision and constant help this dissertation would not have been possible.

In addition, I would like to thank my committee members, Professor David Keyes, Professor Omar Knio, Professor Yu-Shu Wu and Professor Xiao-Chuan Cai, who always instruct me to find out the insignificances of my work, make constructive suggestions and spend much time on improving the dissertation.

Finally, I would like to thank all of the colleagues and friends in CTPL, AMCS and KAUST. Your talents, wisoms and supports contribute to the finish of the dissertation.
TABLE OF CONTENTS

EXAMINATION COMMITTEE APPROVALS FORM ........................................ 2
ABSTRACT .................................................................................................. 3
ACKNOWLEDGEMENTS ............................................................................ 4
TABLE OF CONTENTS ............................................................................. 5
LIST OF ABBREVIATIONS ........................................................................ 7
LIST OF FIGURES .................................................................................... 8
LIST OF TABLES ...................................................................................... 11
1. Introduction and Literature Review ................................................... 12
  1.1 Multi-phase Flow ............................................................................. 12
  1.2 Flash Calculation ............................................................................ 15
  1.3 Wormhole ....................................................................................... 18
  1.4 Parallelism in Reservoir Simulation .............................................. 24
2. Parallel Simulation of Two-phase Compositional Flows .................... 28
  2.1 Basic Equations .............................................................................. 28
  2.2 Discretization ................................................................................ 32
  2.3 Parallelization ............................................................................... 37
  2.4 Performance Evaluation ............................................................... 40
3. Two-phase Flash Calculations ............................................................ 44
  3.1 Basic Concepts .............................................................................. 44
  3.2 Evaluation of Parameters by Peng-Robinson Equation of State ....... 47
  3.3 Computation of Viscosity ............................................................... 52
  3.4 Computation of Partial Molar Volume .......................................... 55
  3.5 Computation of Two-phase Isothermal Compressibility .............. 56
  3.6 Successive Substitution Technique ............................................... 61
  3.7 Stability Analysis ........................................................................... 64
4. Application of Sparse Grids ................................................................. 67
  4.1 Sparse Grid Interpolation ............................................................... 67
  4.2 Sparse Grid Surrogate Model ......................................................... 72
  4.3 Test Environment ........................................................................... 75
  4.4 Surrogate Performance ................................................................. 77
  4.5 Extensibility to Three Dimensions ............................................... 83
  4.6 Performance Improvement of the Two-phase Compositional Flow Simulation ... 84
  4.7 Numerical Results ......................................................................... 87
5. 2D Parallel Simulation of Wormhole Propagation ............................ 90
  5.1 Basic Equations ............................................................................. 90
  5.2 Discretization and Experimenting Field Approach ....................... 95
  5.3 Parallelization .............................................................................. 103
  5.4 Verification .................................................................................. 108
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIP</td>
<td>Binary Interaction Parameters</td>
</tr>
<tr>
<td>CCFD</td>
<td>Cell-Centered Finite Difference</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DBF</td>
<td>Darcy-Brinkman-Forchheimer</td>
</tr>
<tr>
<td>DF</td>
<td>Darcy-Forchheimer</td>
</tr>
<tr>
<td>DS</td>
<td>Dimensional Splitting</td>
</tr>
<tr>
<td>FDM</td>
<td>Finite Difference Method</td>
</tr>
<tr>
<td>FEM</td>
<td>Finite Element Method</td>
</tr>
<tr>
<td>GMRES</td>
<td>Generalized Minimal RESidual</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>HCl</td>
<td>Hydrochloric acid</td>
</tr>
<tr>
<td>IMPEC</td>
<td>IMplicit Pressure and Explicit Concentration method</td>
</tr>
<tr>
<td>KAUST</td>
<td>King Abdullah University of Science and Technology</td>
</tr>
<tr>
<td>LBM</td>
<td>Lattice-Boltzmann Method</td>
</tr>
<tr>
<td>MFEM</td>
<td>Mixed Finite Element Method</td>
</tr>
<tr>
<td>MPI</td>
<td>Message Passing Interface</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PR EOS</td>
<td>Peng-Robinson Equation of State</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>SMG</td>
<td>Semi-coarsening Multigrid</td>
</tr>
<tr>
<td>TPD</td>
<td>Tangent Plane Distance</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1 Flow chart of IMPEC method. ................................................................. 32
Figure 2 Discretization of the 2D domain in the two-phase composition flow simulation. ................................................................................................................. 33
Figure 3 Two strategies of domain decompositions. ........................................ 38
Figure 4 A sketch of the pressure field. ................................................................. 38
Figure 5 A sketch map of the velocity field in the x-direction (green arrows) and in the y-direction (blue arrows). ................................................................. 40
Figure 6 Speedup of the parallel two-phase compositional flow code. ............ 43
Figure 7 A schematic plot of the two-phase flash calculation at constant T and p. ..... 44
Figure 8 Flow chart of the flash calculation. ............................................................ 46
Figure 9 The hierarchical basis in one dimension is depicted on the left. The basis functions of all levels together construct the approximation of a function. In the middle column the 2D hierarchical basis is shown. Taking only the subset with the most important contributions (gray selection) constructs a sparse grid in the top right corner, which has significantly less grid points than the $7^2$ full grid in the bottom right corner. [118] .................................................................................................................. 68
Figure 10 The union of the sparse grids used for the sparse grid surrogates for the temperatures 100 K, 220 K and 480 K, respectively. For each observable only a subset of the shown points is stored to build a surrogate. ................................................. 78
Figure 11 On the left an overview of the number of values stored for each observable and for each of the temperatures can be observed. For comparison the size of the union of all grids is given. On the right the average relative $L_1$ errors of the full grid surrogates $S_f$ are given. Note that the small horizontal indicator above each bar shows the average relative $L_1$ error of the respective sparse grid surrogate $S_s$. ..................... 79
Figure 12 Occurrences of the absolute errors of the $p, z_1$ trajectories created by $S_f$ for all time steps at the three respective locations in the four simulation cases. One can clearly see that for most of the 21024 recorded time steps the trajectories of the simulations using $S_f$ is quite close to the ones of the simulations using the original flash calculations. ........................................................................................................... 80
Figure 13 The deviation of the $p, z_1$ trajectory created by $S_s$ from the one created by $S_f$ for all time steps at the three respective locations in the four simulation cases. The sparse grid surrogate retrieves results that are close to the full grid surrogate, with the deviation being much smaller than the error introduced by the full grid surrogate. . 81
Figure 14 The relative error of propane recovery rate of full grid surrogate is shown on the left. The kink of case 1 is the result of a change in the sign of the absolute error. On the right the relative difference of recovery rates between $S_f$ and $S_s$ is shown. The difference between $S_f$ and $S_s$ is small compared to the error introduced by $S_f$.

Figure 15 The left plot shows the run time of different parts of the code on two different grids with and without the surrogate. The right plot shows the percentages of run time of different parts of the code. The number of processors is 64.

Figure 16 Change in the molar fraction of propane in the domain during the three years. (a), (b), (c) and (d) stand for the condition at the end of 0.1 year, one year, two years and three years, respectively.

Figure 17 (a) Recovery rate of propane during the three years and (b) streamlines in the domain at the end of the third year.

Figure 18 Flow chart of the wormhole simulation.

Figure 19 Discretization of the 2D domain in the wormhole simulation.

Figure 20 Indices of the unknowns in the $x$ vector.

Figure 21 $x$-direction velocity field in nine shifts. The bold arrows stand for one values, and the other arrows stand for the zero values.

Figure 22 Pressure field in two shifts. The black points stand for one values and the white points stand for zero values.

Figure 23 Domain decomposition strategy. The red, black, green and blue parts are allocated to Processors 0, 1, 2 and 3, respectively.

Figure 24 Overlapping edges needed by the $x$-momentum equations (1), $y$-momentum equations (2) and continuity equations (3) in Processor 0. All variables and equations in Processor 0 can be shown in (4).

Figure 25 Indices of unknowns in $x$ vector in the parallel code.

Figure 26 The linear system $Ax = b$ in Hypre.

Figure 27 Shear-driven fluid flow in a square cavity. $u$ stands for the $x$-direction boundary velocity, and $v$ stands for the $y$-direction boundary velocity.

Figure 28 Pressure field of the 2D domain.

Figure 29 $x$-direction velocity field of the 2D domain.

Figure 30 $y$-direction velocity field of the 2D domain.

Figure 31 Streamlines of the 2D domain.

Figure 32 $y$-direction velocity components along the horizontal centerline.

Figure 33 $x$-direction velocity components along the vertical centerline.

Figure 34 Changes in the average left boundary pressures in the three resolutions.
Figure 35 Changes in the average permeabilities in the three resolutions.......................... 115
Figure 36 Porosity fields at the end of the simulation in the three resolutions: (a) 20^2, (b) 40^2 and (c) 60^2. .................................................................................................................. 116
Figure 37 Distributions of the porosities in the Darcy framework (left) and DBF framework (right) at five different moments. The range of color from blue to red indicates the range of porosity from zero to one. ........................................ 117
Figure 38 Average porosity curves in both the Darcy and DBF frameworks. .................. 118
Figure 39 Average permeability curves in both the Darcy and DBF frameworks. ........ 120
Figure 40 Average interfacial area curves in both the Darcy and DBF frameworks...... 120
Figure 41 Right boundary flux curves in both the Darcy and DBF frameworks........... 121
Figure 42 Run time and speedup of the code when simulating the shear driven cavity flow problem.................................................................................................................. 122
Figure 43 Discretization of the 3D domain.................................................................... 123
Figure 44 Indices of unknowns in the x vector............................................................... 126
Figure 45 3D Shear-driven cavity flow configuration and coordinate system [140]. .... 138
Figure 46 Velocity profile of the x-direction component (u) on the vertical centreline of the plane z = 0.5. .................................................................................................................. 138
Figure 47 Velocity profile of the y-direction component (v) on the horizontal centreline of the plane z = 0.5. .................................................................................................................. 139
Figure 48 Porosity isosurfaces of 0.8 in the Darcy framework (left) and DBF framework (right) at four different moments: the 25000th second, the 50000th second, the 75000th second and the 10^5th second from above to bottom, respectively........... 142
Figure 49 Average porosity curves in the Darcy and DBF frameworks....................... 143
Figure 50 Average permeability curves in the Darcy and DBF frameworks. ............... 143
Figure 51 Average interfacial area curves in the Darcy and DBF frameworks.............. 144
Figure 52 Out boundary flux curves in the Darcy and DBF frameworks.................... 144
Figure 53 Five dissolution patterns simulated by the DBF framework......................... 148
Figure 54 Run time, solver time and speedup of different number of processors......... 150
Figure 55 FORTRAN90 code snippet to set up the grid. ............................................ 182
Figure 56 FORTRAN90 code snippet to set up the stencil........................................ 182
Figure 57 FORTRAN90 code snippet to set up the matrix........................................ 183
Figure 58 FORTRAN90 code snippet to set up the right-hand side vector................. 184
LIST OF TABLES

Table 1 Average run time of the parallel code, P is the number of processors, N is the grid size, and the unit is second. ................................................................. 40
Table 2 Speedup of the parallel code, P is the number of processors, N is the grid size.. 41
Table 3 Solver time, P is the number of processors, N is the grid size, and the unit is second. ........................................................................................................... 43
Table 4 The meaning of the observables in the two-component flash calculation........... 73
Table 5 Parameters of the four cases. ................................................................................. 75
Table 6 Average run time and acceleration of flash calculations with and without the use of a surrogate S. N is the grid size, P is the number of processors, and the unit is second. ........................................................................................................... 87
Table 7 Average run time and acceleration of the parallel code with the use of a surrogate S. N is the grid size, P is the number of processors, and the unit is second. The average run time of the parallel code without S is referred to Table 1............. 87
Table 8 Properties of the acid flow and porous medium in the dissolution experiment. 113
Table 9 Average total run time and solver time of the code when solving the shear driven cavity flow problem. P is the number of processors, and the unit is second. ........ 122
Table 10 Properties of the acid flow and matrix................................................................. 149
Table 11 Average run time and solver time of the parallel code. P is the number of processors, and the unit is second................................................................. 150
Table 12 Properties of the acid flow and the porous medium in the error analysis experiment ........................................................................................................... 174
Table 13 Numerical errors of porosities in different rectangular meshes at different time steps. ........................................................................................................... 176
Table 14 Numerical errors of concentrations in different rectangular meshes at different time steps. ........................................................................................................... 176
Table 15 Numerical errors of pressures in different rectangular meshes at different time steps. ........................................................................................................... 176
Table 16 Numerical errors of velocities in different rectangular meshes at different time steps. ........................................................................................................... 176
1. Introduction and Literature Review

1.1 Multi-phase Flow

In many physical and chemical engineering applications, one of the most challenging topics is the simulation of multi-phase flows. In reservoir modeling, the challenge comes when the reservoir is opened to production, and the flows of hydrocarbons form. The reservoir pressure drops at the production point, and the phase equilibrium is broken. As a result, the physical properties of the flow are altered in various degrees in the whole reservoir. With further withdrawal of flows, changes continue, and a series of partial differential equations are needed to describe the unsteady state of the flows. The flows can fall in the oil, gas and water phases, among others. If the phases are immiscible such as oil and water, the flow is not difficult to describe. However, if the phases are miscible, such as oil and gas, more equations are needed to describe the composition conditions in the phases. The miscible flow is called multi-phase compositional flow. The multi-phase flow simulation can be applied in broad regimes such as enhancing oil and gas recovery rate from hydrocarbon reservoirs by the injection of chemicals [1][2][3] in the tertiary recovery stage, storing greenhouse gases in saline aquifers and oil fields [4][5], monitoring the transport of contaminants in the groundwater flow [6][7][8] among others.

The study to the flows in reservoirs can be traced back to the year 1856 when Frenchman Henry Darcy investigated the flow characteristics of sand filters for water purification. He established the foundation of the quantitative theory for the flow of
homogeneous fluids in porous media [9]. Then Muskat and Wyckoff [10]-[13] studied the flow of reservoir fluids in 1930’s, and their work was instrumental in advancing the knowledge of reservoir dynamics to its present state.

Algorithmically, many schemes have been developed to simulate the multi-phase flow, such as the fully implicit scheme and implicit-explicit hybrid schemes. The use of the fully implicit scheme in reservoir simulations can be traced back to the work of Roebuck et al. [14], where they developed an implicit numerical method to simulate the differential and algebraic relations governing 1D three-phase flow in porous media. Then much work to improve the solution procedure of the fully implicit scheme was carried out such as [15] and [16]. The first application of the implicit-explicit hybrid scheme in reservoir simulations can be found in [17], where an implicit equation for the oil-phase pressure and two explicit equations for the over-all composition and water saturation were obtained. Further investigations on the implicit-explicit hybrid scheme can be found in [18]-[22]. The fully implicit scheme can solve for the pressures, velocities, saturations et al. simultaneously, and its time step can be set larger than in the implicit-explicit hybrid scheme. However, the fully implicit scheme requires significantly more computing resources than the implicit-explicit hybrid scheme when solving the discretized linear system. The implicit-explicit hybrid scheme solves for the pressures implicitly, then solve for the saturations or concentrations explicitly. Its computing cost is lower compared to the fully implicit scheme, but to achieve convergence its time step is limited by the Courant-Friedrichs-Lewy condition and the splitting error from decoupling
the pressure and saturation/concentration equations. In many large-scale simulations, the number of unknowns in the discretized linear system of the fully implicit scheme becomes very large, which is a challenge when solving the system. Thus, the implicit-explicit hybrid scheme is favored in such conditions.

Moreover, many discretization methods have been proposed in the multi-phase flow simulation. Popular methods include the finite difference method (FDM) [23][24], the finite element method (FEM) [25]-[28], and the Lattice-Boltzmann method (LBM) [29]-[38]. Traditional FDM dominates both theoretical and practical work in the multi-phase flow simulation. It is based on the physical concepts such as mass conservation law, energy conservation law, Darcy’s law and isothermal fluid phase behavior. Both structured and unstructured grids can be used in discretization to represent the geometry of the reservoir accurately. FDM is simple and easy to implement, but it is not very versatile in dealing with boundaries and achieving stable results. Some restrictive mesh refinement may have to be imposed to obtain a nonsingular system. However, the FEM can easily guarantee well-posedness and its variational framework is amenable to a posteriori error estimation. However, the sparsity patterns from FEM may be less structured and hence more difficult to parallelize efficiently. Although FDM and FEM achieve much success in the multi-phase flow simulation, they cannot tackle the complex pore space and inherent free-boundary issues such as breaking and merging of interfaces in reservoirs. Thus, the LBM is proposed to capture the microscopic effects and reproduce the macroscopic behavior. The LBM does not track interfaces but rather
maintains the sharp interfaces automatically. Macroscopic behaviors such as interface
dynamics can arise naturally from the microscopic effects. FDM is the discretization
method of choice in this work, considering its ready realization of parallelism.

1.2 Flash Calculation

The oil and gas two-phase compositional flow is a special case of multi-phase
compositional flow. Such flow involves the motion of two-phase systems with chemical
constituents partitioning between phases as a result of changes in the operating conditions,
the compositions or both. In some chemical engineering applications involving two-phase
flows, the time scale associated with the flow is on the same order of magnitude as the
one of the chemical process. However, this is not the case for most reservoir flows. In
other words, the time scale of the flow is much longer than that of the composition
splitting. This suggests that a state of the chemical equilibrium is maintained throughout a
reservoir simulation. The calculation of composition partitioning between the phases in
hydrocarbon reservoirs is a computational task known as a “flash calculation”. Flash
calculations can be extremely time consuming, and the search for an acceleration
technique is ongoing.

To improve the speed of the flash calculation, several approaches have been introduced
focusing essentially on the mathematical manipulations and simplifying thermodynamic
considerations. For example, Li and Johns [39] designed a rapid and robust method to
enhance the flash calculation based on the Rachford-Rice algorithm. Rachford-Rice
computations are iterative procedures to determine the phase equilibrium compositions based on predetermined, constant equilibrium ratios. However, the method may converge slowly or even not converge at all in some cases. Therefore, Li and Johns proposed a new objective function that has two fewer asymptotes and is based on the equilibrium phase molar fractions. Furthermore, Li and Johns [40] investigated an improvement of the speed of the flash calculation by utilizing the binary interaction parameters (BIP). They extended Michelsen’s [41]-[43] approach for the case when all BIPs are nonzero without any approximation. The approach is exact in that the equilibrium phase compositions for the same BIPs are identical to those with the conventional flash calculation. Moreover, the approach eliminates the Rachford-Rice procedure, and it is more robust than the conventional flash calculation procedure. In addition to that, Belkadi et al. [44] proposed a tie-line distance-based approximation method to obtain approximate flash calculation results in a two-phase region. The method uses the distance to a previous tie-line in the same grid block to decide whether an approximation should be made. Rasmussen et al. [45] proposed a criterion to decide when it is justified to bypass the stability analysis. The implementation of these methods shows that the computation time spent on the flash calculation can be significantly reduced for a number of simple compositional flow simulations.

Despite many of these efforts to accelerate the flash calculation, they only improve the calculation marginally by some type of mathematical manipulation or by ignoring some complex thermodynamic procedures, such as stability analysis. In this work, an
improvement to the flash calculation in its primitive form without any simplified assumptions is suggested. This has been achieved by decoupling the flash calculation from reservoir simulations using a sparse grid surrogate. In this approach, the reservoir simulations require no flash calculations. Instead, a surrogate model that evaluates a sparse grid approximation is used. The flash calculation is shifted from the reservoir simulation (online) to a pre-simulation (offline) phase where the sparse grid approximation is created. This significantly reduces the online CPU time and yet preserves the accuracy.

Using sparse grid surrogates for speeding up computations by an online-offline splitting has been done before for, e.g., exploration of parameter spaces [46] or computational steering [47]. Our approach is based on these exploration techniques, but so far it is not supporting a scalable implementation [48] because we employ spatially adaptive sparse grids instead of dimensionally adaptive ones. However, sparse grid surrogates have also been employed in studying reaction pathways through a potential energy surface [49]. Using sparse grids in density estimation allows an offline-online splitting for handling large data sets [50]. Sparse grids are also widely employed for stochastic collocation in the field of uncertainty quantification. There they are not used in an online-offline splitting but as surrogates for evaluating moments of stochastic outcomes of random PDEs [51]. Sparse grid interpolation has also been proposed for applications in characterizing oil reservoirs [52], but there it is used as a surrogate for optimization. The sparse grid surrogate used there is based on global interpolating
polynomials, whereas our approach is relying on piece-wise linear basis functions to deal with discontinuities.

Splitting the computation into an offline and an online phase has the advantage that the data from the offline phase can be reused for many different online simulations. Sparse grids are used to estimate the compositions and other relevant parameters over a set of adaptive sparse grid points. Compared to a Cartesian grid surrogate, the sparse grid surrogate requires a lot less values to be stored and consequently less evaluations of the flash calculation, which drastically reduces the computing time of the offline phase as well as the memory requirement during the online phase. The approach is tested for a two-phase and two-component flow, requiring a two-dimensional (2D) surrogate model for the flash calculation. The small test surrogate created in this work is chosen to show the applicability and extensibility of the approach. Our test simulations demonstrate satisfactory results.

1.3 Wormhole

The simulation of wormhole propagation in porous medium is a challenge topic in the field of reservoir simulation. In this phenomenon, a worm-like hole is generated and propagated in subsurface formations from the injection of acids into a supercritical acid dissolution system. Although acid dissolution fronts can be generated in both subcritical and supercritical acid dissolution systems, they are unstable only in the supercritical acid dissolution systems, so that wormholes can only be generated and propagated in such
supercritical acid dissolution systems [53]-[56]. Engineers and hydrologists appeal to this technique when the efficiency of oil production wells deteriorates due to the deposition of mud and fines at the perforated well pore pipe. In this case, different from the multi-phase flow discussed before, a single-phase acid fluid is injected to promote chemical reactions that result in the dissolution of deposits to facilitate the displacement of fluids. The advancement of the chemical reaction front does not propagate uniformly in the injection direction. The heterogeneity of porosity and permeability in the subsurface formations is an essential feature that promotes the non-uniformity of the chemical reaction fronts. The chemical reaction fronts tend to advance in certain directions more than the other directions and a wormhole pattern is established. Because of its importance in subsurface reservoir management, the theoretical, experimental and numerical studies of wormhole generation and propagation in carbonate reservoirs have progressed fast in the last forty years.

Theoretically, wormhole generation and propagation belong to the chemical dissolution-front instability problems in porous rocks. Because physical and chemical dissolution-front instability problems are closely associated with the geosciences [57]-[64], geo-environmental engineering [65]-[69] and petroleum engineering [53]-[55], Zhao et al. have conducted extensive and systematic theoretical studies in recent years [53]-[71]. In particular, many important factors such as the mineral reactive surface area [58], mineral dissolution ratio [60], solute dispersion [61], medium anisotropy [64], temperature effect [70][71], and medium and fluid compressibility [62] have been
successfully considered in both theoretical analyses and computational simulations. Due to the importance of this type of work, it has become an essential part of the emerging field of computational geosciences [56]. The major conclusions of the existing theoretical work are (1) wormholes can only be generated and propagated in a supercritical acid dissolution system in which the Zhao number is greater than the corresponding critical Zhao number of the system; (2) wormholes may have different modes, such as the fundamental mode, the fingering mode and fractal mode, depending on the Zhao number of the supercritical acid dissolution system; (3) by comparing the Zhao number of an acid dissolution system with the corresponding critical Zhao number of the acid dissolution system, the instability of the acid dissolution front in carbonate rocks can be assessed [53]-[55].

Experimentally, it is important to develop methods to characterize wormhole structures. In this regard, a number of techniques have been implemented. One technique is injecting molten metals that have a lower melting point (e.g., Wood’s metal) into the acid-etched channels of the porous medium and allowing the metal to solidify the channels [73]. However, the delicate branches of wormholes tend to break during the dissolution process. Another technique is using Neutron transmission tomography to generate patterns of porosity variations [74]. Furthermore, the effect of different factors on the forming of wormholes is also investigated. For example, Qiu et al. [75] investigated the effects of diffusion coefficients of HCl acid as it reacts with calcite. Sayed et al. [76] investigated the effect of the presence of crude oil in the formation on the performance of
emulsified acid in stimulating carbonate formations. Kumar et al. [77] demonstrated the impact of oil saturation on wormholing characteristics while acidizing field and outcrop cores under reservoir conditions.

Numerically, a number of models have been developed to simulate the propagation dynamics of wormholes. The models include the capillary tube model, the network model and the two-scale model. Although the capillary tube model [78] can describe the effect of fluid leakage and the transport and reaction mechanisms in the wormholes, it cannot provide us with the conditions that cause the formation of wormholes. The network model is developed by Hoefner and Fogler [73], Fredd and Fogler [79], and Daccord et al. [80]. This model uses a network of tubes to represent a porous medium. The acid reacts at the wall of the tube, and the dissolution is developed, resulting in an increase in the tube radius. Although the network model can predict the dissolution patterns, the results are very different from the experimental results. In addition, a core size simulation using the network model is computationally very expensive. A two-scale model (also called averaged model or continuum model) [81]-[83] can describe the dissolution patterns based on both the Darcy scale and pore scale. The information achieved by semi-empirical equations on the pore scale is passed to the Darcy scale equations to describe the dissolution procedure. The development of dissolution changes the pore structures, and consequently the property parameters such as the porosity in the pore scale model. Therefore, the information on both the Darcy scale and pore scale can communicate with each other in the whole simulation procedure. Based on the two-scale
model, much work has been carried out. Liu et al. [84] studied the wormhole propagation behavior in the invaded zone and compressed zone, concluding that the effect of the compressed zone on the wormhole propagation increases with the decrease in the compressibility factor. Liu et al. [85] also studied the effect of the two distribution laws of porosity on the dissolution patterns, concluding that the results from the normal distribution law are closer to the experimental observations than the uniform distribution law. Different from much of the two-scale model research, which has only 2D simulation, Maheshwari et al. [86] extended the 2D continuum model to the 3D case. Based on the 3D case, they studied the effect of the acid-injection rate on the amount of acid required to breakthrough. They also studied the flow dynamics inside a wormhole. In addition to the three popular models, another modeling methodology based on the continuum hypothesis as applied to porous media has been suggested. In this framework, field variables represent continuous functions of space and time, and the governing conservation equations are described in the form of differential equations [87]-[89].

Although there are a large number of numerical simulations that have been conducted to study the wormhole propagation in porous media, most of them are based on the Darcy framework. While this may be adequate for scenarios where there is not a significant change in the porosity, in cases where there is a large change in the porosity (and consequently the permeability), this may be incorrect. There are situations where the porosity becomes large (but less than one) and its spatial distribution also becomes very heterogeneous, and the fluid velocity is high in the high-porosity area. Therefore, another
framework that accounts for both the porous media and clear fluid area is suggested. In this work, the Darcy-Brinkman-Forchheimer (DBF) framework is introduced to describe matrix acidization for the purpose of retrieving more reasonable results. The DBF framework can predict the convective flow reasonably well in comparison with the experimental data on the condition of variable porosities [90]. Moreover, the DBF framework accounts for the boundary-layer development, macroscopic shear stress, microscopic shear stress and inertial force. A porous medium and clear fluid interface is best dealt with by the DBF framework and the continuity of velocities and stresses at the interface [91]. To the best of our knowledge, no work has ever considered simulating wormhole dynamics within the DBF framework; this contribution fills the gap.

Because much work has already focused on the factors that affect the dissolution pattern, such topics are not discussed in this work. Instead, this work contributes to the comparison between the simulation results of the Darcy framework and DBF framework. In addition to that, to simulate the cases accurately, a sufficiently fine grid is needed. In such condition, a traditional serial code is overwhelmed, making parallelism necessary. The implicit-explicit hybrid scheme and FDM discussed before are applied in the parallel simulation of wormholes. Zhao et al. [55]-[57] used a combination of FEM and FDM to simulate the wormhole generation and propagation in carbonate rocks. However, compared with Zhao’s methods, our proposed FDM is more easily parallelized. For reproducibility, a detailed description of the coefficient matrix in the solver is given.
1.4 Parallelism in Reservoir Simulation

Parallelism is a main endeavor in the acceleration of reservoir simulations. The endeavor can be traced back to the early 1980s when research on the parallel reservoir simulations had realized its promising future. For example, Wheeler [92] developed a black-oil simulator on a hypercube in 1989. With this simulator, a correlation of the computation efficiency with the problem size and the number of processors demonstrated that up to 96% of the available CPU time on the hypercube can be used. Kaarstad et al. [93] also developed a 2D oil/water simulator running on a 16384-processor MasPar MP-2 machine. He showed that a model problem using 1 million grid cells could be solved in a few minutes of computer time. In 1997, the simulators on the distributed-memory machines gradually moved from the research to the production-type environment. At that time, many researchers presented the real field applications using the massively parallel simulators. For example, Shiralkar et al. [94] introduced a distributed-memory simulator called Falcon. They used several programming languages to handle the data distribution and communication among the processors. They also reported the largest test problem of 16 million cells with the IMPES algorithm running on a 512-node Cray T3D machine. The run time for 7.6 years of history was slightly more than 1 hour. They stressed the importance of developing a programming language for the distributed-memory system. Chien et al. [95] also presented a distributed-memory simulator based on an existing simulator written in FORTRAN77. They used the domain decomposition and the MPI libraries on IBM SP-2 nodes. The largest field-model size reported is more than 1 million
cells running on a 16- or 32-node SP-2 system. In 2003, the PC cluster technology was rapidly evolving and promised to be a low cost alternative to the simulations on traditional supercomputers. Habiballah and Hayder [96] investigated the use of state of the art PC clusters in the simulation of the massive reservoir models. Their investigation indicates that the large-scale reservoir simulations can be performed efficiently and cost effectively on the latest PC clusters. Edwards et al. [97] investigated the newest development of the reservoir simulation and foresaw the next generation simulators to process more realistic reservoir models. All of these efforts indicate that the parallel computing offers an attractive alternative for simulating the giant oil and gas reservoirs.

In this work, the sparse grid techniques are integrated into the parallel code of the two-phase compositional flow simulation, and the performance improvement is evaluated. Moreover, the simulation of wormholes is also parallelized. To our knowledge, wormhole propagation has not been simulated with parallelism before.

The principal challenge in parallelizing reservoir simulators is the preconditioner and solver to solve the discretized linear system. In the early days of reservoir simulations, direct methods such as Gauss elimination and Gram-Schmidt orthogonalisation were used to solve the linear system. These methods have the disadvantage of producing dense matrices as intermediate data structures. Then crude, slowly converging iterative methods such as Jacobi, Gauss-Seidel and successive over-relaxation were applied. These methods have a convergence rate that degrades as the computational grid is refined but preserve sparsity. More advanced iterative methods, such as Krylov subspace methods, likewise
exploit the sparsity of the coefficient matrix and improve convergence rate. The first introduced Krylov method is conjugate gradient method [98], but it can only tackle the symmetrical and positive definite matrices. Thus, the other Krylov methods such as GMRES [99] and ORTHOMIN [100] were proposed to solve general matrix equations. To increase the efficiency of Krylov methods, the preconditioning techniques such as incomplete decompositions and multigrid smoothers [101] have to be applied. A wide variety of preconditioning techniques have been developed. For example, Klie et al. [102][103] developed the two-stage preconditioners to accelerate linear solvers in two-phase flow simulations. In 2002, Cai and Keyes [104] proposed nonlinear preconditioning in the form of the additive Schwarz preconditioned inexact Newton method which can achieve robustness when processing the shocks and fronts in fluid dynamics problems. Skogestad et al. [105] further applied the method to the implicit discretized system from two-phase flows, but their work did not consider the capillary forces and gravity in the porous medium. Sun et al. [106] filled the gap and considered more general two-phase flow. Recently, the preconditioners and solvers based on GPU are developed. For example, Sudan et al. [107] developed a solver based on the many-core GPU to simulate the black oil model and compositional flows. Khaz’ali et al. [108] employed the computational power of a GPU to build a preconditioner and solved the discretized linear system from the compositional flow simulations.

Lu [109] points out that for parallel simulations, the high performance preconditioner and solver library Hypre [110] shows great advantages over the other solvers of
two-phase problem, and Hypre is highly scalable with the iterative model in parallel
environment. Thus, Hypre is used to solve the discretized linear system of the two-phase
compositional flow simulation in this work. Hypre is used for the first time in wormhole
simulations to evaluate its applicability. Because the coefficient matrices are generally
nonsymmetrical, the GMRES solver or its variants are used as Krylov accelerators.

The rest of the thesis is organized as follows. Chapter 2 describes the parallel
simulation of the two-phase compositional flow. Chapter 3 gives details of the flash
calculation in the two-phase compositional flow. Chapter 4 states the application of
sparse grids to speed up the flash calculation. Chapter 5 and 6 focus on the 2D and 3D
parallel simulations of the wormhole propagation respectively. Chapter 7 analyzes the
wormhole model with mixed finite element methods and we present our conclusions in
Chapter 8.
2. Parallel Simulation of Two-phase Compositional Flows

2.1 Basic Equations

In this work, an oil and gas two-phase compositional flow model is considered. For the sake of simplicity, we consider a 2D case. Generalization to three dimensions (3D) with some effort is possible. The general mass conservation law and the Darcy’s law are the basis for a series of derived equations used in the two-phase compositional flow model. The fact that the two phases jointly fill the voids implies that the following relation holds

\[ S_o + S_g = 1, \]

where \( S \) stands for the saturation. The subscripts \( o \) and \( g \) stand for the oil and gas phases, respectively. In this work, the capillary pressure is omitted, and this implies that the pressures of both oil and gas phases are equal and \( p \) is used to denote this pressure. The mass conservation of the two-phase compositional flow model can be written as

\[
\frac{\partial \left[ \phi \left( x_{mo} \xi_o S_o + x_{mg} \xi_g S_g \right) \right]}{\partial t} = -\nabla \cdot \left( x_{mo} \xi_o u_o + x_{mg} \xi_g u_g \right) \\
+ x_{mo} q_o + x_{mg} q_g, \quad m = 1, 2, ..., c, \tag{1}
\]

where each phase has its own Darcy’s velocity \( u_\alpha \) and mass flux source \( q_\alpha \) \((\alpha = o, g)\). \( \phi \) is the porosity, \( t \) is the time, \( \xi \) is the molar density, and \( c \) is the number of components. \( x_{m\alpha} (\alpha = o, g) \) stands for the molar fraction of component \( m \) in phase \( \alpha \). The molar density of phase \( \alpha \) is given by

\[
\xi_\alpha = \sum_{m=1}^{c} \xi_{m\alpha}, \quad \alpha = o, g.
\]

The molar fraction of component \( m \) in phase \( \alpha \) is given as
$$x_{m\alpha} = \xi_{m\alpha}^{\alpha}, \quad \alpha = o, g.$$  

(1) is the concentration equation of the two-phase compositional flow model.

The Darcy’s law in the model can be expressed as

$$u_\alpha = -\frac{k_{ra}}{\mu_\alpha} k_\alpha (\nabla p - \rho_\alpha g), \quad \alpha = o, g,$$

(2)

where $\mu$ is the viscosity, $\rho$ is the mass density, $g$ is the gravity vector, $k_{ra}$ is the relative permeability and $k$ is the absolute permeability tensor which can be expressed as

$$k = \begin{pmatrix} k_{xx} & k_{xy} \\ k_{yx} & k_{yy} \end{pmatrix}.$$

It is noticed that the absolute permeability is the intrinsic property of rocks and has no relationship with the fluid. If the porous medium is isotropic, there is $k_{xx} = k_{yy}$ and $k_{xy} = k_{yx} = 0$. Relative permeability is a concept without any scientific background. It is used in the multi-phase flow description and based on the observation and laboratory experiments. In this work, the relative permeability of oil phase is defined as

$$k_{ro} = S_o^2,$$

and the relative permeability of gas phase is defined as

$$k_{rg} = (1 - S_o)^2.$$

Viscosity is an important fluid property. It represents the drag forces caused by the attractive forces in adjacent fluid layers. Generally speaking, the oil viscosity is much higher than gas viscosity at the same pressure and temperature.
If there is a small volume $V$, the flow volume $V_f = V\phi$. In most general scenarios, there is

$$\frac{\partial V_f}{\partial t} = \frac{\partial (V\phi)}{\partial t} = V \frac{\partial \phi}{\partial t} = V \frac{\partial \phi}{\partial p} \frac{\partial p}{\partial t} = V C_R \phi \frac{\partial p}{\partial t}. \quad (3)$$

Here the expression

$$C_R = \frac{1}{\phi} \frac{\partial \phi}{\partial p}$$

is defined as the isothermal compressibility of the rocks. In the same way, the isothermal compressibility of the flow can be defined as

$$C_f = -\frac{1}{V_f} \left( \frac{\partial V_f}{\partial p} \right)_{T,N}. \quad (4)$$

Here, $N = (N_1, N_2, \ldots, N_c)$ is the vector composed of the molar amount of each component. Partial molar volume of each component is define as

$$\bar{v}_m = \left( \frac{\partial V_f}{\partial N_m} \right)_{T,p,N_{\neq m}}, m = 1, \ldots, c. \quad (4)$$

Here, $N_m = (N_1, \ldots, N_{m-1}, N_{m+1}, \ldots, N_c)$. Because $V_f$ depends on the pressure and the molar amounts of the components and $V_f = V_f(p,N_1,\ldots,N_c)$, the expression of $\frac{\partial V_f}{\partial t}$ can be derived in another way

$$\frac{\partial V_f}{\partial t} = \left( \frac{\partial V_f}{\partial p} \right)_{T,N} \frac{\partial p}{\partial t} + \left( \frac{\partial V_f}{\partial N_1} \right)_{T,N_1} \frac{\partial N_1}{\partial t} + \cdots + \left( \frac{\partial V_f}{\partial N_c} \right)_{T,N_c} \frac{\partial N_c}{\partial t}$$

$$= -\phi V C_f \frac{\partial p}{\partial t} + \sum_{m=1}^{c} \bar{v}_m \frac{\partial N_m}{\partial t}.$$

Comparing (3) and (4), we have

$$V \phi C_R \frac{\partial p}{\partial t} = -\phi V C_f \frac{\partial p}{\partial t} + \sum_{m=1}^{c} \bar{v}_m \frac{\partial N_m}{\partial t},$$
\[
\frac{\phi(C_R + C_f)}{\partial t} \frac{\partial p}{\partial t} = \sum_{m=1}^{c} \bar{v}_m (x_{mo}q_o + x_{mg}q_g - \nabla \cdot (\sum_{\alpha=o,g} u_{\alpha}x_{m\alpha} \xi_{\alpha})).
\]

If the total isothermal compressibility \( C_T \) is defined as \( C_T = C_R + C_f \), and if we define \( q^t_m = x_{mo}q_o + x_{mg}q_g \), the above equation can be written as

\[
\frac{\phi C_T}{\partial t} \frac{\partial p}{\partial t} + \sum_{m=1}^{c} \bar{v}_m \nabla \cdot (\sum_{\alpha=o,g} u_{\alpha}x_{m\alpha} \xi_{\alpha}) = \sum_{m=1}^{c} \bar{v}_m q^t_m.
\]

Substituting \( u_{\alpha} \) with (2), it follows that

\[
\frac{\phi C_T}{\partial t} \frac{\partial p}{\partial t} + \sum_{m=1}^{c} \bar{v}_m \nabla \cdot (\sum_{\alpha=o,g} \frac{k_{\alpha}}{\mu_{\alpha}} k_{\alpha} (\nabla p - \rho_{\alpha} g) x_{m\alpha} \xi_{\alpha}) = \sum_{m=1}^{c} \bar{v}_m q^t_m. \tag{5}
\]

This is the pressure equation of the two-phase compositional flow model. In this work, \( C_R \) is set to 0, thus \( C_T = C_f \).

To solve the model, the physical and chemical parameters of the flow are first set by flash calculations. Then, the pressure field can be calculated given that all other terms are known from previous time step. After the pressure field is evaluated, the velocity field can be calculated using the Darcy’s law, and the composition field can be updated using the concentration equation. Finally, with the new pressure field and composition field, one can go back to flash calculations to continue the loop. This method is called IMplicit Pressure and Explicit Concentration method (IMPEC). The general flow chart of IMPEC method is shown in Figure 1.
2.2 Discretization

Suppose a 2D domain is divided into $nx \times ny$ cells, as shown in Figure 2. In the cell-centered finite difference (CCFD) method, pressures are imposed on the center of cells and velocities are imposed on the edges of cells. For the cell in the $i$th column and the $j$th row, the notation $p_{i+\frac{1}{2}, j+\frac{1}{2}}$ is used to represent the pressure of the cell, and the
notations \( u_{x,i,j + \frac{1}{2}} \), \( u_{x,i+1,j + \frac{1}{2}} \), \( u_{y,i+\frac{1}{2},j} \) and \( u_{y,i+\frac{1}{2},j+1} \) are used to represent the four velocities on the edges of the cell, respectively.

Figure 2 Discretization of the 2D domain in the two-phase composition flow simulation.

Flash calculations have to be done in each cell of the grid to output the flow parameters in the cell. Then, the pressure equation (5) can be discretized as below. At time step \( t + 1 \), (5) in the cell can be discretized as

\[
\phi_{i + \frac{1}{2},j + \frac{1}{2}}^t C_t^t \left( \frac{p_{i + \frac{1}{2},j + \frac{1}{2}}^{t+1} - p_{i + \frac{1}{2},j + \frac{1}{2}}^t}{\Delta t} \right) - \cot o x 1 \left( \frac{p_{i + \frac{1}{2},j + \frac{1}{2}}^{t+1} - p_{i + \frac{1}{2},j + \frac{1}{2}}^t}{x_{i + \frac{1}{2}} - x_{i + \frac{1}{2}}} - \rho_{o,i+1,j+\frac{1}{2}}^t g x \right) + \cot o x 2 \left( \frac{p_{i + \frac{1}{2},j + \frac{1}{2}}^{t+1} - p_{i - \frac{1}{2},j + \frac{1}{2}}^t}{x_{i + \frac{1}{2}} - x_{i - \frac{1}{2}}} - \rho_{o,i,j+\frac{1}{2}}^t g x \right) + \frac{x_{i+1} - x_{i}}{x_{i+1} - x_{i}}
\]
\[-\cot x_1 \left( p_{l+\frac{1}{2},j+\frac{1}{2}}^{t+1} - p_{l+\frac{1}{2},j+\frac{1}{2}}^{t} \right) - \rho_{g,l+1,j+\frac{1}{2}}^{t} \right) + \cot x_2 \left( p_{l+\frac{1}{2},j+\frac{1}{2}}^{t+1} - p_{l+\frac{1}{2},j+\frac{1}{2}}^{t} \right) - \rho_{g,l+1,j+\frac{1}{2}}^{t} \right) \]

\[= \sum_{m=1}^{c} \bar{v}_{m,i+\frac{1}{2},j+\frac{1}{2}}^{t} q_{m,i+\frac{1}{2},j+\frac{1}{2}}^{t} \]

with

\[cotox_1 = \sum_{m=1}^{c} \frac{k_{r_0,i+1,j+\frac{1}{2}}^{t} \xi_{o,i+1,j+\frac{1}{2}}^{t} x_{o,m,i+1,j+\frac{1}{2}}^{t} \bar{v}_{m,i+1,j+\frac{1}{2}}^{t}}{\mu_{o,i+1,j+\frac{1}{2}}^{t}} \]

\[cotox_2 = \sum_{m=1}^{c} \frac{k_{r_0,l+\frac{1}{2},j+\frac{1}{2}}^{t} \xi_{o,l+\frac{1}{2},j+\frac{1}{2}}^{t} x_{o,m,i+\frac{1}{2},j+\frac{1}{2}}^{t} \bar{v}_{m,i+\frac{1}{2},j+\frac{1}{2}}^{t}}{\mu_{o,l+\frac{1}{2},j+\frac{1}{2}}^{t}} \]

\[cotx_1 = \sum_{m=1}^{c} \frac{k_{r_0,l+\frac{1}{2},j+\frac{1}{2}}^{t} \xi_{o,l+\frac{1}{2},j+\frac{1}{2}}^{t} x_{o,m,i+\frac{1}{2},j+\frac{1}{2}}^{t} \bar{v}_{m,i+\frac{1}{2},j+\frac{1}{2}}^{t}}{\mu_{o,l+\frac{1}{2},j+\frac{1}{2}}^{t}} \]

\[cotx_2 = \sum_{m=1}^{c} \frac{k_{r_0,l+\frac{1}{2},j+\frac{1}{2}}^{t} \xi_{o,l+\frac{1}{2},j+\frac{1}{2}}^{t} x_{o,m,i+\frac{1}{2},j+\frac{1}{2}}^{t} \bar{v}_{m,i+\frac{1}{2},j+\frac{1}{2}}^{t}}{\mu_{o,l+\frac{1}{2},j+\frac{1}{2}}^{t}} \]

\[cotoy_1 = \sum_{m=1}^{c} \frac{k_{r_0,l+\frac{1}{2},j+\frac{1}{2}}^{t} \xi_{o,l+\frac{1}{2},j+\frac{1}{2}}^{t} x_{o,m,i+\frac{1}{2},j+\frac{1}{2}}^{t} \bar{v}_{m,i+\frac{1}{2},j+\frac{1}{2}}^{t}}{\mu_{o,l+\frac{1}{2},j+\frac{1}{2}}^{t}} \]
\[
\text{coty}2 = \sum_{m=1}^{c} k^t_{yy,i+\frac{1}{2}j} * \xi^t_{o,j+\frac{1}{2}j} * \chi^t_{o,m,i+\frac{1}{2}j} * \bar{v}^t_{m,i+\frac{1}{2}j}
\]

\[
\text{coty}1 = \sum_{m=1}^{c} k^t_{yy,i+\frac{1}{2}j} * \xi^t_{g,i+\frac{1}{2}j} * \chi^t_{g,m,i+\frac{1}{2}j} * \bar{v}^t_{m,i+\frac{1}{2}j+1}
\]

\[
\text{coty}2 = \sum_{m=1}^{c} k^t_{yy,i+\frac{1}{2}j} * \xi^t_{g,i+\frac{1}{2}j} * \chi^t_{g,m,i+\frac{1}{2}j} * \bar{v}^t_{m,i+\frac{1}{2}j+1}
\]

The discretization of concentration equation (1) at time step \( t + 1 \) can be shown as

\[
\frac{\xi^t_{m,i+\frac{1}{2}j+1} - \xi^t_{m,i+\frac{1}{2}j}}{\Delta t} = q^t_{m,i+\frac{1}{2}j+1} - \text{div}, \quad m = 1, \ldots, c.
\]

with

\[
\text{div} =
\]

\[
x^t_{o,m,i+1,j+\frac{1}{2}} x^t_{o,i+1,j+\frac{1}{2}} - x^t_{o,m,i+\frac{1}{2}j+\frac{1}{2}} x^t_{o,i+\frac{1}{2}j+\frac{1}{2}} - x^t_{o,m,i+1,j+\frac{1}{2}} x^t_{o,i+1,j+\frac{1}{2}} - x^t_{o,m,i+\frac{1}{2}j+\frac{1}{2}} x^t_{o,i+\frac{1}{2}j+\frac{1}{2}}
\]

\[
+ \frac{x^t_{o,m,i+\frac{1}{2}j+\frac{1}{2}} y^t_{o,i+\frac{1}{2}j+\frac{1}{2}} - x^t_{o,m,i+\frac{1}{2}j+\frac{1}{2}} y^t_{o,i+\frac{1}{2}j+\frac{1}{2}}}{y_{j+1} - y_j}
\]

\[
+ \frac{x^t_{g,m,i+1,j+\frac{1}{2}} x^t_{g,i+1,j+\frac{1}{2}} - x^t_{g,m,i+\frac{1}{2}j+\frac{1}{2}} x^t_{g,i+\frac{1}{2}j+\frac{1}{2}}}{x_{i+1} - x_i}
\]

\[
+ \frac{x^t_{g,m,i+\frac{1}{2}j+\frac{1}{2}} x^t_{g,i+\frac{1}{2}j+\frac{1}{2}} - x^t_{g,m,i+\frac{1}{2}j+\frac{1}{2}} x^t_{g,i+\frac{1}{2}j+\frac{1}{2}}}{y_{j+1} - y_j}
\]

Harmonic weighting method is used to derive \( k \) on the edges. For isotropic porous medium, we have \( k_{xy} = 0 \) and \( k_{yx} = 0 \), and
After discretization, a linear system \( Ax = b \) is generated with \( A \) being the coefficient matrix, \( x \) being the unknown vector of pressures, and \( b \) being the
right-hand side vector. If the pressure is imposed on the domain boundary, it is a Dirichlet boundary condition; if the velocity is imposed on the domain boundary, it is a Neumann boundary condition for the pressure.

2.3 Parallelization

There are many domain decomposition strategies to choose from. Some possible strategies are shown in Figure 3. Here, there are two $40^2$ grids divided into four subdomains. In strategy (a), the grid is divided into four subdomains of vertical stripes. However, in strategy (b), the grid is divided into four subdomains of rectangles. The number of cells on the borders of the subdomains is also illustrated in the figure. If computational tasks are well load balanced, the less the communication cost among different domains, the better the scaling performance of the parallel code. In strategy (a), 6*40 cells are in the boundary zones of the four subdomains, while in strategy (b), only $(4*40-4)$ cells are in the boundary zones of the subdomains. Because the communication between two subdomains only happens in the boundary cells, less boundary cells also mean lower communication and overlapping cost (details to be provided later). Thus, strategy (b) is better than (a) in the sense of communication overhead. In other words, the principle to do domain decompositions is to make sure that the total number of the boundary cells is the minimal. Based on the domain decomposition strategy in Figure 3, the case in Section 2.4 is tested. The average communication time for the four processors
is 0.168 second for strategy (b), while the time is 0.206 second for strategy (a). This verifies the advantage of strategy (b).

![Diagram of domain decompositions](image1)

Figure 3 Two strategies of domain decompositions.

![Diagram of pressure field](image2)

Figure 4 A sketch of the pressure field.

There are two types of variables: one is the cell variable and the other one is the edge variable. The cell variables are the scalars such as the pressure, the saturation, the molar fraction, etc. The edge variables include the velocity vector and the scalars derived from the upwind scheme such as the mass density on the edge. Each processor has to compute and store all the cell and edge variables in its own subdomain. However, to derive the edge variables on the boundaries of its subdomain, the processor also has to know the cell
variables in the surrounding cells of its own subdomain. Because such cell variables do not lie on the processor, they must be transferred from the neighboring processors by the communication. However, if the entire cell variables in the surrounding cells are acquired by the communication, the cost is tremendous, which deteriorates the performance of the parallel code. Thus, only the cell variables of the pressure and the molar fraction of the component are communicated. Because the two types of cell variables are inputs of flash calculations, the other cell variables in the surrounding cells can be calculated just by flash calculations. Such cost is called overlapping cost. The overlapping cost is much smaller than the cost to communicate the other cell variables, especially after the application of sparse grid technique (details are mentioned later).

The cell variable of pressure can be observed in Figure 4. In this figure, a subdomain is composed of 3×4 non-overlapping cells (the grey cells) and 14 overlapping cells (the white cells). Thus, the pressure field is denoted as a grid of 5 rows by 6 columns. The pressure values in the grey zone can be calculated by the processor that owns them, while the pressure values in the white zone either need transferring from the neighboring processors or are given as Dirichlet boundary condition. In this work, the porous medium is supposed to be isotropic, thus it is not necessary to take care of the pressure values in the black zone. The data structures of the other cell variables are similar. The edge variable of velocity can be observed in Figure 5. There is still a subdomain of 3×4 cells. The velocity field in the x-direction is denoted as green arrows of three-rows by five-columns. The velocity field in the y-direction can be shown as blue arrows of
four-rows by four-columns. The data structures of the other edge variables hold the similar philosophy.

Figure 5 A sketch map of the velocity field in the x-direction (green arrows) and in the y-direction (blue arrows).

2.4 Performance Evaluation

Table 1 Average run time of the parallel code, P is the number of processors, N is the grid size, and the unit is second.

<table>
<thead>
<tr>
<th>N</th>
<th>P</th>
<th>480^2</th>
<th>640^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>16866.7</td>
<td>16866.7</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>5096.4</td>
<td>5096.4</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>3124.4</td>
<td>3124.4</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>2040.1</td>
<td>3124.4</td>
<td></td>
</tr>
</tbody>
</table>

After the discretization of the pressure equation, a linear system can be derived. The solver library Hypre is used to solve the linear system in the parallel code. Because the grid is a structured grid, the “struct” solvers in Hypre are considered. Our numerical experiment indicates that the struct SMG solver [111] achieves fast convergence when
solving the linear system from the pressure equation, thus the struct SMG solver is used. The structured-grid system interface of Hypre is called by FORTRAN90 and the calling details can be observed in the appendix.

Table 2 Speedup of the parallel code, P is the number of processors, N is the grid size.

<table>
<thead>
<tr>
<th>N</th>
<th>P</th>
<th>480²</th>
<th>640²</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>1.82</td>
<td>1.86</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>3.12</td>
<td>3.31</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>4.81</td>
<td>5.40</td>
</tr>
</tbody>
</table>

The parallel code is implemented with FORTRAN90 and the processor communication mechanism is realized by MPI. The parallel code is tested on the supercomputer Shaheen [112] in KAUST. Shaheen is a 16-rack Blue Gene/P. Each node is equipped with four 32-bit, 850 Mhz PowerPC 450 cores and 4GB DDR memory. The Blue Gene/P architecture provides a 3D point-to-point Blue/Gene P torus network for general-purpose IPC. More details of Shaheen can be referenced to [112]. The module “blugee” is loaded on Shaheen to compile the parallel code. The experiment is based on Case 3 in Table 5. Suppose there is a homogeneous and isotropic domain of 4*4 meters. Initially, there is 50%-mole methane and 50%-mole propane in it. If the temperature is 220 K and the pressure is 2.012*10⁶ Pa, the mixture is in two phases. Gravity is omitted. Methane is injected from the left bottom corner into the domain at a rate of 0.02 mole/m²s to displace the inner substances. The production point is at the right bottom corner of the domain. Neumann boundary condition is imposed except the production point where Dirichlet
boundary condition is imposed and the boundary pressure is $2.012 \times 10^6$ Pa. The total number of time steps is 876, and the simulation time span is 0.01 year. There are two sizes of grid in the tests: one has $480^2$ cells and the other one has $640^2$ cells. The number of processors is set to 64, 128, 256 and 512, respectively.

The average run time and speedup of the parallel code are shown in Table 1 and Table 2, respectively. All the I/O cost is omitted. The speedup bars can be observed in Figure 6. In this figure, the black line stands for the ideal linear speedup, the blue bars stand for the speedup of $480^2$ cells, and the red bars stand for the speedup of $640^2$ cells. It is stipulated that the speedup of 64 processors is 1. From Figure 6, it is found that for either of the grids, a speedup close to the ideal linear speedup can be achieved. However, the speedup goes away from the ideal linear speedup with the increase in the number of processors. In addition to that, the larger the grid size, the better the speedup, which indicates that the parallel code can achieve more advantages in large-scale problems.

The solver time is the sum of the setup time and the iteration time, and it is shown in Table 3. From the table, it can be observed that in all the cases, the solver time decreases with the increase in the number of processors, but the decrease rate is not linear, which means that when the number of processors is doubled, the solver time is not reduced by two times. This is also the main reason that the speedup of total run time deviates from the ideal linear speedup when the number of processors increases, although the other reasons are also admitted such as the communication and overlapping cost.
Table 3 Solver time, P is the number of processors, N is the grid size, and the unit is second.

<table>
<thead>
<tr>
<th>N x P</th>
<th>480²</th>
<th>640²</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>986.8</td>
<td>1390.7</td>
</tr>
<tr>
<td>128</td>
<td>828.4</td>
<td>1101.6</td>
</tr>
<tr>
<td>256</td>
<td>768.5</td>
<td>975.2</td>
</tr>
<tr>
<td>512</td>
<td>754.4</td>
<td>937.2</td>
</tr>
</tbody>
</table>

Figure 6 Speedup of the parallel two-phase compositional flow code.
3. Two-phase Flash Calculations

3.1 Basic Concepts

In the two-phase compositional flow model, flash calculations determine the component compositions in each phase when the overall thermodynamic conditions change. As explained earlier, such calculations are complex and involve the solution of several equations. The two-phase flash calculation can be stated as follows: given a feed of mixture with the total mole $F$, the molar fractions of the components $x_m$, (i.e., $\sum_{m=1}^{c} x_m = 1$), the pressure $p$ and the temperature $T$, the molar fractions of the components in the two phases are the unknowns to be solved. Figure 7 describes the process.

Figure 7 A schematic plot of the two-phase flash calculation at constant $T$ and $p$. 
Flash calculations rest on the assumption of a thermodynamic equilibrium between different components in the two phases. This implies the equality of the chemical potentials (or fugacities) of each species in the two phases,

\[ f_{mo}(T, p, x_{mo}) = f_{mg}(T, p, x_{mg}), \quad m = 1, \ldots, c, \]

where \( f \) stands for the fugacity. Moreover, the material balance for the components provides the mass conservation

\[ Fx_m = x_{mo}L + x_{mg}V, \quad m = 1, \ldots, c, \quad (6) \]

where \( L \) and \( V \) stand for the molar amount of oil and gas phase, respectively. The molar fraction constraints assure that the sum of the molar fractions in a given phase is equal to unity

\[ \sum_{m=1}^{c} x_{mo} = 1, \quad (7) \]
\[ \sum_{m=1}^{c} x_{mg} = 1. \quad (8) \]

In total, there are \((2c + 2)\) equations for the \((2c + 2)\) unknowns \( x_{mg}, x_{mo}, L \) and \( V \). Popular solution methodologies including the successive substitution technique and the bisection method have been extensively used to solve the above equations [113]. After obtaining the values of the \((2c + 2)\) unknowns, one can further compute the molar density of each phase, the mass density of each phase and the isothermal compressibility of each phase by Peng-Robinson Equation of State (PR EOS). Finally, the saturation of the flow, the partial molar volume of each species and the two-phase isothermal compressibility of the flow can be calculated.
Figure 8 Flow chart of the flash calculation.

It is emphasized that at the beginning of the flash calculation, stability analysis has to be done. The routine analyzes the flow and decides whether there are one or two phases in the flow. If it decides that there are two phases in the flow, the routines stated before are carried out. Otherwise, it is decided that whether the flow is in single-gas phase or single-oil phase. Then PR EOS and the other relative equations are used to calculate the parameters in the single phase. The successive substitution technique is not needed in this
condition. The flowchart of the flash calculation can be observed in Figure 8. More
details of the flash calculation can be observed in the following sections.

3.2 Evaluation of Parameters by Peng-Robinson Equation of State

In flash calculations, the equation of state is used to compute the physical parameters
in each phase. Thus, it is the base to carry out the other routines. Suppose there are \( c \)
components in the flow. Let \( N_m \) be the number of moles of the \( m \)th component and then
the total molar amount of the flow is

\[
N = \sum_{m=1}^{c} N_m .
\]

The molar fraction of each component is defined as

\[
x_m = \frac{N_m}{N}, \quad m = 1, 2, \ldots, c.
\]

In PR EOS, the multicomponent mixing principle is as below

\[
a = \sum_{m=1}^{c} \sum_{n=1}^{c} x_m x_n (1 - K_{mn}) \sqrt{a_m a_n}, \quad (9)
\]

\[
b = \sum_{m=1}^{c} x_m b_m , \quad (10)
\]

where \( K_{mn} \) is a binary interaction parameter between components \( m \) and \( n \), and \( a_m \) and
\( b_m \) are empirical factors of the pure component \( m \). The interaction parameters account
for molecular interactions between two unlike molecules. By definition, \( K_{mn} \) is zero
when \( m \) and \( n \) stand for the same component, small when \( m \) and \( n \) represent components
that do not differ much from each other (e.g., methane and propane), and large when \( m \)
and \( n \) represent components that are substantially different (e.g., methane and carbon
dioxide). Ideally, $K_{mn}$ depends on pressure, temperature and identities of components $m$ and $n$.

The factors $a_m$ and $b_m$ can be computed from

$$a_m = \Omega_{ma} \alpha_m \frac{R^2 T_{cm}^2}{p_{cm}},$$

$$b_m = \Omega_{mb} \frac{RT_{cm}}{p_{cm}},$$

where $R$ is the universal gas constant, $T_{cm}$ is the critical temperature of component $m$, and $p_{cm}$ is the critical pressure of component $m$. The PR EOS parameters $\Omega_{ma}$ and $\Omega_{mb}$ are given by

$$\Omega_{ma} = 0.45724,$$

$$\Omega_{mb} = 0.077796,$$

$$\alpha_m = (1 + \lambda_m (1 - \sqrt{T/T_{cm}}))^2.$$  

$\lambda_m$ is based on vapor pressure data of hydrocarbons and $T$ is the temperature. Suppose $\omega_m$ is the acentric factor of component $m$, then

$$\lambda_m = 0.37464 + 1.5423 \omega_m - 0.26992 \omega_m^2, \quad 0 < \omega_m < 0.5,$$

$$\lambda_m = 0.3796 + 1.485 \omega_m - 0.1644 \omega_m^2 + 0.01667 \omega_m^3, \quad 0.1 < \omega_m < 2.0.$$  

$\omega_m$ approximately expresses the deviation of the shape of a molecule from a sphere. The acentric factor of component $m$ is defined as below

$$\omega_m = -\log \left( \frac{p_{sat}}{p_{cm}} \right)_{T_{rm}=0.7} - 1.$$  

where $T_{rm}$ represents the reduced temperature of component $m$ and is defined as

$$T_{rm} = \frac{T}{T_{cm}}.$$  

If the vapor pressure data is represented by
\[ \log p^{\text{sat}} = \beta + \frac{\tau}{T}, \]

where \( \beta \) and \( \tau \) are constants, and \( p^{\text{sat}} \) is the saturation pressure at absolute temperature \( T \), then

\[ \omega_m = \frac{3}{7} \left( \frac{\log(p_{cm}/14.695)}{(T_{cm}/T_{bm} - 1)} \right) - 1, \]

where \( p_{cm} \) is in PSI absolute and \( T_{bm} \) is the normal boiling point of component \( m \) with the same absolute units as \( T_{cm} \).

Define

\[ A = \frac{ap}{R^2T^2}, \quad (11) \]
\[ B = \frac{bp}{RT}, \quad (12) \]

where the pressure \( p \) is given by the PR EOS

\[ p = \frac{RT}{v - b} - \frac{a}{v(v + b) + b(v - b)}, \]

where \( v \) is the molar volume of the flow. Introduce the compressibility factor

\[ Z = \frac{pv}{RT}, \]

and then PR’s cubic equation in \( Z \) is

\[ Z^3 - (1 - B)Z^2 + (A - 2B - 3B^2)Z - (AB - B^2 - B^3) = 0. \quad (13) \]

The equation has three roots. When only one root is real and positive, the root is selected as the value of compressibility factor. When there are two or three real and positive roots, the largest positive root is selected if the flow is in gas phase, and the smallest positive root is selected if the flow is in oil phase. If the values of \( Z \) do not belong to any of the above cases, it means that the computation has some error.
The flow molar density is
\[ \xi = \frac{p}{RTZ} \]  (14)
and the flow mass density is
\[ \rho = \xi \sum_{m=1}^{c} x_m W_m, \]
where \( W_m \) is the molar weight of component \( m \). Generally, the PR EOS and other similar cubic equations reliably represent the vapor pressure of pure substances because vapor pressure data are used to obtain the parameter \( \alpha_m \). Thus, PR EOS has a good prediction to the gas density, but it has a bad prediction to the oil density. The error of predicted value of oil density from PR EOS is due to the deviation of predicted oil volume from real oil volume, thus a modification to the predicted oil volume is added
\[ v^{\text{true}} = v^{\text{EOS}} + cc. \]
\( cc \) is the volume-translation parameter, \( v^{\text{EOS}} \) is the specific molar volume calculated with the PR EOS and \( v^{\text{true}} \) is the true molar volume of oil phase. Hoyos [114] proposed an expression of \( cc \) for pure substance. With the mixing rule of volume-translation parameter, the expression of \( cc \) for mixture can be derived as below
\[ cc = \sum_{m=1}^{c} x_m cc_m W_m, \]
where
\[ cc_m = cc_{1m} + cc_2(T_{rm} - cc_3)^2, \quad m = 1, 2, ..., c, \]
\( cc_3 \) can be understood as the value of reduced temperature where the maximum correction is required and \( cc_{1m} \) as the value of that maximum correction. The value of
$cc_3$ is obtained as the average of reduced temperature where the maximum deviation of the original PR EOS for saturated oil volume occurs, yielding $cc_3 = 0.89$. On the other hand, at the particular reduced temperature value where the PR EOS does not need correction, the above equation becomes

$$\frac{cc_{1m}}{cc_2} = -(T_{rm} - 0.89)^2.$$  

The ratio $\frac{cc_{1m}}{cc_2}$ can be calculated as a polynomial function of the acentric factor

$$\frac{cc_{1m}}{cc_2} = 110.07\omega_m^4 - 83.807\omega_m^3 + 18.926\omega_m^2 - 1.6348\omega_m - 0.0066.$$  

The $cc_2$ constant remains as a degree of freedom, which is found as the value that minimizes the average of absolute value of relative deviations for all substances, yielding $cc_2 = 2.013645 \times 10^{-3} m^3/kg$. With the value of $cc_2$ and the value of $\frac{cc_{1m}}{cc_2}$, one can get the value of $cc_{1m}$. Then one can get the value of $cc$. Finally, the corrected molar volume of oil phase is

$$\tilde{\xi}_{corr} = 1/(\frac{1}{\xi_{EOS}} + cc).$$

And the corrected mass density of oil phase is

$$\rho^\text{corr} = \tilde{\xi}_{corr} \sum_{m=1}^{c} x_m W_m.$$  

Note that the flow isothermal compressibility is

$$C_f = \frac{1}{\xi} \frac{\partial \xi}{\partial p}. \quad (15)$$  

From (14) there is

$$\frac{\partial \xi}{\partial p} = \frac{1}{RTZ} - \frac{p}{RTZ^2} \frac{\partial Z}{\partial p}.$$  

By (15), there is
\[ C_f = \frac{1}{p} - \frac{1}{Z} \frac{\partial Z}{\partial p}. \]  

(16)

Thus, to get \( C_f \), one only needs to get \( \frac{\partial Z}{\partial p} \). From (11) and (12), there is

\[ \frac{\partial A}{\partial p} = \frac{a}{RT^2}, \]  

(17)

\[ \frac{\partial B}{\partial p} = \frac{b}{RT^2}. \]  

(18)

When doing implicit differentiation on (13), one can get

\[ \frac{\partial Z}{\partial p} = \frac{-\frac{\partial B}{\partial p} Z^2 + \frac{\partial A}{\partial p} - 2(1+3B) \frac{\partial B}{\partial p} Z - \frac{\partial A B + (A-2B-3B^2) \frac{\partial B}{\partial p}}{3Z^2 - 2(1-B)Z + (A-2B-3B^2)}}{3Z^2 - 2(1-B)Z + (A-2B-3B^2)}. \]  

(19)

Then from (16) to (19) one can derive \( C_f \).

In a word, given pressure, temperature and compositions of a phase, PR EOS can output the mass density, the molar density and the isothermal compressibility of the phase. PR EOS can also output the other physical parameters that needed by the other routines.

3.3 Computation of Viscosity

Using the parameters output from PR EOS, one can compute the viscosity of the flow. Because of the different characteristics of gas and oil phase, gas phase viscosity and oil phase viscosity are computed, respectively. The correlation formula due to Lohrenz, Bray and Clark [115] is used to compute the oil phase viscosity because it executes fast in computer code. Firstly, component \( m \)'s viscosity at low pressure \( \mu_m^* \) needs to be computed. If the unit of temperature is K and the unit of pressure is ATM, the unit of viscosity is centipoise (cp) in the following expression

\[ \mu_m^* = \begin{cases} 
34 \times 10^{-5} \frac{T_{rm}^{0.94}}{\eta_m}, & \text{if } T_{rm} < 1.5, \\
17.78 \times 10^{-5} \frac{(4.58T_{rm} - 1.67)^{5/8}}{\eta_m}, & \text{if } T_{rm} \geq 1.5,
\end{cases} \]
where

\[ \eta_m = \frac{1}{\frac{1}{2} \frac{T_{cm}^6}{W_m P_{cm}^{3/2}}}. \]

Then the oil phase viscosity at low pressure is

\[ \mu^* = \frac{\sum_{m=1}^{c} x_m \mu_m^* W_m^2}{\sum_{m=1}^{c} x_m W_m^2}. \]

The reduced density of one component is

\[ \xi_r = \frac{\xi}{\xi_c}, \]

where

\[ \xi_c = \frac{1}{\sum_{m=1}^{c} x_m V_{cm}}. \]

Here, \( V_{cm} \) is the critical volume of component \( m \). For light and medium oil components, one can get the value of \( V_{cm} \) from the literature. However, for heavy oil components \((C_7+)\), the critical volume should be computed as below

\[ V_{cm} = 21.573 + 0.015122 W_m - 27.656 \rho_m + 0.070615 W_m \rho_m. \]

\( \rho_m \) is the mass density of heavy oil component \( m \) in g/cm\(^3\) and the unit of \( V_{cm} \) is ft\(^3\)/lb in the above equation.

Finally, the oil phase viscosity \( \mu \) can be computed as below

\[ \mu = \frac{(0.1023 + 0.023364 \xi_r + 0.058533 \xi_r^2 - 0.40758 \xi_r^3 + 0.0093324 \xi_r^4)^4 - 10^{-4}}{\eta} + \mu^*, \]

where
\[ \eta = \frac{(\sum_{m=1}^{c} x_m T_{cm})^{\frac{1}{6}}}{(\sum_{m=1}^{c} x_m W_m)\left(\sum_{m=1}^{c} x_m p_{cm}\right)^{\frac{2}{3}}} \]

with the unit of \( \mu \) is cp.

Alternatively, the gas phase viscosity is evaluated based on an estimation of the gas density using the real gas law. The gas phase viscosity is computed by the Lee-Gonzalez correction [116]

\[ \mu = \frac{\exp(FF)\mu_c}{T_{red}}, \]

where

\[ FF = AA + BB T_{red} + CCT^2_{red} + DDT^3_{red}, \]

\[ AA = AA_0 + AA_1 p_{red} + AA_2 p_{red}^2 + AA_3 p_{red}^3, \]

\[ BB = BB_0 + BB_1 p_{red} + BB_2 p_{red}^2 + BB_3 p_{red}^3, \]

\[ CC = CC_0 + CC_1 p_{red} + CC_2 p_{red}^2 + CC_3 p_{red}^3, \]

\[ DD = DD_0 + DD_1 p_{red} + DD_2 p_{red}^2 + DD_3 p_{red}^3, \]

with the constants given by

\[ AA_0 = -2.4621182, \]

\[ AA_1 = 2.97054714, \]

\[ AA_2 = -0.286264054, \]

\[ AA_3 = 8.05420522 \times 10^{-3}, \]

\[ BB_0 = 2.80860949, \]

\[ BB_1 = -3.49803305, \]

\[ BB_2 = 0.36037302, \]
\[ BB_3 = -1.04432413 \times 10^{-2}, \]
\[ CC_0 = -0.793385684, \]
\[ CC_1 = 1.39643306, \]
\[ CC_2 = -0.149144925, \]
\[ CC_3 = 4.41015512 \times 10^{-3}, \]
\[ DD_0 = 0.0839387178, \]
\[ DD_1 = 0.186408848, \]
\[ DD_2 = 0.0203367881, \]
\[ DD_3 = 6.09579263 \times 10^{-4}. \]

\( p_{\text{red}} \) is the reduced pressure of mixture gas, \( T_{\text{red}} \) is the reduced temperature of mixture gas, and they are defined as

\[
 p_{\text{red}} = \frac{p}{\sum_{m=1}^{c} x_m p_{cm}} \times 10^{-5},
\]
\[
 T_{\text{red}} = \frac{T}{\sum_{m=1}^{c} x_m T_{cm}}.
\]

\( \mu_c \) is expressed as

\[
 \mu_c = \left( 7.43 + 0.0133 \sum_{m=1}^{c} x_m W_m \right) \times \frac{(1.8T)^{1.5}}{1.8T + 75.4 + 13.9 \sum_{m=1}^{c} x_m W_m} \times 10^{-4},
\]

with the unit of viscosity being cp and the unit of \( W_m \) being g/m³.

### 3.4 Computation of Partial Molar Volume

In flash calculations, there are three conditions to compute partial molar volume of each component. If the flow contains only one substance, there is only one phase. The partial molar volume of the substance is the reciprocal of its mass density.
If the flow contains multiple components, the flow can be in single phase or two phases. If the flow is in single phase, according to Lin and Thomas [117], the partial molar volume of component \( m \) is estimated from PR EOS by

\[
\bar{v}_m = \left[ \frac{RT}{v_a - b} \left( 1 + \frac{b_m}{v_a - b} \right) - \frac{2 \sum x_m K_{mi} - 2 a b_m (v_a - b)}{v_a (v_a + b) + b (v_a - b)} \right] \left[ \frac{RT}{(v_a - b)^2} - \frac{2 a (v_a + b)}{(v_a (v_a + b) + b (v_a - b))^2} \right],
\]

\( \alpha = o, g, \ m = 1, \ldots, c. \)

\( v_o \) and \( v_g \) are the molar volume of oil phase and gas phase respectively, and it can be computed as

\[
v_\alpha = \frac{Z_\alpha RT}{p}, \ \alpha = o, g.
\]

\( Z_\alpha \) is the compressibility factor of phase \( \alpha \).

If the flow falls in two phases, the definition of partial molar volume has to be used to derive its value. Suppose the molar amount of component \( m \) is \( N_m \), and the molar amount of component \( m \) is increased by a very small value \( \delta \), for example, \( 10^{-5} \times N_m \); Keep the molar amounts of the other components constant, and do the flash calculation to get a new volume of the flow \( V_{new} \). If the old volume of the flow is \( V_{old} \), the partial molar volume of component \( m \) is

\[
\bar{v}_m = \frac{V_{new} - V_{old}}{\delta}, m = 1, \ldots, c.
\]

### 3.5 Computation of Two-phase Isothermal Compressibility

The expression of the single-phase isothermal compressibility has been studied in (16) to (19). The isothermal compressibility of a two-phase compositional flow is defined as
\[ C_f = -\frac{1}{V_t} \left( \frac{\partial V_t}{\partial p} \right)_{T,N}. \]

In the equation, \( N = (N_1, N_2, ..., N_c) \) represents the total molar amounts of the components in both phases, and the derivative term on the right side represents the change in the total flow volume, \( V_t \), caused by a small change in the pressure at constant temperature and constant total molar amounts. The expression of \( V_t \) can be shown as

\[ V_t = V_o + V_g, \]

where \( V_o \) and \( V_g \) are oil and gas phase volume, respectively. From the relation

\[ pV_\alpha = Z_\alpha N_{t\alpha} RT, \]

where \( N_{t\alpha} \) is the total molar amount of phase \( \alpha \), one can have

\[ V_t = \frac{RT}{p} \sum_{\alpha=o,g} Z_\alpha N_{t\alpha}. \]

Taking the derivative of the above equation with respect to pressure, and holding temperature and overall molar amount constant yield

\[ \left( \frac{\partial V_t}{\partial p} \right)_{T,N} = -\frac{RT}{p^2} \sum_{\alpha=o,g} Z_\alpha N_{t\alpha} + \frac{RT}{p} \sum_{\alpha=o,g} \left[ N_{t\alpha} \left( \frac{\partial Z_\alpha}{\partial p} \right)_{T,N} + Z_\alpha \left( \frac{\partial N_{t\alpha}}{\partial p} \right)_{T,N} \right]. \]  

(20)

The unknown derivatives on the right side of the above equation are \( \left( \frac{\partial Z_\alpha}{\partial p} \right)_{T,N} \) and \( \left( \frac{\partial N_{t\alpha}}{\partial p} \right)_{T,N} \). The change in the total molar amount of phase \( \alpha \) with pressure can be expressed as the sum of the changes in molar amount of each component in phase \( \alpha \) with pressure

\[ \left( \frac{\partial N_{t\alpha}}{\partial p} \right)_{T,N} = \sum_{m=1}^{c} \left( \frac{\partial N_{m\alpha}}{\partial p} \right)_{T,N}, \quad \alpha = o, g, \]
where \( N_{ma} \) is the molar amount of component \( m \) in phase \( \alpha \). There are \( 2c \) unknown derivatives \( \left( \frac{\partial N_{ma}}{\partial p} \right)_{T,N} \) in the above equations, thus \( 2c \) equations are needed. These equations come from \( c \) material balance equations

\[
N_m = N_{mo} + N_{mg}, \quad m = 1, \ldots, c,
\]

which yields

\[
\left( \frac{\partial N_{mo}}{\partial p} \right)_{T,N} + \left( \frac{\partial N_{mg}}{\partial p} \right)_{T,N} = 0, \quad m = 1, \ldots, c. \tag{21}
\]

The remaining \( c \) equations result from the equilibrium conditions before and after the change in pressure of the system

\[
\left( \frac{\partial f_{mg}}{\partial p} - \frac{\partial f_{mo}}{\partial p} \right)_{T,N} = \left[ \sum_{l=1}^{c} \left( \frac{\partial f_{mg}}{\partial N_{lg}} \right)_{T,N_{lg}} \left( \frac{\partial N_{lg}}{\partial p} \right)_{T,N} - \sum_{l=1}^{c} \left( \frac{\partial f_{mo}}{\partial N_{lo}} \right)_{T,N_{lo}} \left( \frac{\partial N_{lo}}{\partial p} \right)_{T,N} \right] + \left[ \left( \frac{\partial f_{mg}}{\partial p} \right)_{T,N_{og}} - \left( \frac{\partial f_{mo}}{\partial p} \right)_{T,N_{oo}} \right] = 0, \quad m = 1, \ldots, c, \tag{22}
\]

where \( N_{lg} = (N_{1g}, \ldots, N_{l-1,g}, N_{l+1,g}, \ldots, N_{cg}) \) and \( N_g = (N_{1g}, \ldots, N_{cg}) \). A similar definition applies to \( N_{lo} \) and \( N_o \).

Combining (21) and (22), one can have

\[
\sum_{l=1}^{c} \left( \frac{\partial f_{mg}}{\partial N_{lg}} + \frac{\partial f_{mo}}{\partial N_{lo}} \right) \left( \frac{\partial N_{lg}}{\partial p} \right) = \left[ \left( \frac{\partial f_{mg}}{\partial p} \right)_{T,N_{g}} - \left( \frac{\partial f_{mo}}{\partial p} \right)_{T,N_{o}} \right], \quad m = 1, \ldots, c.
\]

After solving for \( \left( \frac{\partial N_{la}}{\partial p} \right)_{T,N} \), \( \alpha = o, g \), the following expression gives the various terms contributing to \( \left( \frac{\partial Z_{\alpha}}{\partial p} \right)_{T,N} \)

\[
\left( \frac{\partial Z_{\alpha}}{\partial p} \right)_{T,N} = \left( \frac{\partial Z_{\alpha}}{\partial p} \right)_{T,N_{\alpha}} + \sum_{l=1}^{c} \left( \frac{\partial Z_{\alpha}}{\partial N_{la}} \right) \left( \frac{\partial N_{la}}{\partial p} \right), \quad \alpha = o, g.
\]

PR EOS can be used to calculate \( \left( \frac{\partial Z_{\alpha}}{\partial p} \right)_{T,N_{\alpha}} \) and \( \left( \frac{\partial Z_{\alpha}}{\partial N_{la}} \right) \) terms on the right side of the above equation.
From PR’s cubic equation (13), it is known that
\[
\frac{\partial Z_\alpha}{\partial A_\alpha} = \frac{B_\alpha - Z_\alpha}{3Z_\alpha^2 - 2Z_\alpha(1 - B_\alpha) + (A_\alpha - 3B_\alpha^2 - 2B_\alpha)}, \quad \alpha = o, g.
\]
The definition of \( A_\alpha \) and \( B_\alpha \) comes from (11) and (12). One can also get
\[
\frac{\partial Z_\alpha}{\partial B_\alpha} = \frac{-Z_\alpha^2 + 2(3B_\alpha + 1)Z_\alpha + (A_\alpha - 2B_\alpha - 3B_\alpha^2)}{3Z_\alpha^2 - 2(1 - B_\alpha)Z_\alpha + (A_\alpha - 3B_\alpha^2 - 2B_\alpha)}, \quad \alpha = o, g,
\]
From (9) and (10), one can get another expression
\[
\frac{\partial a_\alpha}{\partial x_{m\alpha}} = 2 \sum_{n=1}^{c} x_{n\alpha} (1 - K_{mn}) \sqrt{a_m a_n}, \quad \alpha = o, g, m = 1, ..., c.
\]
Then
\[
\frac{\partial A_\alpha}{\partial x_{m\alpha}} = \frac{p}{(RT)^2} \frac{\partial a_\alpha}{\partial x_{m\alpha}}, \quad \alpha = o, g, m = 1, ..., c.
\]
\[
\frac{\partial B_\alpha}{\partial x_{m\alpha}} = \frac{p b_m}{RT}, \quad \alpha = o, g, m = 1, ..., c.
\]
There also exist
\[
\frac{\partial Z_\alpha}{\partial x_{m\alpha}} = \frac{1}{Z_\alpha - B_\alpha} \left( \frac{\partial Z_\alpha}{\partial x_{m\alpha}} - \frac{\partial Z_\alpha}{\partial x_{m\alpha}} \right), \quad \alpha = o, g, m = 1, ..., c,
\]
\[
\frac{\partial [\ln(Z_\alpha - B_\alpha)]}{\partial x_{m\alpha}} = \frac{1}{Z_\alpha - B_\alpha} \left( \frac{\partial Z_\alpha}{\partial x_{m\alpha}} - \frac{\partial Z_\alpha}{\partial x_{m\alpha}} \right), \quad \alpha = o, g, m = 1, ..., c,
\]
\[
\frac{\partial [\ln(Z_\alpha + 2.414B_\alpha)]}{\partial x_{m\alpha}} = \frac{1}{Z_\alpha + 2.414B_\alpha} \left( \frac{\partial Z_\alpha}{\partial x_{m\alpha}} - \frac{\partial Z_\alpha}{\partial x_{m\alpha}} \right), \quad \alpha = o, g, m = 1, ..., c.(25)
\]
\[
\frac{\partial [\ln(Z_\alpha - 0.414B_\alpha)]}{\partial x_{m\alpha}} = \frac{1}{Z_\alpha - 0.414B_\alpha} \left( \frac{\partial Z_\alpha}{\partial x_{m\alpha}} - \frac{\partial Z_\alpha}{\partial x_{m\alpha}} \right), \quad \alpha = o, g, m = 1, ..., c.(25)
\]
\[
\frac{\partial}{\partial x_{n\alpha}} \left[ \frac{2\sum_{l=1}^{c} x_{l\alpha} a_{ml}}{a_\alpha} b_m \right] = \frac{2}{a_\alpha} a_{mn} - 2 \sum_{l=1}^{c} x_{l\alpha} a_{ml} \frac{\partial a_\alpha}{\partial x_{n\alpha}} + \frac{b_m b_n}{b_\alpha^2}, \quad \alpha = o, g, m = 1, ..., c, n = 1, ..., c. (27)
\]
Here \( a_{ml} = (1 - K_{ml}) \sqrt{a_m a_l} \). If one defines the function

\[
\text{right}(m, \alpha) = \frac{b_m}{b_\alpha} (Z_\alpha - 1) - \ln(Z_\alpha - B_\alpha)
\]

\[
- \frac{A_\alpha}{2 \sqrt{2 B_\alpha}} \left[ 2 \sum_{l=1}^c x_{l\alpha} a_{ml} - b_m \right] \frac{Z_\alpha + 2.414 B_\alpha}{Z_\alpha - 0.414 B_\alpha}, \alpha = o, g, m = 1, \ldots, c.
\]

The fugacity of component \( m \) in phase \( \alpha \) is defined as

\[
f_{m\alpha} = x_{m\alpha} p \text{e}^{\text{right}(m, \alpha)}.
\]

Then one can get

\[
\frac{\partial f_{m\alpha}}{\partial x_{n\alpha}} = \begin{cases} 
  x_{m\alpha} p \text{e}^{\text{right}(m, \alpha)} \frac{\partial \text{right}(m, \alpha)}{\partial x_{n\alpha}}, & \text{if } m \neq n, \\
  x_{m\alpha} p \text{e}^{\text{right}(m, \alpha)} \frac{\partial \text{right}(m, \alpha)}{\partial x_{n\alpha}} + p \text{e}^{\text{right}(m, \alpha)}, & \text{if } m = n,
\end{cases}
\]

\[\alpha = o, g, m = 1, \ldots, c, n = 1, \ldots, c.\]

From (23) to (27), one can get \( \frac{\partial \text{right}(m, \alpha)}{\partial x_{n\alpha}} \), then one can get \( \frac{\partial f_{m\alpha}}{\partial x_{n\alpha}} \) easily. From \( \frac{\partial f_{m\alpha}}{\partial x_{n\alpha}} \), one knows that

\[
\frac{\partial f_{m\alpha}}{\partial N_{n\alpha}} = \frac{\partial f_{m\alpha}}{\partial x_{n\alpha}} \sum_{l=1}^c N_{l\alpha}, \alpha = o, g, m = 1, \ldots, c, n = 1, \ldots, c.
\]

Because

\[
\frac{\partial Z_\alpha}{\partial p} = \frac{\partial Z_\alpha}{\partial A_\alpha} \frac{\partial A_\alpha}{\partial p} + \frac{\partial Z_\alpha}{\partial B_\alpha} \frac{\partial B_\alpha}{\partial p}.
\]

\[
= \frac{a_\alpha}{(RT)^2} \frac{3Z_\alpha^2 - 2Z_\alpha (1 - B_\alpha) + (A_\alpha - 3B_\alpha^2 - 2B_\alpha)}{B_\alpha - Z_\alpha} + \frac{b_\alpha}{RT} \frac{-Z_\alpha^2 + 2(3B_\alpha + 1)Z_\alpha + (A_\alpha - 2B_\alpha - 3B_\alpha^2)}{3Z_\alpha^2 - 2(1 - B_\alpha)Z_\alpha + (A_\alpha - 2B_\alpha - 3B_\alpha^2)}.
\]

\[\alpha = o, g, \] (28)

\[
\frac{\partial |\ln(z_{a\alpha}^{-1})|}{\partial p} = \frac{b_m}{b_\alpha} \frac{\partial z_\alpha}{\partial p}, m = 1, \ldots, c, \alpha = o, g, \] (29)

\[
\frac{\partial |\ln(z_{a-B_\alpha})|}{\partial p} = \frac{1}{z_{a-B_\alpha}} \frac{\partial z_\alpha}{\partial p} - \frac{b_\alpha}{RT}, \alpha = o, g, \] (30)
\[
\frac{\partial \left( \frac{A_\alpha}{\sqrt{2} B_\alpha} \right)}{\partial p} = \frac{1}{2 \sqrt{2} B_\alpha} \frac{\partial A_\alpha}{\partial p} - \frac{A_\alpha}{2 \sqrt{2} B_\alpha} \frac{1}{\partial B_\alpha} \frac{\partial B_\alpha}{\partial p}, \alpha = 0, g,
\]

\[
\frac{\partial \left[ \sum_{m=1}^{c} x_\alpha a m \frac{b_m}{b_a} \right]}{\partial p} = 0, m = 1, ..., c, \alpha = 0, g,
\]

\[
\frac{\partial [\ln \frac{Z_\alpha + 2.414 B_\alpha}{Z_\alpha - 0.414 B_\alpha}]}{\partial p} = \frac{(\frac{\partial Z_\alpha}{\partial p} - 0.414 \frac{\partial B_\alpha}{\partial p})(Z_\alpha + 2.414 B_\alpha) - (\frac{\partial Z_\alpha}{\partial p} + 2.414 \frac{\partial B_\alpha}{\partial p})(Z_\alpha - 0.414 B_\alpha)}{(Z_\alpha + 2.414 B_\alpha)(Z_\alpha - 0.414 B_\alpha)}, \alpha = 0, g,
\]

From (28) to (33), one can get \( \frac{\partial f_{ma}}{\partial p} \). Then with \( \frac{\partial f_{ma}}{\partial x_{na}} \) and \( \frac{\partial f_{ma}}{\partial p} \), one can get \( \frac{\partial N_{ta}}{\partial p} \). By substituting the value of \( \frac{\partial N_{ta}}{\partial p} \) to equation (20), one can get the value of isothermal compressibility \( C_f \).

### 3.6 Successive Substitution Technique

The technique is used in the routine of flash calculations to derive the molar fractions of the components in the oil or gas phase. The equilibrium ratio \( K_m \) is defined as

\[
K_m = \frac{x_{mg}}{x_{mo}}, \quad m = 1, ..., c.
\]

Then

\[
x_{mg} = K_m x_{mo}, \quad m = 1, ..., c.
\]

Substitute (34) to (6), and there is

\[
L x_{mo} + V K_m x_{mo} = F x_m, \quad m = 1, ..., c.
\]

Taking the summation over \( m = 1, ..., c \), there is

\[
L = F - V.
\]

Combining (35) and (36) and solving for \( x_{mo} \) result in

\[
x_{mo} = \frac{x_m}{1 + (K_m-1)(V/F)}, \quad m = 1, ..., c.
\]

Similarly, one also obtains
\[ x_{mg} = \frac{K_m x_m}{1 + (K_m - 1)(V/F)}, \quad m = 1, \ldots, c. \]  \hspace{1cm} (38)

Combining (37), (38), (7) and (8), there is

\[ \sum_{m=1}^{c} \frac{(K_m - 1)x_m}{1 + (K_m - 1)(V/F)} = 0. \]

Define \( \alpha = V/F \), and the above equation becomes Rachford-Rice expression

\[ h(\alpha) = \sum_{m=1}^{c} \frac{(K_m - 1)x_m}{1 + \alpha(K_m - 1)} = 0. \]  \hspace{1cm} (39)

The equilibrium ratios \( K_m \) are functions of \( T, p \) and the compositions of one of the phases. Moreover, it is known that the fugacity of component \( m \) in a phase is given by

\[ f_{mo} = x_{mo} \varphi_{mo} p, \quad m = 1, \ldots, c, \]

\[ f_{mg} = x_{mg} \varphi_{mg} p, \quad m = 1, \ldots, c, \]

where \( \varphi_{mo} \) and \( \varphi_{mg} \) are oil and gas phase fugacity coefficients, respectively. At equilibrium, there is

\[ f_{mo} = f_{mg}, \quad m = 1, \ldots, c. \]

Therefore,

\[ K_m = \frac{x_{mg}}{x_{mo}} = \frac{\varphi_{mo}}{\varphi_{mg}}, \quad m = 1, \ldots, c. \]

Here,

\[
\varphi_m = \exp \left( \frac{b_m}{b} (Z - 1) - \ln(Z - B) - \frac{A}{2\sqrt{2B}} \left( 2 \sum_{n=1}^{c} x_n (1 - K_{mn}) \sqrt{a_m a_n} - \frac{b_m}{b} \right) \ln(Z + 2.414B) \right) \frac{Z - 0.414B}{\sqrt{2B}}.
\]

The equation is the general form of the fugacity coefficient. After getting \( K_m \), the values of \( x_{mo} \) and \( x_{mg} \) can be calculated by solving (7) and (8).
To solve $K_m$ iteratively, the following five steps are followed.

1) Guess the initial values of $K_m$ at fixed temperature and pressure. The Wilson correlation can be used for this purpose. In the correlation, based on the direct application of Raoult’s law, $K_m$ is given by

$$\ln K_m = 5.37(1 + \omega_m) \left(1 - \frac{T_{cm}}{T}\right) + \ln \left(\frac{p}{p_{cm}}\right).$$

2) Solve $\alpha$ in (39) by the bisection method.

3) Calculate $x_{mo}$ and $x_{mg}$ from (37) and (38).

4) Calculate $\varphi_{mo}$ and $\varphi_{mg}$, and then update $K_m$ by

$$K_m^{new} = K_m^{old} \exp \left(-\ln \left(\frac{f_{mg}}{f_{mo}}\right)\right).$$

5) Test convergence using the criterion. The tolerance can be adjusted according to the code.

$$\frac{1}{c} \sum_{m=1}^{c} \left(\ln \frac{f_{mg}}{f_{mo}}\right)^2 < 10^{-12}.$$

If the convergence criterion is not satisfied, Step 2 through 5 are repeated. In the case of a poor initial guess of $K_m$, the successive substitution technique may fail. For example, $\alpha$ may be calculated to be outside the interval $[0,1]$. To make the method always go through to the end, stability analysis must be done at first. If the flow is decided in single phase, the routine of the flash calculation is omitted, and PR EOS is directly used to get the parameters; if the flow is decided in two phases, the $K_m$ value given by stability analysis is used as the initial guess of $K_m$ in the successive substitution technique. The details of stability analysis will be given later.
It is emphasized that compared with Newton’s method, bisection method always works. Newton’s method needs to know the slope of the curve \( h(\alpha) \). When the absolute value of the slope is very small, the solution of Newton’s method may go beyond the true solution area and fail.

3.7 Stability Analysis

Suppose the mixture is in single phase, and the Gibbs free energy of the homogeneous single-phase system is

\[
G^1 = \sum_{m=1}^{c} N_m \mu_m(z, T, p),
\]

where \( z = (z_1, z_2, ..., z_{c-1}) \) and \( \mu_m \) is the chemical potential of component \( m \). If a very small amount of a second phase is formed at the same pressure and temperature with mole \( n_i \), and \( n_i \ll N_i \), the Gibbs free energy of the new system \( II \) is

\[
G^{II} = G(n, T, p) + G(N - n, T, p),
\]

where \( G(n, T, p) \) and \( G(N - n, T, p) \) are the Gibbs free energies of the two phases of system \( II \) and \( n = (n_1, n_2, ..., n_c) \).

\[
G(n, T, p) = \sum_{m=1}^{c} n_m \mu_m(x, T, p),
\]

\[
x_i = \frac{n_i}{\sum_{m=1}^{c} n_i} = n_i / n.
\]

Apply the Taylor’s series expansion, and there is

\[
G(N - n, T, p) \approx G^1(N, T, p) - \sum_{m=1}^{c} \left( \frac{\partial G}{\partial N_m} \right)_{N_m} n_m
\]
\[ \Delta G = G^\| - G^\perp = \sum_{m=1}^{c} n_m \mu_m(z, T, p). \]

Thus, the change in Gibbs free energy from I to II is

\[ \Delta G = G^\| - G^\perp = \sum_{m=1}^{c} n_m \left[ \mu_m(x, T, p) - \mu_m(z, T, p) \right]. \]  \hspace{1cm} (40)

If \( \Delta G \geq 0 \) for all feasible values of \( x \), then the original state at I is stable and \( G^\perp \) cannot be further reduced. If \( \Delta G < 0 \) for any feasible value of \( x \), the original state is unstable, and the single phase splits into more than one phase (suppose there are two phases in this work).

Divide (40) by \( n \), there is

\[ \Delta g(x) = \frac{\Delta G}{n} = \sum_{m=1}^{c} x_m \left[ \mu_m(x) - \mu_m(z) \right], \]

where \( \Delta g(x) \) is a molar quantity. Note that \( T \) and \( p \) dependencies have been dropped because \( T \) and \( p \) are held constant. Similarly, \( \Delta g(x) \geq 0 \) means that system I is stable, and \( \Delta g(x) < 0 \) means that system I is unstable. \( \Delta g(x) \) is given a new name: tangent plane distance (TPD).

\[ TPD(x) = \sum_{m=1}^{c} x_m \left[ \mu_m(x) - \mu_m(z) \right]. \]

The criteria of the stability of system I can be stated in terms of TPD: \( TPD(x) \) should be positive over the whole range of \( x \). However, the whole range of \( x \) is huge, thus different methods have been proposed to search for the values of \( x \) to test the stability of a system with a fixed composition \( z \). Michelsen [41][42] suggested locating the stationary points (maxima, minima, or saddle points) of \( TPD(x) \) rather than conducting an
exhaustive search in $x$-space for values of $x$ where $TPD(x) \geq 0$. The stationary points of $TPD(x)$ for a $c$-component system occur at points where

$$\frac{\partial}{\partial x_m} TPD(x) = 0, \quad m = 1, \ldots, c - 1.$$ 

Substituting the expression for $TPD(x)$ and using the Gibbs-Duhem expression

$$\sum_{m=1}^{c} x_m \frac{\partial \mu_m}{\partial x_j} = 0,$$

one can obtain

$$\mu_m(x) - \mu_m(z) = \mu_c(x) - \mu_c(z) = K, \quad m = 1, \ldots, c - 1.$$

$TPD(x)$ at the stationary points are obtained as

$$TPD(x) = \sum_{m=1}^{c} x_m K = K.$$

with $K \geq 0$ for a stable system and $K < 0$ for an unstable system. If the system is in two phases, the $K$ is deemed as the initial $K$ at the beginning of the successive substitution technique.
4. Application of Sparse Grids

4.1 Sparse Grid Interpolation

Firstly, a short introduction to sparse grids is given. Various introductions exist, thus only the major concepts of sparse grids are given, and details like boundary treatment, anisotropic grids and the detailed concept of spatially adaptive refinement are not explained here. Only the standard interval \([0,1]^d\) and functions \(f\) which are zero on the domain boundaries are considered. References [118] and [119] may be consulted for an extensive introduction.

Usually the creation of an approximation of a function \(f(x)\) with \(x \in [0,1]^d\) is done by taking a weighted sum of a suitable set of \(d\)-dimensional basis functions \(\phi_k\)

\[
f(x) \approx \sum_k \alpha_k \phi_k(x),
\]

with \(\alpha_k\) being a scaling coefficient for the basis function \(\phi_k\). The basis function has to be computed from the set of grid points \(y_i \in [0,1]^d\) on which the function is evaluated. The unit interval in the domain is used here for simplicity. The index \(k \in \mathbb{N}^d\) is a multi-index to uniquely label all basis functions. The set of basis functions determines the approximation quality, and possible choices include Lagrange polynomials, radial basis functions, splines and others. For multi-linear interpolation on a Cartesian grid \(V_n\) with \(n \in \mathbb{N}_+\) giving the number of grid points in each of the \(d\) dimensions, the basis functions \(\phi_k(x)\) are tensor products of one-dimensional hat functions.
\[
\phi_k(x) = \prod_{j=1}^{d} \phi_{kj}(x_j) \text{ with } \phi_{kj}(x_j) = \max \left( 1 - \frac{|x_j - (y_k)_j|}{h}, 0 \right).
\]

The functions are centered on the grid points \( y_k \). The support \( h = 1/(n + 1) \) is equal to the distance between the grid points and it is constant for each dimension in the regular Cartesian grids. For details on the boundary treatment we refer to [118]. Because the basis functions \( \phi_k \) are zero at all \( y_i \) with \( i \neq k \), \( \alpha_k = f(y_k) \) can be easily obtained. The regular Cartesian grids used for this approach have \( n^d \) grid points. For piece-wise linear basis functions the approximation error is only reduced with \( O(n^{-2}) \), thus a huge amount of grid points is required to achieve only minor increases in accuracy. This phenomenon is one form of curse of dimensionality.

Figure 9 The hierarchical basis in one dimension is depicted on the left. The basis functions of all levels together construct the approximation of a function. In the middle column the 2D hierarchical basis is shown. Taking only the subset with the most important contributions (gray selection) constructs a sparse grid in the top right corner, which has significantly less grid points than the \( 7^2 \) full grid in the bottom right corner. [118]
For sparse grids, hat functions are used as bases as well, but with varying and widely overlapping supports. The basis functions

\[ \psi_{l,k}(x) = \prod_{i=1}^{d} \psi_{l_i,k_i}(x_i) \text{ with } \psi_{l_i,k_i}(x_i) = \max\left(1 - \frac{|x_i - (y_{l_i,k_i})|}{2^{-l_i}}, 0\right) \]

are now indexed in a different way: the multi-index \( l \in \mathbb{N}_+^d \) corresponds to the resolution level with each level having \( 2^{l_i-1} \) grid points in the \( i \)th dimension. Multi-index \( k \in \mathbb{N}^d \) with \( k_i \in 0, ..., 2^{l_i-1} - 1 \) is the index for a grid point in that level. With that basis, a function can be approximated by

\[ f(x) \approx \sum_{l \in L} \sum_{k \in K_l} \alpha_{l,k} \psi_{l,k}(x), \]

where \( L = \{l \in \mathbb{N}_+^d : \forall i \ l_i \leq l_n \} \) and \( K_l = \{k \in \mathbb{N}^d : \forall i \ k_i < 2^{l_i-1} \} \). The grid points \( y_{l,k} \) are then located at \( y_{l_i,k_i} = k_i2^{1-l_i} + 2^{-l_i} \) for the \( i \)th dimension in the interval \([0,1] \). Contrary to the regular grid described above, the basis function \( \psi_{l,k} \) does not necessarily vanish at other grid points \( y_{l,i \neq k} \) as can be observed in Figure 9. The construction of the points ensures that

\[ \psi_{l,k} = 0 \text{ at all } y_{l,j} \text{ with } i : \exists i_z : i_z < l_z, z \in \{1, ..., d\}, \quad j \in K_i \]

which allows a computation of the hierarchical surpluses \( \alpha_{l,k} \) from all grid points by hierarchization. Thereby, the surplus in each level is built on top of the approximation of all preceding levels. Taking all grid points \( y_{l,k} \) up to a certain level \( l_i \leq l_{\text{max}} \) for \( i = 1, ..., d \), they resemble a regular full Cartesian grid with \( m = 2^{l_{\text{max}}} - 1 \) in each direction. This hierarchical basis exhibits the same approximation quality and requires also \( n^d \) grid points. For sufficiently smooth functions, i.e., those having bounded mixed
second derivatives, it has been shown that the hierarchical surpluses decay rather fast with a decreased size of their support [118]. That behavior is contrary to multi-linear interpolation on a regular Cartesian grid, where the coefficient $\alpha_k$ is solely determined by the local function value $f(y_k)$.

The hierarchical basis can then be used to construct a sparse grid. By neglecting all basis functions $\psi_{l,k}$ with $|l|_1 > l_{\text{max}} + d - 1$, some information of the original function cannot be represented anymore. However, if the smoothness criteria are fulfilled, the neglected hierarchical surpluses are small anyway [118]. The neglected basis functions do hold the majority of the grid points, thus leaving them out drastically reduces the number of grid points required for the approximation. The approximation quality then only slightly deteriorates. The error reduces to $O\left(m^{-2}(\log(m))^{d-1}\right)$ whereas the number of grid points is $O(m(\log m)^{d-1})$ which is much less than the number of points of a regular Cartesian full grid, especially in higher dimensions. The curse of dimensionality cannot be fully circumvented, but the influence of the dimension is moved into a logarithmic term. However, scenarios exist where sparse grids can fully overcome the curse of dimensionality [120].

Because functions can have a rather heterogeneous distribution of smoothness, a chosen maximum resolution level $l_{\text{max}}$ can be too fine for a certain part of the domain and too coarse for another part of the domain. Fortunately, a sparse grid using the standard piecewise linear basis can be refined locally by only adding few basis functions which again have local supports. For example, refining the basis function $\psi_{l,k}$ will
introduce a new basis functions $\psi_{l', k'_1}$ and $\psi_{l', k'_2}$ with $l' = l + e_r$, where $e_r$ is a vector having zeros everywhere but one in the $r$th position. The indices of the new basis function $k'_1 = k + e_r k$ and $k'_2 = k + e_r (k + 1)$ then determine the positions of the new grid points. Note that during the refinement all the basis functions of lower levels which contain the support of $\psi_{l', k'}$ have to be created if they do not exist yet. Different strategies for choosing the points of refinement exist, and the choice of the direction of the refinement $r$ is up to the user. A standard scheme of refinement, which is also used in this paper, is to find the largest $\alpha_{l, k}$ and refine it in all directions, i.e., create the $\psi_{l', k'}$ with $r = 1, \ldots, d$ consecutively. By the local refinement, sparse grids can thus create a good approximation of the functions despite being based on the linear basis functions. Because they have only local supports, it is not necessary to recompute the hierarchical surpluses during the refinement. For creating a subsequent surrogate, only the indices characterizing the grid points $(k, l)$ and the values $f(y_{k, l})$ or the hierarchical surpluses of the function have to be stored. Note that the requirements to store $k$ and $l$ are small compared to the double-precision function values [121]. The stored values are thus the stored grid points.

Sparse grids can also be created using the other basis functions than the piecewise linear hat function used here. Other choices can be trigonometric functions [122][123], higher-order polynomials [119] and Lagrange polynomials [52]. These usually have better approximation qualities and a different distribution of grid points. But the refinement usually requires a lot more effort because the wider support of the functions
requires a recomputation of nearly all surpluses. Moreover, sparse grids using the piecewise linear hat functions can cope better with discontinuities because they do not introduce oscillations which might appear when higher order polynomials are used for the approximation. The functions to be approximated in this work exhibit discontinuities, and some of them even have a bounded codomain of \([0,1]\), which could be violated by the oscillations.

For all sparse grid operations done in this work, we used the excellent sparse grid software SG++ written by Dirk Pflüger [124], which not only provides the functionality but also an easy access by a Python interface.

4.2 Sparse Grid Surrogate Model

As explained before, the flash computation is split into an online and an offline phase. During the offline phase the surrogate model is created, which can be evaluated cheaply during the online phase, i.e., the simulation of the compositional flow. Because the model is created only once, it can be reused repeatedly. The computational effort of the offline phase can be neglected. The original flash calculation \(F\) for two components computes 14 values, which are called observables, from two coordinates, which are the pressure \(p\) and the molar fraction of the first component. The 14 observables and their meanings can be observed in Table 4. For a surrogate \(S\), 12 of these functions are represented on a 2D domain by taking the same input values. The observables liquid and gas in Table 4 do not need to be computed by the surrogate model.
Table 4 The meaning of the observables in the two-component flash calculation

<table>
<thead>
<tr>
<th>Observable</th>
<th>Physical quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{W1}$</td>
<td>Molar fraction of Component 1 in the oil phase</td>
</tr>
<tr>
<td>$x_{W2}$</td>
<td>Molar fraction of Component 2 in the oil phase</td>
</tr>
<tr>
<td>$x_{N1}$</td>
<td>Molar fraction of Component 1 in the gas phase</td>
</tr>
<tr>
<td>$x_{N2}$</td>
<td>Molar fraction of Component 2 in the gas phase</td>
</tr>
<tr>
<td>$x_{W}$</td>
<td>Molar density of oil phase (unit: mole/m$^2$)</td>
</tr>
<tr>
<td>$x_{W}$</td>
<td>Mass density of oil phase (unit: kg/m$^2$)</td>
</tr>
<tr>
<td>$s_{N}$</td>
<td>Mass density of gas phase (unit: kg/m$^2$)</td>
</tr>
<tr>
<td>$s_{W}$</td>
<td>Saturation</td>
</tr>
<tr>
<td>$v_{1}$</td>
<td>Partial molar volume of Component 1</td>
</tr>
<tr>
<td>$v_{2}$</td>
<td>Partial molar volume of Component 2</td>
</tr>
<tr>
<td>$C_{f}$</td>
<td>Isothermal compressibility of the flow</td>
</tr>
<tr>
<td>$liquid$</td>
<td>Boolean variables, if liquid is one in some point, it</td>
</tr>
<tr>
<td></td>
<td>means that there exists an oil phase in this point;</td>
</tr>
<tr>
<td></td>
<td>if liquid is zero, it means that there does not exist</td>
</tr>
<tr>
<td></td>
<td>an oil phase in the point.</td>
</tr>
<tr>
<td>$gas$</td>
<td>Boolean variables, if gas is one in some point, it</td>
</tr>
<tr>
<td></td>
<td>means that there exists a gas phase in this point;</td>
</tr>
<tr>
<td></td>
<td>if gas is zero, it means that there does not exist a</td>
</tr>
<tr>
<td></td>
<td>gas phase in the point.</td>
</tr>
</tbody>
</table>

A straightforward approach for creating a surrogate is storing a multi-dimensional table containing all evaluations of the flash calculation on a regular full Cartesian grid. This table is then interpolated. While this surrogate $S_f$ is a valid approach for problems requiring moderate resolutions and only having few components, it becomes unreasonable for other problems. The first reason is that the creation of the table requires many computational resources because each value in the table has to be computed by a flash calculation. When it is in the offline phase, this can still be considered feasible. However, the data needs to be stored and then accessed in the online phase, which can be a severe problem because such a table with a huge number of entries might not fit into the memory anymore. The second reason is that in higher dimensionalities the number of variables computed by flash calculations is increased, which leads to $O(m_0(c) \cdot n^c)$ data points to be stored with $n$ being the number of entries in a single dimension, $c$.
being the number of components and \( m_o(c) \) being the number of observables depending on \( c \).

Using sparse grids, these requirements in memory can be reduced. Twelve adaptive sparse grid approximations \( S_{s,i} \) with \( i = 1, \ldots, 12 \) were constructed. During the offline phase, each \( S_{s,i} \) starts with a basic low-resolution sparse grid which is then refined in the areas with the highest surplus. Because each flash calculation retrieves all 12 output values, the construction of an \( S_{s,i} \) can reuse values if that particular grid point, i.e. a parameter combination \((p, z_1)\), has already been evaluated by a flash calculation for building another \( S_{s,j} \). This reduces the overall effort in creating the surrogate. Because the flash calculation always computes all the observables, one can also use the union of all the grid points in each surrogate as the union sparse grid for each of the surrogates \( S_{s,i} \). This drastically decreases the amount of data to be stored, but only leads to a minor increase in approximation quality. Because each of the dedicated grids is already refined to achieve a maximum approximation quality with the fewest amount of grid points, the additional gain of accuracy of the union grid would be small. Thus, in the test of next section, the union sparse grid is not used. Sparse grids also have a slightly larger overhead in evaluating \( S_s \) than \( S_f \) due to the overlap of the basis functions. However, this overhead is negligible in the current implementation.
4.3 Test Environment

For testing the applicability of a sparse grid surrogate $S_s$ for the simulation of compositional flow, its performance is evaluated with two references: the simulation using the standard flash calculation $F$ and the full grid surrogate $S_f$. In this manner, the errors can be identified as the ones introduced by the surrogate itself and the ones introduced by the sparse grid representation.

Table 5 Parameters of the four cases.

<table>
<thead>
<tr>
<th>Case</th>
<th>Phase</th>
<th>Temperature (K)</th>
<th>Initial substances in the reservoir (mole)</th>
<th>Initial pressure (Pa)</th>
<th>Injection point</th>
<th>Production point</th>
<th>Gravity in the y-direction (m/s²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Oil</td>
<td>100</td>
<td>100% propane</td>
<td>2012018</td>
<td>Left bottom corner</td>
<td>Right bottom corner</td>
<td>9.807</td>
</tr>
<tr>
<td>2</td>
<td>Gas</td>
<td>480</td>
<td>100% propane</td>
<td>2012018</td>
<td>Left bottom corner</td>
<td>Right bottom corner</td>
<td>9.807</td>
</tr>
<tr>
<td>3</td>
<td>Oil and gas</td>
<td>220</td>
<td>50% methane and 50% propane</td>
<td>2012018</td>
<td>Left bottom corner</td>
<td>Right bottom corner</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Oil and gas</td>
<td>220</td>
<td>50% methane and 50% propane</td>
<td>2012018</td>
<td>Left bottom corner</td>
<td>Right up corner</td>
<td>0</td>
</tr>
</tbody>
</table>

Because a plain comparison of the approximation qualities of $S_f$ and $S_s$ does not give much insight, a small scenario for the two-phase compositional flow is set up, which shall show that the surrogate model is applicable in general. The 2D simulation domain consists of $40^2$ grid cells and represents a homogeneous and isotropic subsurface reservoir with a dimension of four by four meters. Methane is injected into the domain with 0.02 mole/m²s to displace the propane in the reservoir. The simulation time span is 0.2 year. There are 1752 time steps. The temperature is held constant throughout the
simulation and fixed to three different values: 100, 220 and 480 K. These temperatures are chosen to ensure that the components are both in oil phase (100 K), both in gas phase (480 K) and both in two phases (220 K). For the 220 K case two different scenarios are set up, which differ in the production point (top or bottom right corner of the domain). The production point is a design variable for the compositional flow simulation, leading to different flows and recovery rates. However, the two 220 K scenarios use the same surrogate model because they do not differ in the parameters of the flash calculation.

Neumann boundary conditions are imposed on the boundaries of the domain except the production point where a Dirichlet boundary condition is imposed and a constant pressure equaling to the initial pressure is given. An overview of the scenarios can be observed in Table 5. The permeability of the porous medium is $9.869 \times 10^{-15}$ m$^2$, and the porosity of the porous medium is 0.2.

For this two-component simulation, a 2D surrogate is required, which takes the local pressure $p$ and the molar fraction of methane $z_1$ as the input values. In the test, the full grid surrogate $S_f(p, z_1)$ uses a 2D grid with the size of $129^2 = 16641$ points, which can be easily handled for this test purpose, even when the 12 values per grid-point are stored. Its domain ranges from 1.984 MPa to 2.112 MPa in $p$ and from 0 to 1 in $z_1$. The sparse grid surrogate $S_s(p, z_1)$ is built using an adaptive sparse grid for each flash output value. The sparse grid is only refined up to a level $l_{\text{max}} = 7$ which corresponds to a resolution of $2^{l_{\text{max}}} + 1 = 129$, and thus representing the same resolution as $S_f$. The surrogates can
thus be directly compared by evaluating $S_s$ on all entries of $S_f$ and comparing the value of the results.

Comparing the three simulations of the compositional flow (using $F$, $S_f$ and $S_s$) in the four scenarios is done in various ways. The trajectory of $p(t)$ and $z_1(t)$ can be compared. For that, three representative points are chosen: the injection point, the middle-point of the domain and the production point. At all points the trajectories are recorded to be analyzed. Choosing this set of representative points allows us to distinguish the local effects caused by a surrogate. Another important outcome of the compositional flow simulation is the recovery rate $R$ of propane, which should be preserved when using a surrogate model. Again, the trajectory $R(t)$ can be used for analyzing the error. In addition to that, the velocities, pressures and the molar fractions in the whole simulation domain can also be compared at certain time steps.

4.4 Surrogate Performance

Using adaptive sparse grids for creating $S_s$ leads to the grid configurations shown in Figure 10 where the union of the grids of all observables is depicted. The number of values stored for each observable is strongly varying as can be observed on the left of Figure 11. Depending on the temperature of the original flash calculation, rather different sparse grids are required. Whereas the single-phase cases with 100 K and 480 K only require rather small grids with 1024 and 354 evaluations respectively, the two-phase case at 220 K requires 4901 evaluations. That is mainly due to the appearance of
discontinuities in the two-phase case. The adaptive sparse grids are adapting to these, as can be observed in the middle of Figure 10. Thus sparse grids can reduce the number of flash calculations needed to create a surrogate compared to a full grid surrogate. Even for the unfavorable 220 K case, the 4901 evaluations are much less than the $129^2 = 16641$ evaluations required for a full grid. Also the amount of values to be stored for all observables is much smaller. Again for the unfavorable two-phase case, overall 20247 values have to be stored, which is $\approx 10\%$ of the $12 \cdot 129^2 = 199692$ values that would have to be stored using a full grid surrogate.

![Figure 10](image)

Figure 10 The union of the sparse grids used for the sparse grid surrogates for the temperatures 100 K, 220 K and 480 K, respectively. For each observable only a subset of the shown points is stored to build a surrogate.

The accuracy of each of the surrogates can also be estimated by comparing it to a higher resolution full grid surrogate. For that a double resolution, $257^2$ full grid surrogate has been created, and the average relative $L_1$ error of $S_f$ and $S_s$ with respect to this reference result is computed. The error of the full grid surrogate compared to the higher resolution surrogate can be observed on the right of Figure 11. It can be observed
that the 220 K case still has the highest errors for some observables, despite higher number of points already spent. Especially, the approximation of observables \( v_1, v_2 \) and \( C_f \) are still not very accurate. However, these are the ones with the most discontinuities, which can only be resolved poorly. Nevertheless it can also be observed that the corresponding sparse grid surrogates do not introduce much more error. Compared to the error introduced by halving the resolution, the error introduced by switching to a sparse grid representation is only minimal. On the average, the relative \( L_1 \) error is increased by a factor of around 1.4, which is rather small compared to the deviations in the approximation quality throughout the observables.

![Image](image_url)

**Figure 11** On the left an overview of the number of values stored for each observable and for each of the temperatures can be observed. For comparison the size of the union of all grids is given. On the right the average relative \( L_1 \) errors of the full grid surrogates \( S_f \) are given. Note that the small horizontal indicator above each bar shows the average relative \( L_1 \) error of the respective sparse grid surrogate \( S_s \).

The surrogates \( S_s \) and \( S_f \) have to be implemented carefully in the existing code. Using the plain values returned by the bilinear interpolation or the sparse grid evaluation
leads to a divergence of the compositional flow simulation. Due to the linear approximation of the discontinuities, some outcomes of the surrogates do not fulfill the conservation laws. Moreover, phase changes cannot be represented accurately using piecewise linear approximations because at a phase change the standard implementation of the compositional flow requires some observables being evaluated as zero. With the piecewise linear and continuous surrogates used here, this cannot be achieved. The reason is that they evaluate to something between the closest grid-points, which is in the worst case not zero where it should be zero. The simulation has been adjusted to directly detect the phase change, and the surrogate falls back to a piecewise constant approximation at these discontinuities by directly setting these values to zero. This might shift the discontinuity slightly compared to the results of a real flash calculation, but this shift is only of $O(h)$. Note that this post-processing has to be done for both surrogates and is currently dominating the evaluation time of both.

![Graph](image)

Figure 12 Occurrences of the absolute errors of the $p, z_1$ trajectories created by $S_f$ for all time steps at the three respective locations in the four simulation cases. One can clearly see that for most of the 21024 recorded time steps the trajectories of the simulations using $S_f$ is quite close to the ones of the simulations using the original flash calculations.
Figure 13 The deviation of the $p, z_1$ trajectory created by $S_s$ from the one created by $S_f$ for all time steps at the three respective locations in the four simulation cases. The sparse grid surrogate retrieves results that are close to the full grid surrogate, with the deviation being much smaller than the error introduced by the full grid surrogate.

Firstly, the trajectories $p_f(t)$ from $S_f$ are compared with the trajectories $p(t)$ from the original simulation. The trajectory $p_f(t)$ is rather close to the original one with a maximum error of 22 Pa which appears only once at the injection point in the pure gas scenario. It is rather close to the trajectory of the original simulation in most conditions, which can be observed on the left of Figure 12. The trajectory $p_s(t)$ is actually nearly identical with $p_f(t)$ and only deviates minimally on the order of $10^{-3}$ Pa as can be observed on the left of Figure 13. The overall range of $p$ values used is also smaller than the domain of the surrogates, indicating that a much smaller domain can be chosen for this particular simulation. Because the surrogates should span the whole domain of possible values, it is necessary to choose this large domain. In general the trajectories using the surrogates are slightly delayed, meaning that a certain $p(t_0)$ is reached by the simulation using the surrogate after a short time $\delta t$, i.e. $p(t_0) = p_f(t + \delta t)$. This
behavior can also be observed for the trajectories of $z_1$. The delay is of only few time steps for both.

Similar to the trajectories of $p(t)$, the trajectories of $z_1(t)$ from both surrogates have similar accuracies in all probing points, and the maximum absolute error is on the order of the grid spacing in $z_1$ of around $10^{-2}$. In $z_1$ the whole range of values has been exhausted, thus nothing can be saved in terms of domain size. Nevertheless, most of the time the absolute error of $z_1$ is smaller than 0.02, which is acceptable for the current application. Similar to the pressure trajectories, the trajectories of $z_1(t)$ using the surrogates are lying quite close to each other. The error of introducing the full grid surrogate is thus much higher than switching from $S_f$ to $S_s$. The sparse grid surrogate is thus a valid approximation of the full grid surrogate.

![Figure 14](image)

Figure 14 The relative error of propane recovery rate of full grid surrogate is shown on the left. The kink of case 1 is the result of a change in the sign of the absolute error. On the right the relative difference of recovery rates between $S_f$ and $S_s$ is shown. The difference between $S_f$ and $S_s$ is small compared to the error introduced by $S_f$. 
Secondly, the recovery rates of propane in the four scenarios are reproduced by both surrogates with nearly the same accuracies. The relative errors of the recovery rates in the more interesting cases 3 and 4 are around $O(10^{-2})$, whereas in the easier cases 1 and 2 they are around $O(10^{-3})$. Both are shown in Figure 14. Although the recovery rates of case 3 and 4 are similar, they correspond to flows with different characteristics. Longer simulations of the scenario would exhibit this distinction, but are not the focus of this work. The overall recovery rate retrieved by the simulation is only slightly inaccurate for both surrogates. Thus, for recovery rate of propane, both surrogates lead to the satisfactory results. The error from $S_f$ to $S_s$ is much smaller than the error of $S_f$. For larger scenarios, it is necessary to balance both errors to optimize the number of values stored for sparse grids.

4.5 Extensibility to Three Dimensions

Extending the current approach to more than two components would be one of the next steps. The potential for reducing the memory footprint is much higher, and sparse grids use even less points there. Unfortunately for more than two components, many more discontinuities appear, which in turn requires a lot more points to resolve them. The discontinuities themselves are $c-1$ dimensional. Even adaptive sparse grids would require $O(n^{c-1})$ values to be stored with a maximum resolution $n$ to fully resolve the discontinuities. This is unfeasible. Thus, adaptive sparse grids can ease the curse of
dimensionality by building surrogate models for flash calculations, but advanced 
approaches are required to resolve the discontinuities with minimized effort [125].

Another approach is to introduce several sparse grid surrogates for each of the regions 
bordered by the discontinuities. This might require fewer points than a single sparse grid. 
It requires only a highly accurate function that resolves the positions of the 
discontinuities. Because this function is expensive to evaluate, it is necessary to create a 
high-resolution sparse grid surrogate for it and use sparse grid approximants with few 
grid points for the rest of the observables. However, the approach might be not extensible 
to more components because the number of such faces seems to increase with the number 
of components, which in turn increases the number of sparse grids again.

4.6 Performance Improvement of the Two-phase Compositional Flow Simulation

Figure 15 The left plot shows the run time of different parts of the code on two different grids 
with and without the surrogate. The right plot shows the percentages of run time of different parts 
of the code. The number of processors is 64.
There has been a parallel implementation of the two-phase compositional flow model. In this section, sparse grid techniques were applied to the model, and the performance of the model with a sparse grid surrogate was evaluated. The case in Section 2.4 was tested again. The average run time of the flash calculation and the two-phase compositional flow model before and after the application of a surrogate is shown in Table 6 and Table 7, respectively. The acceleration of the code is also given in the two tables, which is defined as the average run time before application of a surrogate over the one after application of a surrogate. The solver time should be similar with or without a surrogate and it is displayed in Table 3. In Table 6 it can be observed that the online flash calculations with the surrogate are more than 1600 times faster than those without the surrogate for all the cases. From Table 7, it is found that the speed of the overall two-phase compositional flow simulation is increased by more than 10 times for the best case. The performance improvement by applying a surrogate is thus tremendous. The left plot of Figure 15 shows the run time of different parts of the code and the right plot shows the percentages of run time of different parts of the code over the total run time when the number of processors is 64. In the simulations without the surrogate, it can be observed that the flash calculation is really a bottleneck of the two-phase compositional flow simulation and takes more than 87% of total run time. It is followed by the solver time, which takes more than 8% of total run time. After the application of a surrogate, the bottleneck of the flash calculation has been removed. The run time of the flash
calculation takes less than 1% of total run time, and the solver part becomes the new bottleneck.

There is no distinction in the surrogate models employed because the online-phase execution time of surrogates \( S_f \) and \( S_s \) is nearly the same in the current implementation, which is dominated by the post-processing for dealing with the discontinuities. The difference between the two surrogates does not influence the overall execution time because anyway it only makes up about 1% of the overall simulation time. The resolutions and the dimensionality of the surrogates are currently too small to see a measurable difference in the execution time between the bilinear interpolation of a regular Cartesian grid in \( S_f \) and the evaluation of the adaptive sparse grid in \( S_s \). Note that these differences will gain significance in case of higher resolution surrogates and higher dimensional problems. In this condition, the evaluation of a sparse grid can take longer time due to a large number of hierarchical subspaces that have to be evaluated respectively. Nevertheless, for the sparse grid surrogate \( S_s \) much less values are computed and stored during the offline phase compared to the full grid surrogate \( S_f \). The time to create the surrogates in the offline phase is mainly determined by the number of evaluations of the original flash calculation. For the 100 K case it requires 6.58 s, for the 480 K case it requires 2.02 s, and for the 220 K case used for the large experiments here it takes 30.9 s, which corresponds to 4901 evaluations of the original flash calculation. This is still much less than the \( 8.07 \times 10^8 \) and \( 1.44 \times 10^9 \) flash calculations required in the original \( 480^2 \) and \( 640^2 \) simulation, respectively. So even if the time to construct the
surrogate is added to the time of the online phase, the overall computing time is still
tremendously reduced.

Table 6 Average run time and acceleration of flash calculations with and without the use of a
surrogate S. N is the grid size, P is the number of processors, and the unit is second.

<table>
<thead>
<tr>
<th>N</th>
<th>480°</th>
<th>640°</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Without S</td>
<td>With S</td>
</tr>
<tr>
<td>64</td>
<td>8607.3</td>
<td>1684.4</td>
</tr>
<tr>
<td>128</td>
<td>4360.0</td>
<td>1633.0</td>
</tr>
<tr>
<td>256</td>
<td>2284.1</td>
<td>1655.1</td>
</tr>
<tr>
<td>512</td>
<td>1208.4</td>
<td>1633.0</td>
</tr>
</tbody>
</table>

Table 7 Average run time and acceleration of the parallel code with the use of a surrogate S. N is
the grid size, P is the number of processors, and the unit is second. The average run time of the
parallel code without S is referred to Table 1.

<table>
<thead>
<tr>
<th>N</th>
<th>480°</th>
<th>640°</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>With S</td>
<td>Acceleration</td>
</tr>
<tr>
<td>64</td>
<td>1142.2</td>
<td>8.60</td>
</tr>
<tr>
<td>128</td>
<td>937.2</td>
<td>5.77</td>
</tr>
<tr>
<td>256</td>
<td>852.8</td>
<td>3.69</td>
</tr>
<tr>
<td>512</td>
<td>830.0</td>
<td>2.46</td>
</tr>
</tbody>
</table>

4.7 Numerical Results

Case 3 in Table 5 was simulated for three years on the grid of $480^2$ cells. The
correctness of flash calculations can be validated by the online software VLEFlash [126]
and Flash Distillation Program Online [127]. The change in the molar fraction of propane
in the domain during the three years is shown in Figure 16. Because propane has more
economic value than methane, the recovery rate of propane during the three years is an
important index for oil companies, as is shown in Figure 17(a). The streamlines of the
compositional flow at the end of the third year can be observed in Figure 17(b). From the
results, it can be observed that propane is displaced gradually by methane from the
injection point (the left down corner) and produced from the production point (the right down corner) smoothly. At the end of the third year, about 22% of propane is pushed out from the domain. However, the propane displacement speed is not uniform in the domain. More propane is displaced in the $x$-direction than in the $y$-direction, which can be explained by the detour of the streamlines in the $y$-direction.

Figure 16 Change in the molar fraction of propane in the domain during the three years. (a), (b), (c) and (d) stand for the condition at the end of 0.1 year, one year, two years and three years, respectively.
Figure 17 (a) Recovery rate of propane during the three years and (b) streamlines in the domain at the end of the third year.
5. 2D Parallel Simulation of Wormhole Propagation

5.1 Basic Equations

This chapter introduces the topic of wormhole simulations. Different from the two-phase compositional flow simulations discussed before, only single-phase flows (liquid flows) are considered in wormhole simulations. At the Darcy scale, the equation system that describes the wormhole propagation in porous media includes momentum conservation equations (41), mass conservation equations (42), solute transport equations (43) and a number of semi-empirical equations (44), (45) and (46) that bridge the scale differences, as described earlier. They can be listed as

\[
\frac{\rho}{\phi} \frac{\partial \mathbf{u}}{\partial t} + \frac{\rho}{\phi^2} \nabla \cdot \mathbf{u} \mathbf{u} = -\nabla p + \frac{\mu}{\phi} \nabla^2 \mathbf{u} - \frac{\mu}{K} \mathbf{u} - \frac{\rho F}{\sqrt{K}} |\mathbf{u}| \mathbf{u} + \rho \mathbf{g}, \tag{41}
\]

with \( F = \frac{1.75}{\sqrt{150\phi^3}} \) being the Forchheimer coefficient.

\[
\varepsilon \frac{\partial p}{\partial t} + \frac{\partial \phi}{\partial t} + \nabla \cdot \mathbf{u} = 0, \tag{42}
\]

\[
\frac{\partial (\phi C_f)}{\partial t} + \nabla \cdot (\mathbf{u} C_f) = \nabla \cdot (\phi \mathbf{D}_e \cdot \nabla C_f) - k_c a_v (C_f - C_s), \tag{43}
\]

\[
k_c (C_f - C_s) = R(C_s), \tag{44}
\]

\[
\frac{\partial \phi}{\partial t} = \frac{R(C_s)a_v \alpha}{\rho_s}, \tag{45}
\]

\[
C_s = \frac{C_f}{1 + \frac{k_s}{k_c}}. \tag{46}
\]

\( \mathbf{u} \) is the velocity vector, \( \rho \) is the fluid density, \( \rho_s \) is the solid density, \( \phi \) is the porosity, \( t \) is the time and \( \mu \) is the fluid viscosity. If the porous medium is isotropic and homogeneous, then \( K \) is the permeability value, \( \mathbf{g} \) is the gravity vector, \( p \) is the pressure, \( C_f \) is the concentration of acid in the fluid phase, \( C_s \) is the concentration of acid at the fluid-solid interface, \( R(C_s) \) is the reaction kinetics, \( k_c \) is the local mass-transfer
coefficient, $k_s$ is the surface reaction rate constant, $a_v$ is the interfacial area available for reaction per unit volume of the medium and $\alpha$ is the dissolving power of acid. $\varepsilon$ is a pseudo-compressibility parameter that is introduced to facilitate the system. As stated in [128], in some conditions, when $\varepsilon$ is 0, the system may be singular and cannot be solved by the solver. A suggested solution is to set $\varepsilon$ to be a very small positive number to ensure an invertible coefficient matrix. $D_e$ is the effective dispersion tensor and it is suggested to use the method on Page 136-137 in [129] to calculate it.

$$D_e(u) = d_m I + |u|(d_l E(u) + d_t E^\perp(u)),$$

where $d_m$ is the molecular diffusion coefficient, and $d_l$ and $d_t$ are the longitudinal and transverse dispersion coefficients, respectively. All the three coefficients are positive. $|u|$ is the Euclidean norm of the velocity, and $E(u)$ is the orthogonal projection along the velocity. In 2D condition,

$$E(u) = \frac{1}{|u|^2} \begin{pmatrix} u_x^2 & u_x u_y \\ u_y u_x & u_y^2 \end{pmatrix},$$

and $E^\perp(u) = I - E(u)$ with $I$ being the identity matrix. In some cases, the tensor dispersion is more significant than the molecular diffusion, and $d_l$ is usually considerably larger than $d_t$. (41) is based on the general Navier-Stokes equation [130], extended by the Darcy term, the Brinkman term and the Forchheimer term. In (41), the term $-\nabla p - \frac{\mu}{\kappa} u + \rho g$ is called the Darcy term. The term $\frac{\mu}{\phi} \nabla^2 u$ is called the Brinkman term, which is used to account for transitional flows between boundaries of porous media and clear fluids [131]. In 1901, an Austrian scientist Phillip Forchheimer investigated
fluid flows through porous media in high velocity regime [132]. He pointed out that as the flow velocity increases, the inertial effects dominate the flow. To account for the high velocity inertial effects, he added to Darcy’s equation an inertial term representing the kinetic energy of the fluid. This term is called the Forchheimer term, which is expressed as $\frac{\rho_F}{\sqrt{K}}|u|^3$ in (41). In (42), both the medium and fluid compressibility have been neglected, although they may have some effects on the wormhole generation and propagation in carbonate rocks [62]. Equation (43) computes the concentration of the acid species. The first three terms stand for the accumulation, advection [72] and dispersion of acid, respectively. The fourth term describes the transfer of the acid species from the fluid phase to the fluid-solid interface. (44) balances the amounts of reactant transferred to the surface and reacted. The mineral reactive surface area has been neglected. (45) describes the evolution of the porosity in the domain as a result of the reaction, and (46) shows the relationship between $C_f$ and $C_s$. It is noticed that temperature effects [70][71] have been omitted in (41) - (45), so that only isothermal acid dissolution systems are considered in this work.
In pore scale, the following three equations are given

\[
\frac{K}{K_0} = \frac{\phi}{\phi_0} \left( \frac{\phi(1-\phi_0)}{\phi_0(1-\phi)} \right)^2, \quad (48)
\]

\[
\frac{r_p}{r_0} = \sqrt{\frac{K\phi_0}{K_0\phi'}} \quad (49)
\]

\[
\frac{a_v}{a_0} = \frac{\phi r_0}{\phi_0 r_p} \quad (50)
\]

\( r_p \) represents the value of the average pore radius. The subscript 0 stands for the initial value. (48) is the Carman-Kozeny correlation which establishes the relationship between
the permeability and porosity. (49) and (50) show the relationship between the permeability and interfacial area.

From (41) to (50), \( \rho, \mu, g, k_c, \alpha, \rho_s, k_s, K_0, \phi_0, r_0 \) and \( a_0 \) are considered constant, while the other variables change in the loop of time iterations. An implicit-explicit hybrid scheme is used to solve (41) and (42) together to derive the pressures and velocities. The implicit-explicit hybrid scheme can be written as

\[
\frac{\rho \mathbf{u}^{\ell+1} - \mathbf{u}^\ell}{\Delta t} = -\frac{\rho}{\phi^2} \nabla \cdot \left( \mathbf{u}^{\ell+1} \mathbf{u}^\ell - \nabla P^{\ell+1} \right) + \frac{\mu}{\phi} \nabla^2 \mathbf{u}^{\ell+1} - \frac{\mu}{K} \mathbf{u}^{\ell+1} - \frac{\rho F}{\sqrt{K}} |\mathbf{u}^\ell| \mathbf{u}^{\ell+1} + \rho \mathbf{g},
\]

\[
\frac{\phi^{\ell+1} - \phi^\ell}{\Delta t} = -\nabla \cdot \left( \mathbf{u}^{\ell+1} \right) - \frac{\rho(C_s)\alpha v}{\rho_s} \phi^{\ell+1},
\]

where \( \ell \) stands for the time step. In addition to the pressures and velocities, the values of the other variables are from time step \( \ell \), and the superscripts are abbreviated here. With the new velocities and the effective dispersion tensor, \( C_f \) and \( C_s \) may be computed by (43) and (46) with a fully implicit method. Moreover, (44), (45) and (50) can be used to derive the new porosity \( \phi \) with an implicit-explicit hybrid scheme. Combining the three equations, a new equation is generated as

\[
\frac{\partial \phi}{\partial t} = \frac{a_0 \alpha C_f k_c k_s (1 - \phi)}{\rho_s (k_c + k_s) (1 - \phi_0)}.
\]

With the implicit-explicit hybrid scheme, there is

\[
\frac{\phi^{\ell+1} - \phi^\ell}{\Delta t} = \frac{a_0 \alpha C_f^{\ell+1} k_c k_s (1 - \phi^{\ell+1})}{\rho_s (k_c + k_s) (1 - \phi_0)},
\]

\[
\phi^{\ell+1} = \frac{A + \phi^\ell}{1 + A},
\]
with \( A = \frac{a_0 a C_{f+1} k_c k_s \Delta t}{\rho_s (k_c + k_s)(1 - \phi_0)} \). The new porosity can then be substituted into (48) to update the permeability \( K \). With (49) and (50), one can derive the new \( a_v \). Finally, with all the new parameter values, one can go back to (41) and (42) to continue the loop. The flow chart of the solution procedure is shown in Figure 18.

5.2 Discretization and Experimenting Field Approach

After the discretization by CCFD method, two linear systems are generated. One is to solve pressures and velocities, and the other one is to solve concentrations. The discretization of the first linear system will be discussed in the following section, and the discretization of the second linear system is similar. The experimenting field approach in this work [133]-[137] is adapted. While this approach has been implemented for pressure field in these works, it is generalized to include velocity field as well. In Figure 19, suppose there is a 2D domain divided into \( 4^2 \) cells, and the velocity field is represented as arrows on the edges and the pressure field is represented as black points in the center of cells.
In the following discussion, (41) is also called the momentum equation, and (42) is also called the continuity equation. The discretized equations of (41) and (42) form a linear system. For each vertical edge of the cells, a discretized $x$-momentum equation can be imposed on it. If a vertical edge lies in the $i$th column and the $j$th row, then (41) can be discretized as

$$
a_{i,j-1}^{(o)} u_{x,i,j-1} + a_{i-1,j}^{(o)} u_{x,i-1,j} + a_{i,j}^{(o)} u_{x,i,j} + a_{i+1,j}^{(o)} u_{x,i+1,j} + a_{i,j+1}^{(o)} u_{x,i,j+1} + \\
b_{i-1,j}^{(o)} p_{i-1,j} + b_{i,j}^{(o)} p_{i,j} = c^{(o)}, \quad i = 1, \ldots, 5, j = 1, \ldots, 4.
$$

$a$ is the coefficient of the $x$-direction velocity, $b$ is the coefficient of the pressure, and $c$ is a constant term on the right-hand side. The superscript $(o)$ of the coefficients represents the index of the equation in the linear system. In the same way, a discretized
y-momentum equation is imposed on the horizontal edge. Suppose a horizontal edge lies in the \( i \)th column and the \( j \)th row, \((41)\) can then be discretized as

\[
\begin{align*}
   a_i^{(o)} u_{y,i-1,j} + a_{i-1,j}^{(o)} u_{y,i,j-1} + a_{i,j}^{(o)} u_{y,i+1,j} + a_{i+1,j}^{(o)} u_{y,i,j+1} + a_i^{(o)} u_{y,i,j} + a_{i,j}^{(o)} u_{y,i,j} &= \\
   b_i^{(o)} p_{i-1,j} + b_{i,j}^{(o)} p_{i,j} &= c^{(o)}, \quad i = 1, \ldots, 4, \quad j = 1, \ldots, 5.
\end{align*}
\]

\((52)\)

\(a'\) is the coefficient of the \( y \)-direction velocity, \(b'\) is the coefficient of the pressure, and \(c'\) is a constant term on the right-hand side. Under the Neumann boundary condition, the \( x \)-momentum equations and \( y \)-momentum equations on the boundaries degenerate to

\[
   u = u_B,
\]

with \( u_B \) is the boundary velocity. The continuity equation \((42)\) is imposed in the center of the cell so that for the cell in the \( i \)th column and the \( j \)th row, \((42)\) can be discretized as

\[
   b_i^{(o)} p_{i,j} + u_{x,i+1,j} = u_{x,i,j} - u_{x,i,j} + u_{y,i,j+1} - u_{y,i,j} = c^{(o)}, \quad i = 1, \ldots, 4, \quad j = 1, \ldots, 4.
\]

\((53)\)

\(b''\) is the coefficient of the pressure, and \(c''\) is the constant term on the right-hand side.

It is noticed that \((45)\) is substituted into \((42)\) to cancel the unknown variable \( \phi \).
Combining (51), (52) and (53), a linear system $Ax = b$ is generated with

$$x = (u_{x,1,1}, u_{x,2,1}, \ldots, u_{x,5,4}, u_{y,1,1}, u_{y,2,1}, \ldots, u_{y,4,5}, p_{1,1}, p_{2,1}, \ldots, p_{4,4})^T.$$  

$b$ is the right-hand side vector and $A$ is the coefficient matrix. Figure 20 shows the indices of the unknown variables in the linear system in a more direct way. The numbers on the edges stand for the indices of the velocities, and the numbers in the cells stand for the indices of the pressures. Moreover, matrix $A$ can be divided into nine submatrices, and $Ax = b$ can then be rewritten as

$$
\begin{pmatrix}
A_{xx} & A_{xy} & A_{xp} \\
A_{yx} & A_{yy} & A_{yp} \\
A_{cx} & A_{cy} & A_{cp}
\end{pmatrix}
\begin{pmatrix}
u_x \\
u_y \\
p
\end{pmatrix}
= b.
$$

$A_{xx}$ consists of the coefficients of $u_x$ in the $x$-momentum equations, $A_{xp}$ contains the coefficients of $p$ in the $x$-momentum equations, and $A_{xy} = 0$ because there is no $u_y$ in the $x$-momentum equations. $A_{yy}$ consists of the coefficients of $u_y$ in the
y-momentum equations, $A_{yp}$ contains the coefficients of $p$ in the y-momentum equations, and $A_{yx} = 0$ because there is no $u_x$ in the y-momentum equations. $A_{cx}$ consists of the coefficients of $u_x$ in the continuity equations, $A_{cy}$ contains the coefficients of $u_y$ in the continuity equations, and $A_{cp}$ contains the coefficients of $p$ in the continuity equations.

Traditionally, an algebraic method is used to derive the equation coefficient expressions by hand, which is cumbersome and error-prone. Thus, a new method called the experimenting field approach is proposed to calculate the coefficients directly without knowing their concrete expressions. It is easy to see that the numbers in Figure 20 also represent the indices of the equations in $Ax = b$. The numbers on the vertical edges stand for the indices of the $x$-momentum equations, the numbers on the horizontal edges stand for the indices of the $y$-momentum equations, and the numbers in the cells stand for the indices of the continuity equations. For example, the 8th equation is imposed on the edge numbered 8. It can be expressed as

$$a_3^{(8)} u_{x3} + a_7^{(8)} u_{x7} + a_8^{(8)} u_{x8} + a_9^{(8)} u_{x9} + a_{13}^{(8)} u_{x13} + b_{46}^{(8)} p_{46} + b_{47}^{(8)} p_{47} = c^{(8)}.$$  

It is noticed that the subscripts are represented by the index numbers here, instead of the coordinates as before. If one sets $u_{x8} = 1$ and the other unknowns as 0, the left-hand side of the equation is equal to the coefficient of $u_{x8}$, which is $a_8^{(8)}$. In the same way, the other coefficients can be calculated without knowing their expressions. Furthermore, it is also noticed that there are five equations that contain the unknown variable $u_{x8}$. They are
the 3rd equation, the 7th equation, the 8th equation, the 9th equation and the 13th equation, as listed below

\[
a_2^{(3)} u_{x2} + a_3^{(3)} u_{x3} + a_4^{(3)} u_{x4} + a_8^{(3)} u_{x8} + b_{42}^{(3)} p_{42} + b_{43}^{(3)} p_{43} = c^{(3)},
\]
\[
a_2^{(7)} u_{x2} + a_6^{(7)} u_{x6} + a_7^{(7)} u_{x7} + a_8^{(7)} u_{x8} + a_{12}^{(7)} u_{x12} + b_{45}^{(7)} p_{45} + b_{46}^{(7)} p_{46} = c^{(7)},
\]
\[
a_3^{(8)} u_{x3} + a_7^{(8)} u_{x7} + a_8^{(8)} u_{x8} + a_9^{(8)} u_{x9} + a_{13}^{(8)} u_{x13} + b_{46}^{(8)} p_{46} + b_{47}^{(8)} p_{47} = c^{(8)},
\]
\[
a_4^{(9)} u_{x4} + a_8^{(9)} u_{x8} + a_9^{(9)} u_{x9} + a_{10}^{(9)} u_{x10} + a_{14}^{(9)} u_{x14} + b_{47}^{(9)} p_{47} + b_{48}^{(9)} p_{48} = c^{(9)},
\]
\[
a_8^{(13)} u_{x8} + a_{12}^{(13)} u_{x12} + a_{13}^{(13)} u_{x13} + a_{14}^{(13)} u_{x14} + a_{18}^{(13)} u_{x18} + b_{50}^{(13)} p_{50} + b_{51}^{(13)} p_{51} = c^{(13)}.
\]

If one sets \( u_{x8} = 1 \) and the other unknowns as 0, one can derive five coefficients at the same time. They are \( a_8^{(3)} \), \( a_8^{(7)} \), \( a_8^{(8)} \), \( a_8^{(9)} \) and \( a_8^{(13)} \). Thus, this method needs only \( N \) loops to derive all the coefficient values of the linear system with \( N \) equaling with the number of unknowns. It is emphasized that the method can calculate the coefficient values automatically, and there is no need to know the details of their algebraic expressions.
Figure 21 $x$-direction velocity field in nine shifts. The bold arrows stand for one values, and the other arrows stand for the zero values.

The number of loops can be reduced even further. All the unknowns are divided into three fields: the $x$-direction velocity field, the $y$-direction velocity field and the pressure field. The $x$-direction velocity field corresponding to the grid in Figure 20 is shown in Figure 21. In the $x$-momentum equations, if one sets $u_{x,1,1} = 1$, one can derive the coefficient values from the 1st, 2nd and 6th equations and cannot derive any coefficient values from the other equations because the other equations do not contain the variable $u_{x,1,1}$. With this philosophy, when one sets $u_{x,1,1} = 1$, one can set $u_{x,4,1} = 1$, $u_{x,1,4} = 1$
and \( u_{x,4,4} = 1 \) at the same time, as is shown as the bold arrows in the first plot of Figure 21. In fact, one can set \( u_x \) every other two as one in each direction at the same time. Then, one can shift the bold arrows by one place and set another four \( u_x \) to the value one, as is shown in the second plot of Figure 21. The figure shows the entire shift procedure. Thus, a total of only nine shifts are needed to derive all the coefficient values of \( u_x \) in the x-momentum equations, which is \( A_{xx} \). The loop number of nine is reduced from the loop number of 20 before the improvement. Before the acceleration, \( O(N) \) loops are needed to derive all the coefficients, but now, only nine shifts are needed, which is \( O(1) \). When \( N \) is very large, which is very common in large-scale problems, the acceleration is tremendous.

To derive the coefficient values of the pressures in the x-momentum equations, the pressure field is shifted. Because each pressure variable appears only in the two equations of its left and right side, one can set \( p \) every other one as one in the x-direction. A total of only two shifts are needed to derive all the \( p \) coefficient values in the x-momentum equations. All the values make up the nonzero entries of the matrix \( A_{xp} \). The shift procedure is shown in Figure 22. The same philosophy is held to derive the coefficient values of the y-momentum equations, which are also the nonzero entries of matrices \( A_{yy} \) and \( A_{yp} \). The field shift procedure of the continuity equations is similar, and the description is omitted here.
In (41), the porosity \( \phi \) and the permeability tensor \( k \) on the edge are calculated using the harmonic weighting method. (43) is imposed on the center of the cells, and the concentration of acid \( C_f \) on the edge is calculated by the single-point upwind method.

5.3 Parallelization

Figure 23 Domain decomposition strategy. The red, black, green and blue parts are allocated to Processors 0, 1, 2 and 3, respectively.
For the wormhole simulation, it is necessary to have a fine enough mesh to describe the natural distribution of the porosity in the porous medium. Because the grid size becomes large, it is reasonable to leverage parallelism to speed up the simulation procedure. To the best of our knowledge, no work before ours has ever tried to use parallelism to simulate the wormhole propagation before. Suppose there are four processors, and then the domain in Figure 19 can be decomposed as shown in Figure 23. It has been already stipulated in the former section that an edge of a cell can represent either a velocity on the edge or a momentum equation imposed on the edge. Similarly, the central point of a cell can represent either a pressure in the cell or a continuity equation imposed on the point. With this philosophy, the notations in Figure 23 are easy to understand. All the variables and equations represented by red, black, green and blue edges and points are allocated to Processor 0, 1, 2 and 3, respectively. In this manner, no two processors have overlapping variables and equations. Such a decomposition strategy enables the achievement of load balance for each processor. Moreover, it reduces the communication cost among the processors to the lowest level.

However, it is still necessary to allow some overlapping data in one processor. Take Processor 0 as an example. In Figure 24 (1), Numbers 1 to 12 mark all the variables and equations in Processor 0. Firstly, it is considered that what overlapping data is needed to generate the $x$-momentum equations in Processor 0. For example, to derive the coefficient of $u_{x,13}$ in the 2nd equation (the $x$-momentum equation imposed on the edge numbered 2), one has to set $u_{x,13} = 1$. However, $u_{x,13}$ is not in Processor 0 but rather in Processor
1. Thus, Processor 0 has to store the overlapping variable $u_{x,13}$ and the overlapping equation imposed on Edge 13. The other overlapping edges of such type in Processor 0 can be derived in the same way, which are shown as the yellow edges in Figure 24 (1). It is also noticed that the 13th equation needs the velocities $u_{y,6}$, $u_{y,8}$, $u_{y,19}$ and $u_{y,21}$ to calculate its advection term, so that Edges 19 and 21 are included in Processor 0. Such a type of overlapping edges in Processor 0 is represented by the grey edges in Figure 24 (1). With the same philosophy, all the overlapping edges needed by the $y$-momentum equations and continuity equations in Processor 0 are shown in Figure 24 (2) and (3), respectively. Finally, by combining all the overlapping variables and equations in Figure 24 (1), (2) and (3) together, all the variables and equations in Processor 0 can be observed in Figure 24 (4).

In addition to creating the overlapping edges, Processor 0 also needs to communicate with the neighboring processors to derive the necessary variable values. For example, in Figure 24 (1), all the $y$-direction velocities on the grey edges should be acquired from the neighboring processors. In Figure 24 (2), all the $x$-direction velocities on the grey edges should be transferred to Processor 0 from the neighboring processors. There is no need to transfer pressures among the processors. All the communications above constitute the communication cost of the parallel code. The overlapping edges and the communication cost of the other processors hold to the same philosophy as Processor 0, so the description is omitted here.
Figure 24 Overlapping edges needed by the $x$-momentum equations (1), $y$-momentum equations (2) and continuity equations (3) in Processor 0. All variables and equations in Processor 0 can be shown in (4).

Figure 25 Indices of unknowns in $x$ vector in the parallel code.
Figure 26 The linear system $Ax = b$ in Hypre

In the parallel code, the structure of the linear system $Ax = b$ is different from its counterpart in the serial code. The library Hypre is used to solve the system. In Hypre, the unknown variables in vector $x$ are arranged processor by processor, which means that the variables in Processor 0 are put into $x$ at first, and then it is the variables in Processor 1 that are put into $x$ among others. In each processor, the variables are still arranged in the order of the $x$-direction velocities, then the $y$-direction velocities and lastly the pressures. According to the domain decomposition strategy in Figure 23, the indices of the unknown variables in $x$ in the parallel code are shown in Figure 25. It can be observed that the indices in Figure 25 also represent the indices of equations in the linear system. Because the indices of the unknowns in $x$ and the indices of the equations
have changed, the locations of the nonzero entries of matrix $A$ also need to be changed. The nine submatrices of $A$ in the serial code no longer exist.

The linear system $Ax = b$ in Hypre can be observed in Figure 26. P0 to P3 represent Processor 0 to 3, respectively. From the figure, it can be observed that each processor owns one part of the data of the linear system, which is separated by three dashed lines. After the processors generate their own data, the solver in Hypre assembles them together and solves the linear system. It is emphasized that different from the two-phase compositional flow simulation where structured-grid system interface of Hypre is called, the wormhole simulation code called linear-algebraic system interface of Hypre to set up the linear system.

5.4 Verification

A parallel code with FORTRAN90 and MPI is developed to simulate the wormhole propagation procedure. The preconditioner DS [138] and the solver GMRES in Hypre are chosen to solve $Ax = b$. To verify the correctness of the code, the benchmark problem shear-driven cavity flow [139] is reproduced with it. In the problem, there is a laminar incompressible flow inside a square cavity whose top wall moves with a uniform velocity, as is shown in Figure 27. The governing equations (41) and (42) reduce to the following two equations, respectively.

\[
\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{u} \mathbf{u} = -\nabla p + \frac{1}{Re\nu^2} \nabla^2 \mathbf{u},
\]

\[
\nabla \cdot \mathbf{u} = 0.
\]
The length of the square cavity is 1 m, and the wall velocity is 1 m/s. The Reynolds number ($Re$) is 100. Gravity is omitted. The grid has $100^2$ cells. In such condition, because there is zero in the main diagonal of the coefficient matrix, iterative solvers cannot be applied. Thus, the sparse direct solver UMFPACK [141] is used. When the flow becomes stable, the simulation results on Shaheen are shown from Figure 28 to Figure 33. Figure 28 to Figure 30 can be compared with the results from the commercial software such as CFD from ANSYS [142]. Figure 31 shows the streamlines in the cavity, which can be compared with Figure 3 in [139]. Their configurations are very similar. Figure 32 shows the $y$-direction velocity component along the horizontal centerline, and Figure 33 shows the $x$-direction velocity component along the vertical centerline. Both of the present results are compared with Ghia’s data in [139]. It is found that the present results match Ghia’s data to eyeball resolution. All the above experimental results verify the correctness of the code.

![Figure 27 Shear-driven fluid flow in a square cavity. $u$ stands for the $x$-direction boundary velocity, and $v$ stands for the $y$-direction boundary velocity.](image)
Figure 28 Pressure field of the 2D domain.

Figure 29 $x$-direction velocity field of the 2D domain.
Figure 30 $y$-direction velocity field of the 2D domain.

Figure 31 Streamlines of the 2D domain.
Figure 32: $y$-direction velocity components along the horizontal centerline.

Figure 33: $x$-direction velocity components along the vertical centerline.
5.5 Sensitivity Analysis

Table 8 Properties of the acid flow and porous medium in the dissolution experiment

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viscosity of the acid flow ($\mu$)</td>
<td>$10^{-2}$ Pa $\cdot$ s</td>
</tr>
<tr>
<td>Density of the acid flow ($\rho$)</td>
<td>$10^3$ kg/m$^2$</td>
</tr>
<tr>
<td>Longitudinal dispersion coefficient</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>Transverse dispersion coefficient</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>Molecular diffusion coefficient</td>
<td>$10^{-9}$</td>
</tr>
<tr>
<td>Surface reaction rate constant ($k_s$)</td>
<td>10 m/s</td>
</tr>
<tr>
<td>Local mass-transfer coefficient ($k_c$)</td>
<td>1 m/s</td>
</tr>
<tr>
<td>Dissolving power of acid ($\alpha$)</td>
<td>0.1 kg/mole</td>
</tr>
<tr>
<td>Density of the rock ($\rho_s$)</td>
<td>2500 kg/m$^3$</td>
</tr>
<tr>
<td>Initial interfacial area ($a_0$)</td>
<td>0.5 m$^2$</td>
</tr>
<tr>
<td>Initial concentration of the acid flow ($C_f^0$)</td>
<td>10 mole/m$^2$</td>
</tr>
</tbody>
</table>

Figure 34 Changes in the average left boundary pressures in the three resolutions

A sensitivity analysis has to be carried out to enable that the simulation grid is sufficiently fine. A small porous medium of 0.2-meter by 0.2-meter is analyzed based on
the consideration of saving computing time. Three sizes of grid with $20^2$, $40^2$ and $60^2$ cells are tested. The initial porosity in the porous medium is 0.2 everywhere, and the initial permeability of the porous medium is $10^{-8}$ m$^2$ everywhere. An acid flow is injected from the left boundary of the porous medium at a velocity of 0.01 m/s. A constant pressure of $10^5$ Pa is imposed on the right boundary of the porous medium. The upper and lower boundaries of the porous medium are closed. The initial pressure inside the porous medium is also $10^5$ Pa. Two high-porosity entries are set on the left boundary of the porous medium, one 0.05 meter above the bottom with a porosity of 0.6, and the other 0.1 meter above the bottom with a porosity of 0.5. The simulation period is $10^4$ seconds, and each time step is 100 seconds. The simulation is based on the DBF framework. The other properties of the acid flow and porous medium are shown in Table 8. The parameter $\epsilon$ is set to zero. In the experiment, the condition number of the generated linear system is very large, and popular preconditioners and solvers cannot achieve convergence in a reasonable number of iterations. Thus, the sparse direct solver UMFPACK is used. The changes in the average left boundary pressures for the three resolutions are shown in Figure 34. The changes in the average permeabilities for the three resolutions are shown in Figure 35. The porosity fields in the porous medium at the end of the simulation are shown in Figure 36.
Figure 35 Changes in the average permeabilities in the three resolutions.

From Figure 34, it can be observed that the three curves generated by the three resolutions are almost the same. In Figure 35, the curve generated by the grid of $20^2$ cells is slightly different from the other two curves which are very close to each other. From Figure 36 it can be observed that the three porosity fields of (b) and (c) are very similar to each other. From the observations, we conclude that the differences between the results of $40^2$ and $60^2$ cells can be negligible. Thus, the grid of $40^2$ cells is fine enough to simulate the 0.2-meter by 0.2-meter porous medium. In such a resolution, the length of a cell is 5 mm.
Figure 36 Porosity fields at the end of the simulation in the three resolutions: (a) $20^2$, (b) $40^2$ and (c) $60^2$.

5.6 Dissolution Experiment

After the sensitivity analysis, the wormhole propagation in a 1-meter by 0.2-meter porous medium using both the Darcy and DBF frameworks can be simulated. The initial porosity in the porous medium complies with the uniform distribution, and the range is from 0.05 to 0.35. The initial permeability of the porous medium also obeys the uniform distribution, and the average value is $10^{-8}$ m$^2$. The other physical parameters are the same as those in the sensitivity analysis. The simulation period is $10^5$ seconds, and each time step is 1000 seconds. From the conclusion of the sensitivity analysis, the simulation grid should have 200 cells in the length direction and 40 cells in the width direction. The distributions of the porosities in both frameworks at five different moments are compared, as shown in Figure 37. The development of four properties in both frameworks is also compared, including the average porosity, the average permeability, the average interfacial area and the flux of the right boundary in the porous medium. Figure 38 to
Figure 41 show the differences in the four properties in both frameworks during the procedure of the matrix acidization.

Figure 37 Distributions of the porosities in the Darcy framework (left) and DBF framework (right) at five different moments. The range of color from blue to red indicates the range of porosity from zero to one.

In Figure 37, it can be observed that at the beginning of the propagation, both frameworks show the formation of some small fingers. As time elapses, two major fingers (upper finger and lower finger) appear. However, the lower finger develops much
faster than the upper finger. In the Darcy framework, the upper finger even disappears at the 10^5th second. Furthermore, it can be observed that there are two major differences between the simulation results of both frameworks. One difference is that the major finger in the Darcy framework is shorter than the one in the DBF framework at all five moments. For example, at the 10^5th second, the major finger of the DBF framework has achieved a breakthrough, while the tip of the major finger of the Darcy framework is still near the right boundary of the porous medium. The other difference is that the major finger in the DBF framework is thinner than the one in the DBF framework at all five moments. Both of these differences show that the DBF framework simulates the wormhole propagation more accurately than the Darcy framework.

Figure 38 Average porosity curves in both the Darcy and DBF frameworks.
In Figure 38, it may be observed that the average porosities in both frameworks are always increasing, which indicates that the matrix is eaten by the acid. The DBF and Darcy framework curves are almost the same before the 3.5*10^4th second, but the Darcy framework curve is higher than the DBF framework curve after that moment. The DBF framework curve tends to converge to a limit, while the Darcy framework curve tends to increase faster. Thus, the DBF framework curve captures the wormhole phenomenon better than the Darcy framework curve. Figure 39 shows the increases in the average permeabilities in both frameworks, with the DBF framework curve increasing much faster than the Darcy framework curve after the 8.9*10^4th second. In Figure 40, both curves are decreasing over the whole simulation procedure, which indicates the decreasing of the average interfacial area with the formation of wormholes. However, the average interfacial area of the Darcy framework decreases faster than that of the DBF framework after the 3.5*10^4th second. This moment coincides with the moment in Figure 38 when faster increase in the Darcy framework curve can be observed. In Figure 41, the right boundary flux of the DBF framework is stable over most of the simulation time at a value of approximately 0.01 m/s, which is just the injection velocity of the acid flow. However, many oscillations can be observed in the Darcy framework curve. This also demonstrates that the numerical results of the DBF framework are more stable than those of the Darcy framework.
Figure 39 Average permeability curves in both the Darcy and DBF frameworks.

Figure 40 Average interfacial area curves in both the Darcy and DBF frameworks.
Figure 41 Right boundary flux curves in both the Darcy and DBF frameworks.

5.7 Performance Evaluation

To evaluate the parallel performance of the code, the benchmark problem in Section 5.4 is tested on Shaheen again. The simulation time is 30 seconds, and there are 1000 time steps. The concentration field is constant in this case, so there is no need to solve the linear system in the concentration field. In this experiment, only one linear system is solved for the pressure and velocity fields. $\varepsilon$ is set to 0.9 to make sure the iterative solver can be applied. The grid has $160^2$ cells. The number of processors increases from 16 to 256. All the I/O cost is omitted. The average total run time and solver time for different number of processors can be observed in Table 9 and Figure 42. The speedup of 16 processors is stipulated as one. From this data, we conclude that the speedup of the solver is close to the ideal linear speedup, although the speedup deteriorates with the
increase in the number of processors. In addition to that, the speedup of the whole parallel code is nearly the same as the speedup of the solver, since the matrix setting time is very small compared to the solver time.

Table 9 Average total run time and solver time of the code when solving the shear driven cavity flow problem. P is the number of processors, and the unit is second.

<table>
<thead>
<tr>
<th>P</th>
<th>Total Run Time</th>
<th>Solver Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>336.03</td>
<td>295.26</td>
</tr>
<tr>
<td>32</td>
<td>173.15</td>
<td>151.94</td>
</tr>
<tr>
<td>64</td>
<td>91.73</td>
<td>80.42</td>
</tr>
<tr>
<td>128</td>
<td>54.12</td>
<td>47.95</td>
</tr>
<tr>
<td>256</td>
<td>34.48</td>
<td>30.61</td>
</tr>
</tbody>
</table>

Figure 42 Run time and speedup of the code when simulating the shear driven cavity flow problem.
6. 3D Parallel Simulation of Wormhole Propagation

6.1 Discretization and Experimenting Field Approach

![Discretization of the 3D domain.](image)

In this chapter, we expand the 2D simulation of last chapter to the 3D case. Firstly, the 3D discretization and experimenting field approach are introduced. The CCFD method is still used for discretization. After the discretization, there are two linear systems: one to solve the pressure and velocity and the other one to solve the concentration. Firstly, the discretization of the first linear system is described. Suppose there is a 3D domain and it is divided into eight cubes as shown in Figure 43. The velocity field is represented as
arrows on the faces of the cubes, and the pressure field is represented as points in the centre of the cubes.

The discretized equations of (41) and (42) form a linear system. For each face vertical to the $x$-axis ($x$-direction face), a discretized $x$-momentum equation can be imposed on it. Suppose the grid is of equal distance in each direction, and for the $x$-direction face in the $i$-th column, $j$-th row and $k$-th lay, (41) can be discretized as

$$
\frac{\rho}{\phi} \frac{u_{x,i,j,k}^{\ell+1} - u_{x,i,j,k}^{\ell}}{\Delta t} = -\frac{\rho}{\phi^2} \left( u_{x,i+1,j,k}^{\ell} - u_{x,i,j,k}^{\ell+1} \right) \frac{1}{2\Delta x} \left( u_{x,i,j+1,k}^{\ell+1} - u_{x,i,j-1,k}^{\ell} \right) \frac{1}{2\Delta y} - \frac{p_{i,j,k}^{\ell+1} - p_{i-1,j,k}^{\ell+1}}{\Delta x} + \frac{\mu}{\phi} \left( \frac{u_{x,i+1,j,k}^{\ell+1} - u_{x,i,j,k}^{\ell+1}}{\Delta x} - \frac{u_{x,i,j,k}^{\ell+1} - u_{x,i,j,k-1}^{\ell+1}}{\Delta x} \right) \frac{1}{\Delta x} + \frac{u_{x,i+1,j,k}^{\ell+1} - u_{x,i,j,k}^{\ell+1}}{\Delta y} \frac{1}{\Delta y} - \frac{u_{x,i,j+1,k}^{\ell+1} - u_{x,i,j,k}^{\ell+1}}{\Delta y} \frac{1}{\Delta y} + \frac{u_{x,i,j,k+1}^{\ell+1} - u_{x,i,j,k}^{\ell+1}}{\Delta z} \frac{1}{\Delta z} - \frac{u_{x,i,j,k}^{\ell+1} - u_{x,i,j,k-1}^{\ell+1}}{\Delta z} \frac{1}{\Delta z} - \frac{\mu}{K} u_{x,i,j,k}^{\ell+1} \right)

\begin{align*}
\text{and the pressure field is represented as points in the centre of the cubes.}
\end{align*}

\begin{align*}
\text{The discretized equations of (41) and (42) form a linear system. For each face vertical to the } x\text{-axis (x-direction face), a discretized x-momentum equation can be imposed on it. Suppose the grid is of equal distance in each direction, and for the x-direction face in the } i\text{-th column, } j\text{-th row and } k\text{-th lay, (41) can be discretized as}
\end{align*}

\begin{align*}
\text{In the equation, the variables of } \rho, \phi, \mu \text{ and } K \text{ are imposed on the faces and their superscripts } \ell \text{ and subscripts } (i,j,k) \text{ are abbreviated. The porosity } \phi \text{ and the permeability tensor } K \text{ on the face are calculated using the harmonic weighting method.}
\end{align*}
\( \rho \) and \( \mu \) on the face are calculated by the single-point upwind method. \( nx, ny \) and \( nz \) stand for the number of cubes in the \( x \)-, \( y \)- and \( z \)-direction, respectively. \( \Delta x \) stands for the distance of adjacent grid points in the \( x \)-direction. The meaning of \( \Delta y \) and \( \Delta z \) is similar. \( \bar{u}_{y,i,j,k}^{\ell} \) stands for the average \( y \)-direction velocity on the face, which is defined as the average value of the four \( y \)-direction velocities imposed on the four \( y \)-direction faces adjacent to the face. The meaning of \( \bar{u}_{z,i,j,k}^{\ell} \) is similar. The discretization of the \( y \)-momentum equation and \( z \)-momentum equation holds the same philosophy. Furthermore, the discretized \( x \)-momentum equations can rewritten as

\[
a_i^{(o)} u_{x,i,j,k} + a_{i-1,j,k}^{(o)} u_{x,i-1,j,k} + a_{i+1,j,k}^{(o)} u_{x,i+1,j,k} + a_{i,j-1,k}^{(o)} u_{x,i,j-1,k} + a_{i,j+1,k}^{(o)} u_{x,i,j+1,k} \\
+ a_{i,j,k-1}^{(o)} u_{x,i,j,k-1} + a_{i,j,k+1}^{(o)} u_{x,i,j,k+1} + b_{i-1,j,k}^{(o)} p_{i-1,j,k} + b_{i,j,k}^{(o)} p_{i,j,k} = c^{(o)},
\]

\( i = 1, \ldots, nx + 1, j = 1, \ldots, ny, k = 1, \ldots, nz. \) \( (54) \)

\( a \) is the coefficient of the \( x \)-direction velocity, \( b \) is the coefficient of the pressure, and \( c \) is a constant term on the right-hand side. The superscript \( \ell + 1 \) is abbreviated in \( (54) \). The superscript \( (o) \) represents the index of this equation in the linear system. In the same way, the discretized \( y \)-momentum equations and \( z \)-momentum equations can be derived. Under Neumann boundary condition, the momentum equations on the boundaries degenerate to

\[ u = u_B. \]

\( u_B \) is the boundary velocity.

\( (42) \) is imposed in the centre of the cube. For the cube in the \( i \)-th column, \( j \)-th row and \( k \)-th lay, \( (42) \) can be discretized as
\[ b'_{i,j,k}^{(o)} p_{i,j,k} + u_{x,i+1,j,k} - u_{x,i,j,k} + u_{y,i,j+1,k} - u_{y,i,j,k} + u_{z,i,j,k+1} - u_{z,i,j,k} = c'^{(o)}, \ i = 1, \ldots, nx, j = 1, \ldots, ny, k = 1, \ldots, nz. \]  

(55)

\( b' \) is the coefficient of pressure, and \( c' \) is the constant term on the right-hand side.

The superscript \( \ell + 1 \) is also abbreviated in (55).

By combining the discretized equations above, a linear system \( Ax = b \) can be generated with

\[ x = (u_{x,1,1,1}, \ldots, u_{x,nx+1,ny,nz}, u_{y,1,1,1}, \ldots, u_{y,nx,ny+1,nz}). \]
\( u_{z,1,1,1}, u_{z,2,1,1}, \ldots, u_{z, nx, ny, nz+1}, p_{1,1,1}, p_{2,1,1}, \ldots, p_{nx, ny, nz} \)\( ^T \).

\( b \) is the right-hand side vector, and \( A \) is the coefficient matrix. In the vector \( x \), the items are arranged in the order of the \( x \)-direction velocities, \( y \)-direction velocities, \( z \)-direction velocities and pressures. Figure 44 shows indices of the unknown variables in \( x \) in a more direct way for the discretization in Figure 43. The indices of the velocities are represented by the numbers at the tails of the arrows, and indices of the pressures are represented by the numbers beside the points. Moreover, matrix \( A \) can be divided into 16 submatrices, and \( Ax = b \) can then be rewritten as

\[
\begin{pmatrix}
A_{xx} & A_{xy} & A_{xz} & A_{xp} \\
A_{yx} & A_{yy} & A_{yz} & A_{yp} \\
A_{zx} & A_{zy} & A_{zz} & A_{zp} \\
A_{cx} & A_{cy} & A_{cz} & A_{cp}
\end{pmatrix}
\begin{pmatrix}
u_x \\
u_y \\
u_z \\
p
\end{pmatrix}
= b.
\]

\( A_{xx} \) consists of the coefficients of \( u_x \) in the \( x \)-momentum equations, \( A_{xp} \) contains the coefficients of \( p \) in the \( x \)-momentum equations, and \( A_{xy} = 0, A_{xz} = 0 \) because there are no \( u_y \) and \( u_z \) in the \( x \)-momentum equations. The meaning of the other submatrices is similar.

It is easy to see that the numbers in Figure 44 also represent the indices of the equations in \( Ax = b \). The numbers on the \( x \)-direction faces stand for the indices of the \( x \)-momentum equations, the numbers on the \( y \)-direction faces stand for the indices of the \( y \)-momentum equations, the numbers on the \( z \)-direction faces stand for the indices of the \( z \)-momentum equations, and the numbers in the cubes stand for the indices of the
continuity equations. Suppose the \( n \)-th equation is an \( x \)-momentum equation, and it is imposed on the face numbered \( n \). It can be expressed as

\[
\begin{align*}
a^{(n)}_{n-(nx+1)*ny} u_{x,n-(nx+1)*ny} + a^{(n)}_{n-(nx+1)} u_{x,n-(nx+1)} + a^{(n)}_{n-1} u_{x,n-1} + a^{(n)}_{n} u_{x,n} \\
+ a^{(n)}_{n+1} u_{x,n+1} + a^{(n)}_{n+(nx+1)} u_{x,n+(nx+1)} + a^{(n)}_{n+(nx+1)*ny} u_{x,n+(nx+1)*ny} + b^{(n)}_{left} p_{left} \\
+ b^{(n)}_{right} p_{right} = c^{(n)}.
\end{align*}
\]

It is noticed that the subscripts are represented by the index numbers here instead of the coordinates as before. \( left \) and \( right \) stand for the indices of the pressure variables on both sides of the face. If one sets \( u_{x,n} = 1 \) and the other unknowns as zero, the left-hand side of the equation is equal to the coefficient of \( u_{x,n} \), which is \( a^{(n)}_{n} \). In the same way, the other coefficients can be calculated easily without knowing their expressions. Furthermore, it is also noticed that there are at most seven equations that contain the unknown variable \( u_{x,n} \). They are the equations numbered \( n-(nx+1)*ny \), \( n-(nx+1) \), \( n-1 \), \( n \), \( n+1 \), \( n+(nx+1) \) and \( n+(nx+1)*ny \), respectively, which can be listed as below

\[
\begin{align*}
a^{(m)}_{m-(nx+1)*ny} u_{x,m-(nx+1)*ny} + a^{(m)}_{m-(nx+1)} u_{x,m-(nx+1)} + a^{(m)}_{m-1} u_{x,m-1} + a^{(m)}_{m} u_{x,m} \\
+ a^{(m)}_{m+1} u_{x,m+1} + a^{(m)}_{m+(nx+1)} u_{x,m+(nx+1)} + a^{(m)}_{m+(nx+1)*ny} u_{x,m+(nx+1)*ny} + b^{(m)}_{left} p_{left} \\
+ b^{(m)}_{right} p_{right} = c^{(m)},
\end{align*}
\]

\( m = n - (nx + 1) * ny, n - (nx + 1), n - 1, n, n + 1, n + (nx + 1), n + (nx + 1) * ny \).

In the experimenting field approach, if one sets \( u_{x,n} = 1 \) and the other unknowns as zero, one can derive seven coefficients at the same time: \( a^{(n-(nx+1)*ny)}_{n} \), \( a^{(n-(nx+1))}_{n} \), \( a^{(n-1)}_{n} \), \( a^{(n)}_{n} \), \( a^{(n+1)}_{n} \), \( a^{(n+(nx+1))}_{n} \) and \( a^{(n+(nx+1)*ny)}_{n} \). Thus, this method needs only \( N \)
loops to derive all the coefficient values of the linear system, with \( N \) equalling the number of unknowns.

The number of loops can be reduced even further. All the unknowns are divided into four fields: the \( x \)-direction velocity field, the \( y \)-direction velocity field, the \( z \)-direction velocity field and the pressure field. In the \( x \)-momentum equations, if one sets \( u_{x,1,1,1} = 1 \), one can derive coefficient values of \( u_{x,1,1,1} \) from the 1st, 2nd, 4th and 7th equations and cannot derive any coefficient values from the other equations because the other equations do not contain the variable \( u_{x,1,1,1} \). With this philosophy, when one sets \( u_{x,i,j,k} = 1 \), one can set \( u_{x,i\pm3,j,k} = 1 \), \( u_{x,i,j\pm3,k} = 1 \) and \( u_{x,i,j,k\pm3} = 1 \) at the same time. Then, one can shift the velocities with the value “1” by one place in a direction and set another group of \( u_x \) as one. The loop continues until all the velocities have been set as one for one time. Thus, a total of only \( 3^3 = 27 \) shifts are needed to derive all the coefficient values of \( u_x \) in the \( x \)-momentum equations, which is \( A_{xx} \). Before the acceleration, \( O(N) \) loops are needed to derive all the coefficients, but now, only 27 shifts are needed, which is \( O(1) \). When \( N \) is large, which is very common in large-scale problems, the acceleration is tremendous.

To derive the coefficient values of the pressures in the \( x \)-momentum equations, the pressure field is shifted. Because each pressure variable appears only in the two \( x \)-momentum equations of its left and right sides, when one sets \( p_{i,j,k} = 1 \), one can set \( p_{i\pm2,j,k} = 1 \) as well. Thus, a total of only two shifts are needed to derive all the \( p \)
Coefficient values in the $x$-momentum equations. All the values make up the nonzero entries of the matrix $A_{xp}$.

We can use the same method to derive all the coefficient values of the $y$-momentum equations and $z$-momentum equations. The field shift procedure of the continuity equations is similar, and the description is omitted.

The discretization of the second linear system is discussed now. (43) is imposed in the centre of the cube. For the cube in the $i$-th column, $j$-th row and $k$-th lay, the left-hand side of (43) is discretized as

$$
\frac{\phi_{i,j,k}^{\ell+1} - \phi_{i,j,k}^{\ell}}{\Delta t} + \frac{u_{x,i+1,j,k}^{\ell+1} - u_{x,i,j,k}^{\ell+1}}{\Delta x} + \frac{u_{y,i,j+1,k}^{\ell+1} - u_{y,i,j,k}^{\ell+1}}{\Delta y} + \frac{u_{z,i,j,k+1}^{\ell+1} - u_{z,i,j,k}^{\ell+1}}{\Delta z} + f_{\ell_{x,y,j,k}},
$$

with

$$
C_{f_{x,i,j,k}}^{\ell+1} = \begin{cases} 
C_{f_{i,j,k}}^{\ell+1} & \text{if } u_{x,i,j,k}^{\ell+1} \leq 0 \\
C_{f_{i-1,j,k}}^{\ell+1} & \text{if } u_{x,i,j,k}^{\ell+1} > 0
\end{cases}
$$

$$
C_{f_{y,i,j,k}}^{\ell+1} = \begin{cases} 
C_{f_{i,j,k}}^{\ell+1} & \text{if } u_{y,i,j,k}^{\ell+1} \leq 0 \\
C_{f_{i,j-1,k}}^{\ell+1} & \text{if } u_{y,i,j,k}^{\ell+1} > 0
\end{cases}
$$

$$
C_{f_{z,i,j,k}}^{\ell+1} = \begin{cases} 
C_{f_{i,j,k}}^{\ell+1} & \text{if } u_{z,i,j,k}^{\ell+1} \leq 0 \\
C_{f_{i,j,k-1}}^{\ell+1} & \text{if } u_{z,i,j,k}^{\ell+1} > 0
\end{cases}
$$

$i = 1, \ldots, nx, j = 1, \ldots, ny, k = 1, \ldots, nz$.

The notation $C_{f_{i,j,k}}$ stands for the concentration in the centre of the cube, while the notations $C_{f_{x,i,j,k}}$, $C_{f_{y,i,j,k}}$ and $C_{f_{z,i,j,k}}$ stand for the concentrations on the $x$-direction, $y$-direction and $z$-direction faces of the cube, respectively. In other words, the
concentration of acid $C_f$ on the face is calculated by the single-point upwind method. In
3D condition,

$$E(u) = \frac{1}{|u|^2} \begin{pmatrix} u_x^2 & u_xu_y & u_xu_z \\ u_yu_x & u_y^2 & u_yu_z \\ u_zu_x & u_zu_y & u_z^2 \end{pmatrix},$$

and $E^+(u) = I - E(u)$ with $I$ being the identity matrix. If $D_e(u)$ is written as

$$D_e(u) = \begin{pmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{pmatrix},$$

and then the first term of the right-hand side of (43) can be discretized as

$$\frac{C_{f_{i,j,k}}^{p+1} - C_{f_{i,j,k}}^p}{\Delta x} \phi_{Fx,i+1,j,k}^p * D_{xx}^{i+1,j,k} \Delta x + \frac{\partial C_{f_{i,j,k}}}{\partial y}_{Fx,i+1,j,k} \phi_{Fx,i,j,k}^p * D_{xy}^{i+1,j,k} \Delta x + \frac{\partial C_{f_{i,j,k}}}{\partial z}_{Fx,i+1,j,k} \phi_{Fx,i,j,k}^p * D_{xz}^{i+1,j,k} \Delta x +$$

$$\frac{\partial C_{f_{i,j,k}}}{\partial y}_{Fy,i+1,j,k} \phi_{Fy,i,j+1,k}^p * D_{yx}^{i+1,j,k} \Delta y + \frac{\partial C_{f_{i,j,k}}}{\partial z}_{Fy,i+1,j,k} \phi_{Fy,i,j,k}^p * D_{yx}^{i+1,j,k} \Delta y +$$

$$\frac{C_{f_{i,j,k}}^{p+1} - C_{f_{i,j,k}}^p}{\Delta y} \phi_{Fy,i,j+1,k}^p * D_{yy}^{i,j+1,k} \Delta y + \frac{\partial C_{f_{i,j,k}}}{\partial z}_{Fy,i,j+1,k} \phi_{Fy,i,j,k}^p * D_{yy}^{i,j+1,k} \Delta y +$$

$$\frac{\partial C_{f_{i,j,k}}}{\partial x}_{Fz,i,j+1,k} \phi_{Fz,i,j,k+1}^p * D_{zx}^{i,j,k+1} \Delta z + \frac{\partial C_{f_{i,j,k}}}{\partial y}_{Fz,i,j,k+1} \phi_{Fz,i,j,k}^p * D_{zx}^{i,j,k+1} \Delta z +$$

$$\frac{\partial C_{f_{i,j,k}}}{\partial z}_{Fz,i,j,k+1} \phi_{Fz,i,j,k}^p * D_{zx}^{i,j,k+1} \Delta z +$$
\[ \frac{dC_f}{dy}_{F_z,i,j,k+1} \star \Phi_{F_z,i,j,k+1}^\ell \star D_{zy} y_{i,j,k+1}^{\ell+1} - \frac{dC_f}{dy}_{F_z,i,j,k} \star \Phi_{F_z,i,j,k}^\ell \star D_{zy} y_{i,j,k}^{\ell+1} \]

\[ \frac{C_{f,i,j,k+1}^{\ell+1} - C_{f,i,j,k}^{\ell+1}}{\Delta z} \star \Phi_{F_z,i,j,k+1}^\ell \star D_{zz} z_{i,j,k+1}^{\ell+1} - \frac{C_{f,i,j,k}^{\ell+1} - C_{f,i,j,k-1}^{\ell+1}}{\Delta z} \star \Phi_{F_z,i,j,k}^\ell \star D_{zz} z_{i,j,k}^{\ell+1} \]

\[ i = 1, \ldots, \text{nx}, j = 1, \ldots, \text{ny}, k = 1, \ldots, \text{nz}. \]

In the above formula, \( \Phi_{F_x,i,j,k}^\ell \), \( \Phi_{F_y,i,j,k}^\ell \) and \( \Phi_{F_z,i,j,k}^\ell \) stand for the porosity on the \( x \)-direction, \( y \)-direction and \( z \)-direction face, respectively. The notation \( \frac{dC_f}{dx}_{F_y,i,j,k} \) stands for the average value of \( \frac{dC_f}{dx} \) on the \( y \)-direction face \((i,j,k)\), which is calculated by the four \( \frac{dC_f}{dx} \) on the four \( x \)-direction faces adjacent to the \( y \)-direction face. The discretization of the other terms of (43) is straightforward. In the discretized equations of (43), the concentrations in the center of the cubes at time step \( \ell + 1 \) are the unknowns. After discretization of (43), a linear system with the unknowns is generated.

When discretizing (43) in the cube with the coordinate \((i,j,k)\), the concentrations \( C_{f,i,j,k}^{\ell+1} \), except the eight concentrations at the eight corners, are considered as the unknowns. Thus, when applying the experimenting field approach on the concentration field, we can set the concentration every other two as one in each direction. In other words, when setting \( C_{f,i,j,k} = 1 \), we can set \( C_{f,i\pm1,j,k} = 1 \), \( C_{f,i,j\pm1,k} = 1 \) and \( C_{f,i,j\pm1,k} = 1 \) at the same time. Therefore, a total of 27 shifts are needed to derive all the coefficient values of \( C_f \) in the linear system.
6.2 Parallelization

The allocation strategy of the equations to the processors in the two linear systems is discussed firstly. For both of the linear systems, each equation should be allocated to one and only one processor. Suppose there is a grid with $nx$, $ny$ and $nz$ cubes in the $x$-direction, $y$-direction and $z$-direction, respectively. We also suppose there are $npx$, $npy$ and $npz$ processors in the $x$-direction, $y$-direction and $z$-direction, respectively, and $nx$, $ny$ and $nz$ are divisible by $npx$, $npy$ and $npz$, respectively. For the more, it is stipulated that $\frac{nx}{npx} \geq 2$, $\frac{ny}{npy} \geq 2$ and $\frac{nz}{npz} \geq 2$. For the first linear system, the allocation of the $x$-momentum equations can be expressed as this: for the processor with the coordinate $(I,J,K)$, the following $x$-momentum equations with the coordinate $(i,j,k)$ are allocated to it

\[
(I - 1) \times \frac{nx}{npx} + 1 \leq i \leq I \times \frac{nx}{npx} + \delta_x,
\]

\[
(J - 1) \times \frac{ny}{npy} + 1 \leq j \leq J \times \frac{ny}{npy},
\]

\[
(K - 1) \times \frac{nz}{npz} + 1 \leq k \leq K \times \frac{nz}{npz},
\]

\[
\delta_x = \begin{cases} 
1, & \text{if } I = npx \\
0, & \text{otherwise}
\end{cases}
\]

\[
1 \leq I \leq npx, 1 \leq J \leq npy, 1 \leq K \leq npz.
\]

With the same philosophy, the $y$-momentum equations allocated to the processor with the coordinate $(I,J,K)$ can be expressed as

\[
(I - 1) \times \frac{nx}{npx} + 1 \leq i \leq I \times \frac{nx}{npx},
\]

\[
(J - 1) \times \frac{ny}{npy} + 1 \leq j \leq J \times \frac{ny}{npy} + \delta_y,
\]

\[
(K - 1) \times \frac{nz}{npz} + 1 \leq k \leq K \times \frac{nz}{npz},
\]

\[
\delta_y = \begin{cases} 
1, & \text{if } J = npy \\
0, & \text{otherwise}
\end{cases}
\]
\[ 1 \leq I \leq npx, 1 \leq J \leq npy, 1 \leq K \leq npz. \]

The \( z \)-momentum equations allocated to the processor with the coordinate \((I,J,K)\) can be expressed as

\[
\begin{align*}
(I - 1) \times \frac{nx}{npx} + 1 & \leq i \leq I \times \frac{nx}{npx}, \\
(J - 1) \times \frac{ny}{npy} + 1 & \leq j \leq J \times \frac{ny}{npy}, \\
(K - 1) \times \frac{nz}{npz} + 1 & \leq k \leq K \times \frac{nz}{npz} + \delta_z,
\end{align*}
\]

\[ \delta_z = \begin{cases} 
1, & \text{if } K = npz \\
0, & \text{otherwise} 
\end{cases} \]

\[ 1 \leq I \leq npx, 1 \leq J \leq npy, 1 \leq K \leq npz. \]

The allocation strategy of the continuity equations is straightforward and can be expressed as this: for the processor with the coordinate \((I,J,K)\), the following continuity equations with the coordinate \((i,j,k)\) are allocated to it

\[
\begin{align*}
(I - 1) \times \frac{nx}{npx} + 1 & \leq i \leq I \times \frac{nx}{npx}, \\
(J - 1) \times \frac{ny}{npy} + 1 & \leq j \leq J \times \frac{ny}{npy}, \\
(K - 1) \times \frac{nz}{npz} + 1 & \leq k \leq K \times \frac{nz}{npz},
\end{align*}
\]

\[ 1 \leq I \leq npx, 1 \leq J \leq npy, 1 \leq K \leq npz. \]

In the second linear system, because the concentration equations are imposed in the centre of the cubes, which is the same as the continuity equations in the first linear system, the allocation strategy of the concentration equations is the same as the one of the continuity equations.

The allocation strategy above is called a block allocation strategy, since the equations are allocated to the processors in the way of “block by block”. The block allocation
strategy can keep the load balance of the processors, and it can also reduce the communication cost among the processors.

After studying the allocation strategy of the equations, it comes to study the domain decomposition strategy for the variables. The domain decomposition strategy holds the similar philosophy as the equation allocation strategy, and the main rule is to allocate different subdomain blocks of the variables with similar sizes to different processors. For different variables, the configurations of the blocks can be different. The domain decomposition strategy for the variable \( x \)-direction velocity is discussed firstly. The variable appears in (41), (42) and (43) in the model. Thus, the \( x \)-direction velocities in the subdomain of the \( x \)-momentum equations of a processor are automatically included in the subdomain of the \( x \)-direction velocity field for the processor. However, the current \( x \)-direction velocity subdomain is not enough to be used to calculate all the coefficient values of (41), (42) and (43) owned by the processor, and some expansions of the subdomain have to be done. For example, in the processor, for the \( x \)-momentum equation with the coordinate \((1,j,k)\), if we use the experimenting field approach to calculate the coefficient value of \( u_{0,j,k}^{\ell+1} \), we have to know the value of \( u_{0,j,k}^{\ell} \) because there exist the convection term and Forchheimer term in (41). Thus, the \( x \)-direction velocities with the coordinates \((0,j,k)\) should be included in the \( x \)-direction velocity subdomain of the processor. The description of the other \( x \)-direction velocity subdomain expansions required by the momentum equations are abbreviated here. Besides that, in order to derive all the coefficient values of (43), the \( x \)-direction velocity subdomain of the
processor needs further expansion. For example, for the concentration equation with the coordinate \((nx,j,k)\), in order to know the coefficient value of \(C_{f,nx+1,j,k}^{\ell+1}\) by the experimenting field approach, the \(x\)-direction velocity with the coordinate \((nx+2,j,k)\) has to be included. Therefore, after all the expansions, for the processor with the coordinate \((I,J,K)\), the coordinate \((i,j,k)\) of its \(x\)-direction velocity subdomain can be expressed as

\[
(l - 1) \frac{nx}{np_x} + 1 - \delta_{xl} \leq i \leq l \frac{nx}{np_x} + 2 - \delta_{xr},
\]

\[
(j - 1) \frac{ny}{np_y} + 1 - 2 \delta_{yl} \leq j \leq j \frac{ny}{np_y} + 2 \delta_{yr},
\]

\[
(k - 1) \frac{nz}{np_z} + 1 - 2 \delta_{zl} \leq k \leq K \frac{nz}{np_z} + 2 \delta_{zr},
\]

\[
\delta_{xl} = \begin{cases} 0, & \text{if } I = 1 \\ 1, & \text{otherwise} \end{cases},
\]

\[
\delta_{xr} = \begin{cases} 1, & \text{if } I = np_x \\ 0, & \text{otherwise} \end{cases},
\]

\[
\delta_{yl} = \begin{cases} 0, & \text{if } J = 1 \\ 1, & \text{otherwise} \end{cases},
\]

\[
\delta_{yr} = \begin{cases} 0, & \text{if } J = np_y \\ 1, & \text{otherwise} \end{cases},
\]

\[
\delta_{zl} = \begin{cases} 0, & \text{if } K = 1 \\ 1, & \text{otherwise} \end{cases},
\]

\[
\delta_{zr} = \begin{cases} 0, & \text{if } K = np_z \\ 1, & \text{otherwise} \end{cases},
\]

\[1 \leq l \leq np_x, \ 1 \leq j \leq np_y, \ 1 \leq k \leq np_z.\]

The expansions of the \(y\)-direction velocity and \(z\)-direction velocity subdomains of the processor hold the same philosophy, and the description is abbreviated.

For the concentration subdomain of the processor, the concentrations in the subdomain of the concentration equations of the processor are automatically included. With the same philosophy as the velocity subdomain expansion, the concentration subdomain of the processor is expanded as below
\[(I - 1) \* \frac{nx}{npx} + 1 - \delta_{xl} \leq i \leq I \* \frac{nx}{npx} + \delta_{xr},\]
\[(J - 1) \* \frac{ny}{npy} + 1 - \delta_{yl} \leq j \leq J \* \frac{ny}{npy} + \delta_{yr},\]
\[(K - 1) \* \frac{nz}{npz} + 1 - \delta_{zl} \leq k \leq K \* \frac{nz}{npz} + \delta_{zr},\]

\[
\delta_{xl} = \begin{cases} 
0, & \text{if } I = 1 \\
1, & \text{otherwise}, \end{cases}
\]
\[
\delta_{xr} = \begin{cases} 
0, & \text{if } I = npx \\
1, & \text{otherwise}, \end{cases}
\]
\[
\delta_{yl} = \begin{cases} 
0, & \text{if } J = 1 \\
1, & \text{otherwise}, \end{cases}
\]
\[
\delta_{yr} = \begin{cases} 
0, & \text{if } J = npy \\
1, & \text{otherwise}, \end{cases}
\]
\[
\delta_{zl} = \begin{cases} 
0, & \text{if } K = 1 \\
1, & \text{otherwise}, \end{cases}
\]
\[
\delta_{zr} = \begin{cases} 
0, & \text{if } K = npz \\
1, & \text{otherwise}, \end{cases}
\]

\[1 \leq i \leq npx, 1 \leq j \leq npy, 1 \leq K \leq npz,\]

with \((i,j,k)\) being the coordinate of the concentration. The domain decomposition strategies of the other variables are similar, and the descriptions are abbreviated.

6.3 Verification

To verify the correctness of the 3D code, the benchmark problem of 3D shear-driven cavity flows [140] is reproduced with it on the supercomputer Shaheen. In the problem, there is a laminar incompressible flow inside a unit cubic cavity whose \(y\)-direction top surface is moved by an \(x\)-direction uniform velocity of 1 m/s, as is shown in Figure 45. The Reynolds number \((Re)\) is 100. The gravity effect is ignored. The grid has \(20^3\) cubes. When the flow becomes stable, the simulation results are shown in Figure 46 and Figure 47. The two figures display the velocity profiles of the \(x\)-direction component on the
vertical centreline and the $y$-direction component on the horizontal centreline of the plane $z = 0.5$ respectively. The simulation results can be compared with Fig. 6 in [140]. To the eyeball resolution, we cannot clarify their differences. An exact comparison is not possible because there are not sufficient details.

Figure 45 3D Shear-driven cavity flow configuration and coordinate system [140].

Figure 46 Velocity profile of the $x$-direction component ($u$) on the vertical centreline of the plane $z = 0.5$. 
6.4 Expansion of Dissolution Experiment

In this section, based on the 2D dissolution experiment, we expand the dissolution experiment to the 3D condition. We consider a matrix of 0.2-m, 1-m and 0.1-m length in the x-direction, y-direction and z-direction, respectively. The initial porosities in the matrix comply with the uniform distribution, and the range is from 0.05 to 0.35. The initial permeability of the matrix also obeys the uniform distribution, and the average value is $10^{-8}$ m$^2$. An acid flow is injected from the boundary of $y = 1$ at a velocity of 0.01 m/s. A constant pressure of $10^5$ Pa is imposed on the boundary of $y = 0$. The other boundaries of the matrix are closed. The initial pressure inside the matrix is also $10^5$ Pa. The length of the time step is $10^3$ s, and the grid has 20, 100 and 10 cubes in the x-direction, y-direction and z-direction, respectively. The other properties of the acid flow
and the matrix are the same as the 2D dissolution experiment. We use both DBF framework and Darcy framework to simulate the matrix acidization procedure.

After the simulation, the distributions of the porosities in both frameworks at four different moments are compared, as is shown in Figure 48. In the figure, the left column shows the simulation results by the Darcy Framework, and the right column shows the simulation results by the DBF framework. The four moments are the 25000th second, the 50000th second, the 75000th second and the 10^5th second, respectively. In the figure, the isosurfaces with the value of 0.8 are shown. It can be observed that there are no obvious fingers forming for the Darcy framework in the whole simulation procedure. However, for the DBF framework, a major finger with some small fingers form, and the acidization has achieved breakthrough at the 10^5th second. In other words, the simulation result of the Darcy framework is more like a facial dissolution, but the simulation result of the DBF framework appears as a ramified wormhole phenomenon.

The development of four properties in both frameworks is also compared, including the average porosity, the average permeability, the average interfacial area and the flux of the out boundary in the matrix. Figure 49 to Figure 52 show the differences in the four properties in both frameworks during the procedure of matrix acidization. From Figure 49, it may be observed that the average porosities in both frameworks are always increasing, which indicates that the matrix is eaten by the acid. The DBF and Darcy framework curves are almost the same before the 2.2*10^4th second, but Darcy framework curve is higher than the DBF framework curve after that moment. The DBF framework
curve tends to converge to a limit, while the Darcy framework curve tends to increase faster. Figure 50 shows the increase of the average permeability in both frameworks, with the DBF framework curve increasing much faster than the Darcy framework curve after the $8.9 \times 10^4$th second. In Figure 51, both curves are decreasing over the whole simulation procedure, which indicates the decreasing of the average interfacial area with the procedure of matrix acidization. However, the average interfacial area of the Darcy framework decreases faster than that of the DBF framework after the $2.2 \times 10^4$th second. The moment coincides with the moment in Figure 49 when the faster increase of Darcy framework curve can be observed. In Figure 52, the right boundary flux of the DBF framework is stable over most of the simulation time at a value of approximately 0.01 m/s, which is the injection velocity of the acid flow. However, many oscillations can be observed in the Darcy framework curve. This demonstrates that the numerical results of the DBF framework are more stable than those of the Darcy framework. From the figure, we can also decide the breakthrough time for the DBF framework, which is about $6.7 \times 10^4$th second. Before the moment, the flux is stable, but the flux increases after the moment.

By expanding the 2D case to the 3D case, it is observed that the conclusions derived from the numerical results of the 2D case can also be applied in the 3D case: the porosity simulation result of the DBF framework shows an obvious ramified wormhole phenomenon, but the one of the Darcy framework shows a facial dissolution phenomenon.
Moreover, the changes of the four properties in the dissolution procedure are also similar in both 2D and 3D conditions.

Figure 48 Porosity isosurfaces of 0.8 in the Darcy framework (left) and DBF framework (right) at four different moments: the 25000th second, the 50000th second, the 75000th second and the 100000th second from above to bottom, respectively.
Figure 49 Average porosity curves in the Darcy and DBF frameworks.

Figure 50 Average permeability curves in the Darcy and DBF frameworks.
Figure 51 Average interfacial area curves in the Darcy and DBF frameworks.

Figure 52 Out boundary flux curves in the Darcy and DBF frameworks.
6.5 3D Dissolution Patterns

To the best of our knowledge, no work has ever studied the dissolution patterns of matrix acidization in the DBF framework, not to mention the 3D case. Thus, this section investigates the topic. It is known from [83] that a lot of factors will affect the dissolution patterns of matrix acidization, such as convection and transverse dispersion, the acid injection rate and the magnitude of heterogeneity. However, we will not study the effects of all the factors in this work. Instead, the effect of the acid injection rate on the dissolution patterns is discussed.

Based on the experiment in last section, the acid injection velocity is changed. Figure 53 shows the porosity isosurfaces of 0.8 at different injection velocities ($v$). In Figure 53(a), $v = 0.0001$, which is very small. The acid is spent immediately by the rocks, and a facial dissolution pattern is formed. The penetration depth of the acid is also limited to a region which is not far from the injection boundary. In Figure 53(b-d), three different types of conductive channels are observed. When $v = 0.0003$, the channel is in the conical pattern; when $v = 0.001$, the channel is in the wormhole pattern; when $v = 0.01$, the channel is in the ramified pattern. It can be observed that with the increase of the injection velocity, more acid reacts with the rocks, so that in Figure 53(e), when $v = 0.1$, a uniform pattern forms. In the uniform pattern, a very high-velocity acid is injected into the matrix, and it penetrates into the matrix deeply. However, the permeability is not increased largely in this way, since the acid reacts over a large region. Thus, too low or too high acid velocity cannot develop channels to facilitate the flow of
the oil. In order to enhance the recovery rate of the oil, it is required to produce wormholes with the minimum amount of acid. Furthermore, it can be observed that the DBF framework can also develop five different patterns of matrix acidization, which coincides with the simulation results based on the Darcy framework.

(a) \( v = 0.0001 \), face.
(b) $v = 0.0003$, conical.

(c) $v = 0.001$, wormhole.
(d) $v = 0.01$, ramified.

(e) $v = 0.1$, uniform.

Figure 53 Five dissolution patterns simulated by the DBF framework.
6.6 Performance Evaluation

Table 10 Properties of the acid flow and matrix

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viscosity of acid flow ($\mu$)</td>
<td>$10^{-2}$ Pa·s</td>
</tr>
<tr>
<td>Density of acid flow ($\rho$)</td>
<td>$10^3$ kg/m$^2$</td>
</tr>
<tr>
<td>Longitudinal dispersion coefficient</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>Transverse dispersion coefficient</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>Molecular diffusion coefficient</td>
<td>$10^{-9}$</td>
</tr>
<tr>
<td>Surface reaction rate constant ($k_s$)</td>
<td>$10^{-2}$ m/s</td>
</tr>
<tr>
<td>Local mass-transfer coefficient ($k_c$)</td>
<td>$10^{-3}$ m/s</td>
</tr>
<tr>
<td>Dissolving power of acid ($\alpha$)</td>
<td>0.1 kg/mole</td>
</tr>
<tr>
<td>Density of the rock ($\rho_s$)</td>
<td>2500 kg/m$^2$</td>
</tr>
<tr>
<td>Initial interfacial area ($a_0$)</td>
<td>0.5 m$^2$</td>
</tr>
<tr>
<td>Initial concentration of acid flow ($C_f^0$)</td>
<td>10 mole/m$^2$</td>
</tr>
</tbody>
</table>

To evaluate the parallel performance of the 3D code with the DBF framework, a case is designed based on the 3D benchmark shear driven cavity flow problem. There is a laminar compressible flow inside a unit cubic cavity whose y-direction top surface is moved by an x-direction uniform velocity of 1 m/s, as is shown in Figure 45. The other surfaces of the cavity are closed. There is gravity in the y-direction. The initial porosity is 0.2 everywhere, and the initial permeability is $10^{-8}$ m$^2$ everywhere. The initial pressure is zero in the cavity. The initial concentration of the acid is also zero in the cavity. We inject the acid in the cubes with the coordinate of $x = 50$ and $y = 100$, and the injection rate is 1 mole/m$^3$·s. The simulation grid has 100 cubes in each direction. The simulation time is 16000 seconds, and there are 1000 time steps. The other parameters of the case are shown in Table 10. In order to leverage the iterative solver of Hypre, $\varepsilon$ is set to 0.1. The preconditioner DS and the solver GMRES in Hypre are chosen to solve $Ax = b$. The case is tested on Shaheen. All the I/O cost is omitted.
Table 11 Average run time and solver time of the parallel code. P is the number of processors, and the unit is second.

<table>
<thead>
<tr>
<th>P</th>
<th>Run Time</th>
<th>Solver Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>6706.9</td>
<td>3480.3</td>
</tr>
<tr>
<td>250</td>
<td>3602.3</td>
<td>1852.4</td>
</tr>
<tr>
<td>500</td>
<td>1965.8</td>
<td>1020.1</td>
</tr>
<tr>
<td>1000</td>
<td>1019.6</td>
<td>503.8</td>
</tr>
</tbody>
</table>

Figure 54 Run time, solver time and speedup of different number of processors.

Since this section focuses on the performance evaluation of the code, the physical results of the case are not given. The average run time and solver time for different number of processors can be observed in Table 11 and Figure 54. The speedup of 125 processors is stipulated as one. For the performance, the speedup of the solver and the whole parallel code is very close to the ideal linear speedup.

7.1 Introduction

In this chapter, mixed finite element method (MFEM) is used to analyze the wormhole model. The advantages of MFEM can be listed as below: firstly, they are capable to achieve accurate and stable approximations for both pressure variables and flux variables across grid-cell interfaces; secondly, they also have the local mass conservation property. Therefore, MFEM is preferable instrument for the wormhole simulation. As stated in [143], CCFD used in last two chapters is equivalent to MFEM when the mesh is rectangular and the function space is RT₀ (Raviart-Thomas), thus the analysis in this chapter is also applied to the wormhole simulations in last two chapters.

Considering the complexity to analyze the non-linear terms of \( \frac{\mu}{
abla^2} \nabla \cdot \mathbf{u} \mathbf{u} \) and \( \nabla^2 \mathbf{u} \) in (41), the two terms are omitted in the following analysis. A complete analysis of (41) will be future work. Because the Brinkman term is not used here, the framework to be analyzed is called the Darcy-Forchheimer (DF) framework. MFEM has been developed for DF framework in [144]-[149]. In the proposed MFEM for wormhole simulations, the DF equation is similar to that introduced in [147]. However, our methods take into account the change in the porosity and permeability, which is a distinct physical property of wormholes. A numerical scheme has been proposed in [150] to solve the incompressible miscible displacement problem with the DF framework. In their scheme, MFEM is used to handle the DF equation, and standard FEM is used to approximate the
concentration. Compared to the problem in [150], the wormhole problem is more complicated because of the change in the porosity and permeability. Furthermore, different from [150] which uses standard FEM for the solute reaction transport equation, our proposed scheme uses MFEM. Actually, in our proposed scheme, to guarantee mass conservation of the fluid, MFEM is used not only for the DF equation and mass conservation equation, but also for the solute reaction transport equation by introducing an auxiliary flux variable to reform (43). Thus, compared with these existing results, there are at least three main issues in the theoretical analysis of the proposed methods for wormhole simulations: the first one is to control and handle the change in the porosity with time, the second one is the complication resulted from an auxiliary flux variable, and the third one is the fully coupling relation of multivariable. To resolve the first issue, based on the cut-off operator of the solute concentration, an analytical function is constructed to bound the porosity; for the second issue, we define an auxiliary function of the velocity and establish its properties; for the third issue, we employ the coupled analysis approach to achieve the estimates of multivariable.

7.2 Mixed Weak Formulation

Suppose that $\Omega \subset \mathbb{R}^d$ ($d = 2,3$) is a polygonal and bounded domain. Here, the traditional notations are used for the Lebesgue spaces $L^p(\Omega)$ and the Sobolev spaces $W^{k,p}(\Omega)$ with $1 \leq p \leq \infty$ over the domain $\Omega$. In particular, $W^{k,2}(\Omega)$ is denoted by $H^k(\Omega)$. The corresponding norms are denoted by $\|\cdot\|_{L^p(\Omega)}$, $\|\cdot\|_{W^{k,p}(\Omega)}$ and $\|\cdot\|_{H^k(\Omega)}$, 

respectively. For simplicity, we also denote \( \| \cdot \|_{L^p(\Omega)} \) by \( \| \cdot \|_L^P \) and \( \| \cdot \|_{L^2(\Omega)} \) by \( \| \cdot \|_L^2 \).

The inner product of \( L^p(\Omega) \) or \( (L^p(\Omega))^d \) is denoted by \( (\cdot, \cdot)_r \), and the inner product of the functions in the boundary \( \Gamma \) is denoted by \( (\cdot, \cdot)_r \). We also use \( | \cdot | \) to represent the 2-norm of a vector in \( \mathbb{R}^d \) or a matrix in \( \mathbb{R}^{d \times d} \). The norms for a vector-value function \( \mathbf{v} \in (L^p(\Omega))^d \) and a matrix-value function \( \mathbf{A} \in (L^p(\Omega))^{d \times d} \) are defined as

\[
\| \mathbf{v} \|_{(L^p(\Omega))^d} := \| \mathbf{v} \|_{L^p(\Omega)},
\]

\[
\| \mathbf{A} \|_{(L^p(\Omega))^{d \times d}} := \| \mathbf{A} \|_{L^p(\Omega)}.
\]

For notation simplicity, the norms \( \| \cdot \|_{L^2(\Omega)}, \| \cdot \|_{(L^2(\Omega))^d} \) and \( \| \cdot \|_{(L^2(\Omega))^{d \times d}} \) are also written as \( \| \cdot \| \) for scalar-value, vector-value and matrix-value functions, respectively.

Let \( T \) be the final simulation time. For a given normed space \( X \) and \( 1 \leq p \leq \infty \), we define

\[
L^p(0, T, X) := \{ \varphi: \varphi(t) \in X, \| \varphi \|_X \in L^p(0, T) \},
\]

which is a normed linear space equipped with the norm given by

\[
\| \varphi \|_{L^p(0, T; X)} := \| (\| \varphi \|_X) \|_{L^p(0, T)}.
\]

We now describe the initial boundary conditions of the wormhole model. The initial pressure \( p_0 \), initial velocity \( \mathbf{u}_0 \) and initial acid concentration \( C_f^0 \) are provided. In addition to that, we also need to specify the initial porosity \( \phi_0 \), initial permeability \( K_0 \) and initial interfacial area \( a_0 \). We consider the following boundary conditions for wormhole problem

\[
\mathbf{u} \cdot \mathbf{n} = 0, \mathbf{x} \in \partial \Omega, t \in (0, T],
\]

\[
(\mathbf{u} C_f - \phi D_e \nabla C_f) \cdot \mathbf{n} = 0, \mathbf{x} \in \partial \Omega, t \in (0, T].
\]
Here, $n$ denotes the unit outward normal vector to $\partial \Omega$.

For the physical quantities, we assume that $0 < \phi_0, \leq \phi_0 \leq \phi_0^* < 1$, $0 < K_0, \leq K_0 \leq K_0^*$ and $0 < a_0, \leq a_0 \leq a_0^*$. We introduce the functions $f_P$ and $f_I$ to represent the production and injection rates respectively, and $C_I$ is the injected concentration of acid. We assume that $f_P, f_I, C_I \in L^\infty(0,T;L^\infty(\Omega))$ and $f_P \leq 0$, $f_I \geq 0$, $0 \leq C_I \leq 1$.

In addition to that, we assume that $\rho, \mu, k_c$ and $k_s$ are all given positive constants.

Define two functional spaces as

$$V = \{ v \in (L^3(\Omega))^d : \nabla \cdot v \in L^2(\Omega) \},$$

and

$$V_0 = \{ v \in V : v \cdot n = 0 \text{ on } \partial \Omega \}.$$

Furthermore, we define another functional spaces as

$$H(div; \Omega) = \{ v \in (L^2(\Omega))^d : \nabla \cdot v \in L^2(\Omega) \},$$

and

$$H_0(div; \Omega) = \{ v \in H(div; \Omega) : v \cdot n = 0 \text{ on } \partial \Omega \}.$$

Using the relation (46), the solute reaction transport equation (43) becomes

$$\frac{\partial (\phi C_f)}{\partial t} + \nabla \cdot (u C_f) - \nabla \cdot (\phi D_e \nabla C_f) = -\frac{k_c k_s}{k_c+k_s} a_v C_f + f_P C_f + f_I C_I. \quad (56)$$

It is noticed that $f_P$, $f_I$ and $C_I$ are considered in (56). In (47), $u = [u_1, \cdots , u_d]^T$ and

$$(E(u))_{ij} = \frac{u_i u_j}{|u|^2}, \quad 1 \leq i, j \leq d.$$

To apply MFEM to (56), we define an auxiliary flux variable $q$ as

$$q = u C_f - \phi D_e \nabla C_f.$$

Then (56) is rewritten as
\[ \frac{\partial (\phi C_f)}{\partial t} + \nabla \cdot q = -\frac{k_c k_s}{k_c + k_s} a_v C_f + f_p C_f + f_i C_i. \]

Furthermore, we define an auxiliary function of the velocity as

\[ M_e(u) = (D_e(u))^{-1} u. \]

Thus, the mixed weak form describing the wormhole problem is formulated as below

\[ \varepsilon \left( \frac{\partial p}{\partial t}, \phi \right) + \left( \frac{\partial \phi}{\partial t}, \varphi \right) + (\nabla \cdot u, \varphi) = (f, \varphi), \varphi \in L^2(\Omega), \]

\[ \left( \frac{\rho}{\phi} \frac{\partial u}{\partial t}, v \right) + \left( \frac{\mu}{K(\phi)} u, v \right) + \left( \frac{\rho F(\phi)}{\sqrt{K(\phi)}} |u| u, v \right) = (p, \nabla \cdot v) + (\rho g, v), v \in V_0, \]

\[ \left( \frac{\partial (\phi C_f)}{\partial t}, \psi \right) + (\nabla \cdot q, \psi) + \frac{k_c k_s}{k_c + k_s} (a_v(\phi) C_f, \psi) - (f_p C_f, \psi) = (f_i C_i, \psi), \psi \in L^2(\Omega), \]

\[ \left( \phi D_e(u) \right)^{-1} q, w \right) = (\phi^{-1} M_e(u) C_f, w) + (C_f, \nabla \cdot w), \quad w \in H_0(div; \Omega). \]

Denote \( R_\phi = \frac{a_0 k_c k_s}{\rho_s (k_c + k_s)}. \) Note that because of the boundedness of \( a_0, \) \( R_\phi \) is also bounded as \( 0 \leq R_{\phi_*} \leq R_\phi \leq R_{\phi*}. \) Using the relations (46), (48), (49) and (50), the evolution of porosity is reduced to

\[ \frac{\partial \phi}{\partial t} = \frac{R_\phi(1 - \phi)}{1 - \phi_0} C_f. \]

We use a partition \( \varepsilon_h \) of the domain \( \Omega, \) which is composed of triangles or quadrilaterals if \( d = 2 \) and tetrahedra, prism or hexahedra if \( d = 3. \) Denote by \( h_E \) the diameter of \( E \in \varepsilon_h \) and let \( h = \max_{E \in \varepsilon_h} h_E \) such that \( h/h_E \leq C. \) We assume that \( \varepsilon_h \) is non-degenerate, which means each element is convex, and that there exists \( \rho > 0 \) such that each of the sub-triangles (for \( d = 2 \)) or sub-tetrahedra (for \( d = 3 \)) of element \( E \) contains a ball of radius \( \rho h_E \) in its interior. Denote by \( \Gamma_h \) the set of element boundaries and assume that no element crosses the boundaries of \( \partial \Omega. \) Let the
approximating subspaces \( X \subset V, \ X_0 \subset V_0, \ Y \subset H(div; \Omega), \ Y_0 \subset H_0(div; \Omega) \) and \( \mathcal{W} \subset L^2(\Omega) \) be the \( r \)th order \((r \geq 0)\) Raviart-Thomas space \((\text{RT}_r)\) on the partition \( \epsilon_h \).

Define the cut-off operator \( \mathcal{M} \) for velocity as

\[
\mathcal{M}(u) := \begin{cases} u, & \text{if } |u| \leq M, \\ M \frac{u}{|u|}, & \text{if } |u| > M, \end{cases}
\]

where \( M \) is a given large positive constant.

Denote \( J = (0, T] \), where \( T > 0 \). The MFEM formulation is stated as below: find \( p_h(\cdot, t), \ C_h(\cdot, t) \in \mathcal{W} \) and \( u_h(\cdot, t) \in X_0, \ q_h(\cdot, t) \in Y_0(t \in J) \) such that

\[
\varepsilon \left( \frac{\partial p_h}{\partial t}, \phi \right) + \left( \frac{\partial \phi_h}{\partial t}, \phi \right) + (\nabla \cdot u_h, \phi) = (f, \phi), \ \phi \in \mathcal{W},
\]

\[
\left( \frac{\rho}{\phi_h} \frac{\partial u_h}{\partial t}, v \right) + \left( \frac{\mu}{K(\phi_h)} u_h, v \right) + \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} |u_h| u_h, v \right) \nonumber
\]

\[
= (p_h, \nabla \cdot v) + (\rho g, v), \ v \in X_0,
\]

\[
\left( \frac{\partial (\phi_h C_h)}{\partial t}, \psi \right) + (\nabla \cdot q_h, \psi) + \frac{k_c k_s}{k_c + k_s} (a_v(\phi_h) C_h, \psi) - (f_p C_h, \psi)
\]

\[
= (f, \psi), \ \psi \in \mathcal{W},
\]

\[
((\phi_h D_e(u_h^M))^{-1} q_h, w) = (\phi_h^{-1} M_e(u_h) C_h, w) + (C_h, \nabla \cdot w), \ w \in Y_0,
\]

where \( u_h^M = \mathcal{M}(u_h) \). The discrete porosity is computed as

\[
\frac{\partial \phi_h}{\partial t} = \frac{R(1-\phi_h)}{1-\phi_0} \bar{C}_h,
\]

where \( \phi_h(\cdot, 0) = \phi(\cdot, 0) = \phi_0 \) and \( \bar{C}_h = \max(0, \min(C_h, 1)) \). The discrete porosity is a point-wise function, and it will be shown that \( \phi_h \in L^\infty(J \times \Omega) \) and \( \frac{\partial \phi_h}{\partial t} \in L^\infty(J \times \Omega) \).
7.3 Stability Analysis

In this section, we will prove a priori bounds for discrete solutions $\phi_h$, $u_h$, $p_h$ and $C_h$. In what follows, we use $C$ to indicate a generic positive constant independent of mesh size, but the values of $C$ are probably different in different occurrences.

For the purpose of theoretical analysis, we first analyze the properties of $D_e$ and $M_e$.

**Lemma 1.** Let $D_e(u)$ be defined as in (47) and $d_m > 0$. Suppose that $d_l$ and $d_t$ are non-negative functions of domain $x \in \Omega$ and are uniformly bounded as $d_l(x) \leq d_l^*$ and $d_t(x) \leq d_t^*$, then for $u, v \in \mathbb{R}^d$,

\[
|D_e(u)^{-1}v| \leq (d_m + \min(d_l, d_t)|u|)^{-1}|v| \leq d_m^{-1}|v|, \quad (64)
\]

\[
|D_e(u)v| \leq (d_m + \max(d_l, d_t^*)|u|)|v|, \quad (65)
\]

\[
|D_e(u)^{-1} - D_e(v)^{-1}| \leq d_m^{-2}(7\alpha_t^* + 6\alpha_l^*)d^3|u - v|, \quad (66)
\]

where $d$ is the dimension of domain $\Omega$.

**Proof.** The inequalities (64) and (65) are deduced from the uniform positive definiteness of $D_e(u)$, which is proven in [143]. Because

\[
|D_e(u)^{-1} - D_e(v)^{-1}| \leq |D_e(u)^{-1}||D_e(u) - D_e(v)||D_e(v)^{-1}|
\]

\[
\leq \frac{|D_e(u) - D_e(v)|}{(d_m + \min(d_l, d_t)|u|)(d_m + \min(d_l, d_t)|v|)},
\]

the inequality (66) is obtained by applying the uniform Lipschitz continuity of $D_e(u)$ proven in [143]. □

**Lemma 2.** Let $M_e(u)$ be defined as in (57) and $d_m > 0$. Suppose that $d_l$ and $d_t$ are positive functions of domain $x \in \Omega$, $0 < d_l^* \leq d_l(x) \leq d_l^*$ and $0 < d_t^* \leq d_t(x) \leq d_t^*$. Then for $u, v \in \mathbb{R}^d$,
where $L_M = d_m^{-1} + (d_m \min(d_{ls}, d_{ts}))^{-1}(7d_t^* + 6d_t^*)d^{3/2}$ and $d$ is the dimension of domain $\Omega$.

**Proof.** With the definition of $M_e(u)$, the inequality (67) is a direct result of (64) when taking $v = u$. From (67) and Lemma 1, we have

$$|M_e(u) - M_e(v)| \leq L_M|u - v|,$$

(68)

Applying the uniform Lipschitz continuity of $D_e(u)$ proven in [143], we obtain the inequality (68). □

We now prove the stability results for the discrete solutions of (59) - (63).

**Theorem 1.** The discrete porosity $\phi_h$ is bounded almost everywhere, i.e., $\phi_0 \leq \phi_h \leq 1$, where $\phi_0 > 0$ is the initial porosity, and more precisely,

$$\phi_0 \leq \phi_h \leq 1 - (1 - \phi_0)e^{-\zeta t}, \quad t \in J,$$

(69)

where $\zeta = \frac{R \phi}{1 - \phi_0}$. It also holds that

$$0 \leq \frac{\partial \phi_h}{\partial t} \leq R \phi, \quad t \in J.$$

(70)

**Proof.** It is known that $\zeta > 0$. Define $(\phi_h - 1)^+ = \max(\phi_h - 1, 0)$. We rewrite (63) as

$$\frac{\partial (\phi_h - 1)}{\partial t} = -\zeta \bar{C}_h (\phi_h - 1).$$

(71)

Multiplying (71) by $(\phi_h - 1)^+$ gives
\[ \frac{1}{2} \frac{\partial((\phi_h - 1)^+)^2}{\partial t} = -\zeta C_h ((\phi_h - 1)^+)^2 \leq 0. \]

Noticing that \((\phi_h - 1)^+=0\) at \(t=0\), we deduce from (71) that \(((\phi_h - 1)^+)^2 \leq 0\) for \(t > 0\). This means that \(((\phi_h - 1)^+)^2 = 0\) for \(t > 0\) because we also have \(((\phi_h - 1)^+)^2 \geq 0\). Thus, \(\phi_h \leq 1\) always holds for \(t > 0\). It is observed from (63) that the discrete porosity is increasing with time, therefore \(\phi_h \geq \phi_0\).

We now estimate the more precise bound of \(\phi_h\) with time. Because \(0 < \phi_0 < 1\), we can choose \(t > 0\) such that \(\phi_h < 1\). Noticing that
\[ \frac{\partial(\phi_h - 1)}{\partial t} \leq \zeta (1 - \phi_h), \]
we deduce that
\[ \frac{\partial \ln(1 - \phi_h)}{\partial t} = \frac{1}{1 - \phi_h} \frac{\partial(1 - \phi_h)}{\partial t} \geq -\zeta. \]

Integrating the above equation from 0 to \(t\) gives
\[ \ln(1 - \phi_h) \geq \ln(1 - \phi_0) - \zeta t, \]
and furthermore
\[ 1 - \phi_h \geq (1 - \phi_0) e^{-\zeta t}. \]

Thus, the estimate (69) is obtained. The inequality (70) is reached by the boundedness of \(\phi_h\). \(\square\)

**Lemma 3.** The permeability \(K(\phi_h)\) is bounded below, i.e., \(K(\phi_h) \geq K_0\), where \(K_0 > 0\) is the initial permeability. Moreover, for given time \(T > 0\),
\[ K(\phi_h) \leq K_0 \frac{\phi_T (\phi_T (1 - \phi_0))^2}{\phi_0 (\phi_0 (1 - \phi_T))}, \]
where $\phi_T = 1 - (1 - \phi_0)e^{-\xi T}$ and $\xi = \frac{R \phi}{1 - \phi_0}$. In addition to that, as the functions of $\phi(\phi_0 \leq \phi < 1), K(\phi)^{-1}$ and $F(\phi)K(\phi)^{-1/2}$ are uniformly Lipschitz continuous.

**Proof.** The proof is immediately obtained by combining Theorem 1 and the monotonicity of $K(\phi)$ and $F(\phi)$. □

**Theorem 2.** The approximate solutions of (59) - (62) satisfy

$$
\|p_h\|^2_{L^\infty(J; L^2)} + \|u_h\|^2_{L^\infty(J; L^2)} + \|u_h\|^3_{L^2(J; L^2)}
\leq C \left( |\rho g|^2 + R^2 \phi^2 + \|p_0\|^2 + \|u_0\|^2 + \|f\|^2_{L^2(J; L^2)} \right),
$$

(72)

$$
\|C_h\|^2_{L^\infty(J; L^2)} + \|D_h(u_h^M)^{-1/2} q_h \|^2_{L^2(J; L^2)} \leq C \left( \|C_{f0}\|^2 + \|f_i C_i\|^2_{L^2(J; L^2)} \right).
$$

(73)

**Proof.** Take $\varphi = p_h$ in (59) and $\nu = u_h$ in (60), we obtain

$$
\frac{1}{2} \frac{\partial}{\partial t} \|p_h\|^2 + \left( \frac{\partial \phi_h}{\partial t}, p_h \right) + (\nabla \cdot u_h, p_h) = (f, p_h),
$$

(74)

$$
\left( \frac{\rho}{\phi_h} \frac{\partial u_h}{\partial t}, u_h \right) + \left( \frac{\mu}{K(\phi_h)} u_h, u_h \right) + \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} |u_h| u_h, u_h \right)
\leq (p_h, \nabla \cdot u_h) + (\rho g, u_h).
$$

(75)

Notice that

$$
\left( \frac{\rho}{\phi_h} \frac{\partial u_h}{\partial t}, u_h \right) = \frac{1}{2} \left( \frac{\partial}{\partial t} \left( \frac{\rho}{\phi_h} u_h, u_h \right) + \left( \frac{\rho}{\phi_h^2} \frac{\partial \phi_h}{\partial t} u_h, u_h \right) \right),
$$

and we sum the two equations (74) and (75) to get

$$
\frac{\varepsilon}{2} \frac{\partial}{\partial t} \|p_h\|^2 + \frac{1}{2} \frac{\partial}{\partial t} \left( \frac{\rho}{\phi_h} u_h, u_h \right) + \left( \frac{\mu}{K(\phi_h)} u_h, u_h \right) + \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} |u_h| u_h, u_h \right)
\leq (f, p_h) + (\rho g, u_h) - \left( \frac{\partial \phi_h}{\partial t}, p_h \right) - \frac{1}{2} \left( \frac{\rho}{\phi_h^2} \frac{\partial \phi_h}{\partial t} u_h, u_h \right).
$$

It follows by using Theorem 1 and Young’s inequality that
\[
\frac{\varepsilon}{2} \frac{\partial}{\partial t} \left\| p_h \right\|^2 + \frac{1}{2} \frac{\partial}{\partial t} \left( \frac{\rho}{\phi_h} u_h, u_h \right) + \mu \left\| K(\phi_h)^{-1/2} u_h \right\|^2 + \left\| \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} \right\|_{L^3}^3 \leq \frac{1}{2} \left\| \rho g \right\|^2_{L^2} + \frac{1}{2} \left\| u_h \right\|^2 + \frac{1}{2} R^*_\phi + \frac{1}{2} \left\| f \right\|^2 + \left\| p_h \right\|^2,
\]

where we have also used the fact \( \left( \frac{\rho}{\phi_h} \frac{\partial \phi_h}{\partial t} u_h, u_h \right) \geq 0 \). With the boundedness of \( \phi_h \), \( F(\phi_h) \) and \( K(\phi_h) \), integrating (76) in time from 0 to \( t \) \( (t \leq T) \) gives

\[
\left\| p_h \right\|^2(t) + \left\| u_h \right\|^2(t) + \int_0^t \| u_h \|_2^2 + \int_0^t \| u_h \|_{L^2}^3 \leq C \left\| p_0 \right\|^2 + C \left\| u_0 \right\|^2 \]
\[
+ C \left\| \rho g \right\|^2_{L^2} + C R^*_\phi + C \left\| f \right\|^2_{L^2(J;L^2)} + C \int_0^t \left( \left\| u_h \right\|^2 + \left\| p_h \right\|^2 \right).
\]

The estimate (72) is obtained by using Gronwall’s lemma to the above inequality.

We now turn to prove (73). Taking \( \psi = C_h \) in (61) and \( w = q_h \) in (62) leads to

\[
\left( \frac{\partial (\phi_h C_h)}{\partial t}, C_h \right) + (\nabla \cdot q, C_h) + k_{c+k_s} (a_v(\phi_h) C_h, C_h) - (f_p C_h, C_h) = (f_I C_h, C_h),
\]

\[
\left( (\phi_h D_e(u_h^M))^{-1} q_h, q_h \right) = (\phi_h^{-1} M_e(u_h) C_h, q_h) + (C_h, \nabla \cdot q_h).
\]

Because \( \left( \frac{\partial \phi_h}{\partial t}, C_h \right) \geq 0 \), we get

\[
\left( \frac{\partial (\phi_h C_h)}{\partial t}, C_h \right) = \frac{1}{2} \frac{\partial}{\partial t} (\phi_h C_h, C_h) + \frac{1}{2} \left( \frac{\partial \phi_h}{\partial t} C_h, C_h \right) \geq \frac{1}{2} \frac{\partial}{\partial t} (\phi_h C_h, C_h).
\]

Noticing that \( k_{c+k_s} (a_v(\phi_h) C_h, C_h) \geq 0 \) and \( f_p \leq 0 \), we sum the equations (77) and (78) and obtain

\[
\frac{1}{2} \frac{\partial}{\partial t} \left\| \phi_h^{1/2} C_h \right\|^2 + \left\| (\phi_h D_e(u_h^M))^{-1/2} q_h \right\|^2 \leq (\phi_h^{-1} M_e(u_h) C_h, q_h) + (f_I C_h, C_h).
\]

For the first term on the right-hand side of (79), the boundedness of \( \phi_h \), \( D_e(u_h^M) \) and \( M_e(u_h) \) gives us

\[
(\phi_h^{-1} M_e(u_h) C_h, q_h) \leq C \left\| C_h \right\|^2 + \frac{1}{2} \left\| (\phi_h D_e(u_h^M))^{-1/2} q_h \right\|^2.
\]
With the estimate \((f_{f I}, C_h) \leq \frac{1}{2}((f_{f I})^2 + \|C_h\|^2)\), it is followed by integrating (79) in time from 0 to \(t \leq T\) and combining (80)

\[
\|C_h\|^2(t) + \int_0^t \|(\phi_h D_e(u_h^M))^{-1/2} q_h\|^2 \leq C\|C_{f0}\|^2 + C\|f_{f I}\|^2_{L^2(J;L^2)} + C\int_0^t \|C_h\|^2,
\]

from which, (73) is obtained by Gronwall’s lemma. □

7.4 Error Estimate

In this section, the approximate errors of MFEM solutions of (59) - (63) are estimated. Denote by \(h\) the mesh size and \(h \leq h_0(h_0 > 0)\). Furthermore, denote by \(r\) the order of Raviart-Thomas spaces. Let \(P_h\) be \(L^2\)-projection operator, there exists a unique \(P_h p \in \mathcal{W}\) such that

\[(p - P_h p, \omega) = 0, \quad \forall \omega \in \mathcal{W}.
\]

Moreover, \(P_h\) satisfies the following approximation property

\[
\|P_h p - p\| \leq C\|p\|_{H^j(\Omega)}, \quad 0 \leq j \leq r + 1.
\]

Let \(\Pi_h\) denote the usual Raviart-Thomas projection which has the following properties [152]-[156]

\[
(\nabla \cdot (u - \Pi_h u), \omega) = 0, \quad \omega \in \mathcal{W},
\]

\[
\|u - \Pi_h u\|_{L^p} \leq C\|u\|_{W^{j,p}(\Omega)}^\alpha h^\gamma, \quad 1 \leq j \leq r + 1,
\]

Assume that the exact solutions satisfy the following regularity requirements:

1) \(p \in L^\infty(J;H^s(\Omega)), \frac{\partial p}{\partial t} \in L^2(J;H^s(\Omega))\) and \(p_0 \in H^s(\Omega)\);

2) \(u \in (L^\infty(J;W^{s,q}(\Omega)))^d \cap (L^\infty(J \times \Omega))^d\), \(\frac{\partial u}{\partial t} \in (L^2(J;H^s(\Omega)))^d \cap (L^\infty(J \times \Omega))^d\)

and \(u_0 \in (H^s(\Omega))^d \cap (L^\infty(\Omega))^d\);
3) \( C_f \in L^{\infty}(J; H^{s+1}(\Omega)) \cap W^{2,\infty}(J \times \Omega), \frac{\partial C_f}{\partial t} \in L^2(J; H^{s}(\Omega)), \ C_{f0} \in H^s(\Omega); \)

where \( s \geq r + 1 \) and \( q \geq 3 \).

For the cut-off operator for velocity defined in (58), we assume that the constant \( M \) is chosen as \( M > 2 \| u \|_{(L^{\infty}(J \times \Omega))^d} \). Under this assumption, we can deduce that \( |u_0^M - u| \leq |u_h - u| \) pointwise. In fact, it obviously holds if \( |u_h| \leq M \), so we only need to consider the case \( |u_h| > M \). In this case, denote \( \kappa = M/|u_h| \), and then \( \kappa < 1 \). By the definition of \( u^M \), we have

\[
|u_0^M - u| = |\kappa u_h - u| \leq \kappa |u_h - u| + (1 - \kappa) |u|.
\]

On the other hand, the assumption tells us \( |u_h| > M > 2 |u| \), and as a result

\[
|u_h - u| \geq |u_h| - |u| \geq |u|.
\]

Consequently, the required result is obtained.

For the continuous-in-time scheme, finite element solution errors are defined as

\[
E_\phi = \phi - \phi_h,
E_p = p - p_h,
E_u = u - u_h,
E_c = C_f - C_h,
E_q = q - q_h.
\]

Furthermore, define the projection errors as

\[
E_p^I = P_h p - p,
E_u^I = \Pi_h u - u,
E_c^I = P_h C_f - C_f,
\]
\( E^l_q = \Pi_h q - q \),

and also define the auxiliary errors as

\[
E^A_p = P_h p - p_h,
\]
\[
E^A_u = \Pi_h u - u_h,
\]
\[
E^A_c = P_h C_f - C_h,
\]
\[
E^A_q = \Pi_h q - q_h.
\]

**Lemma 4.** The approximate error of the discrete porosity satisfies

\[
\| E_\phi \|^2(t) \leq R^*_\phi \int_0^t \left( \| E_\phi \|^2 + \| E_c \|^2 \right),
\]

\[
\int_0^t \left\| \frac{\partial E_\phi}{\partial t} \right\|^2 \leq C \int_0^t \left( \| E_\phi \|^2 + \| E_c \|^2 \right),
\]

where \( t \in (0,T] \).

**Proof.** It is observed that

\[
\frac{\partial E_\phi}{\partial t} = -\frac{R_\phi}{1-\phi_0} E_\phi C_f + \frac{R_\phi}{1-\phi_0} (1-\phi_h)(C_f - \bar{C}_h). \tag{82}
\]

Multiplying (82) by \( E_\phi \), and then integrating it over \( \Omega \), we obtain that

\[
\frac{1}{2} \frac{\partial}{\partial t} \| E_\phi \|^2 = \left( \frac{R_\phi}{1-\phi_0} C_f, E_\phi \right) E_\phi \\
+ \left( \frac{R_\phi}{1-\phi_0} (1-\phi_h)(C_f - \bar{C}_h), E_\phi \right) \leq \frac{R^*_\phi}{2} \left( \| E_\phi \|^2 + \| E_c \|^2 \right), \tag{83}
\]

where we used the fact \(|C_f - \bar{C}_h| \leq |C_f - C_h|\). The inequality (81) is obtained by integrating (83) in time from 0 to \( t \) and taking into account \( \| E_\phi \|(0) = 0 \). On the other hand, it follows from (82) that

\[
\left\| \frac{\partial E_\phi}{\partial t} \right\| \leq \left\| \frac{R_\phi}{1-\phi_0} C_f E_\phi \right\| + \left\| \frac{R_\phi}{1-\phi_0} (1-\phi_h)(C_f - \bar{C}_h) \right\|.
\[
\leq \left\| R_{\phi}^{c_f} \right\|_{L^\infty} \left\| E_\phi \right\| + R_{\phi}^c \left\| E_c \right\|. \tag{84}
\]

Thus, the second estimate is obtained from (84). \(\square\)

**Lemma 5.** The auxiliary errors of discrete pressure \(p_h\) and velocity satisfy

\[
\left\| E_{p}^A \right\|^2(t) + \left\| E_{u}^A \right\|^2(t) + \int_0^t \left\| K(\phi_h)^{-1/2} E_{u}^A \right\|^2 + \int_0^t \left\| \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} \right)^{1/3} E_{u}^A \right\|_{L^2}^3 \\
\leq C \int_0^t \left( \left\| E_{p}^A \right\|^2 + \left\| E_{u}^A \right\|^2 + \left\| E_{c}^A \right\|^2 + \left\| E_\phi \right\|^2 \right) + Ch^{2r+2} \tag{85}
\]

where \(t \in (0, T]\) and the values of different \(C\) depend on \(T\) and the exact solutions, but independent of \(h\).

**Proof.** Firstly, from the MFEM formulations of the pressure and velocity, using the approximation properties, we obtain

\[
\begin{align*}
\varepsilon \left( \frac{\partial E_{p}^A}{\partial t}, \varphi \right) + (\nabla \cdot E_{u}^A, \varphi) &= - \left( \frac{\partial E_{\phi}}{\partial t}, \varphi \right), \\
\left( \frac{\rho}{\phi_h} \frac{\partial E_{u}^A}{\partial t}, \nu \right) + \left( \frac{\mu}{K(\phi_h)} E_{u}^A, \nu \right) &= \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} (|u|_u - |u_h|_u), \nu \right) \\
&= \left( E_{p}^A, \nabla \cdot \nu \right) + \left( \frac{\rho}{\phi_h} \frac{\partial E_{u}^l}{\partial t}, \nu \right) + \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} (|u|_u - |u_h|_u), \nu \right) \\
&\quad + \left( \left( \frac{\mu}{K(\phi_h)} - \frac{\mu}{\phi_h} \right) u, \nu \right) + \left( \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} - \frac{\rho F(\phi)}{\sqrt{K(\phi)}} \right) |u|_u, \nu \right). \tag{87}
\end{align*}
\]

Taking \(\varphi = E_{p}^A\) into (86) and \(\nu = E_{u}^A\) into (87), and then summing these two equations, we get

\[
\begin{align*}
\frac{\varepsilon}{2} \frac{\partial}{\partial t} \left\| E_{p}^A \right\|^2 + \left( \frac{\rho}{\phi_h} \frac{\partial E_{u}^A}{\partial t}, E_{u}^A \right) + \mu \left\| K(\phi_h)^{-1/2} E_{u}^A \right\|^2 + \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} (|u|_u - |u_h|_u), E_{u}^A \right) \\
&= - \left( \frac{\partial E_{\phi}}{\partial t}, E_{p}^A \right) + \left( \frac{\rho F(\phi_h)}{\phi_h \phi_h \phi_h} \frac{\partial E_{u}^l}{\partial t}, E_{u}^A \right) + \left( \frac{\rho}{\phi_h} \frac{\partial E_{u}^l}{\partial t}, E_{u}^A \right) + \left( \frac{\mu}{k(\phi)} E_{u}^l, E_{u}^A \right).
\end{align*}
\]
+ \left( \frac{\mu}{K(\phi_h)} - \frac{\mu}{K(\phi)} \right) u, E_u^A \right) + \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} - \frac{\rho F(\phi)}{\sqrt{K(\phi)}} \right) |u| u, E_u^A \right). 

(88)

Then the second term on the left-hand side of (88) is bounded below as

\begin{align*}
\left( \frac{\rho}{\phi_h} \frac{\partial E_u^A}{\partial t}, E_u^A \right) &= \frac{1}{2} \left( \frac{\partial}{\partial t} \left( \frac{\rho}{\phi_h} E_u^A, E_u^A \right) + \left( \frac{\partial}{\partial t} \frac{\partial \phi_h}{\partial t} E_u^A, E_u^A \right) \right) \\
&\geq \frac{1}{2} \frac{\partial}{\partial t} \left( \frac{\rho}{\phi_h} E_u^A, E_u^A \right),
\end{align*}

because \( \frac{\rho}{\phi_h} \frac{\partial \phi_h}{\partial t} \geq 0 \). The last term on the left-hand side of (88) can be rewritten as

\begin{align*}
\left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} (|u| |u - u_h|), E_u^A \right) &= \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} (|u| - |u_h|) u, E_u^A \right) \\
&+ \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} |u - u_h| (u - u_h), E_u^A \right) + \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} (|u_h| - |u - u_h|) (u - u_h), E_u^A \right) \\
&=: T_1 + T_2 + T_3.
\end{align*}

We recall the elementary lemma

\[ ||w| - |w - v|| \leq |v| \]

For any \( d \)-dimensional vectors \( w \) and \( v \). The terms \( T_1 \) and \( T_3 \) are estimated as

\[ |T_1| + |T_3| \leq 2 \left( \frac{\rho F(\phi_0)}{\sqrt{K_0}} \right) \| u \|_{L^\infty} \| u - u_h \| E_u^A \leq C (\| E_u^A \|^2 + \| E_u^A \|^2). \]

With Young’s inequality, the term \( T_2 \) is estimated as

\begin{align*}
T_2 &= \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} |E_u^A| E_u^A, E_u^A \right) + \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} (|E_u^A| - |E_u^A|) E_u^A, E_u^A \right) \\
&\leq \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} |E_u^A| E_u^A, E_u^A \right) \\
&\leq \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} \right)^{1/3} E_u^A \left( \frac{1}{L^3} \right)^3 - 2 \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} \right)^{1/3} E_u^A \left( \frac{1}{L^3} \right)^3 - 2 \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} \right)^{1/3} E_u^A \left( \frac{1}{L^3} \right)^3
\end{align*}
\[
- \left\| \left( \frac{\partial F(\phi_h)}{\sqrt{K(\phi_h)}} \right)^{1/3} E_u^A \right\|_{L^3}^2 \geq \frac{1}{2} \left\| \left( \frac{\partial F(\phi_h)}{\sqrt{K(\phi_h)}} \right)^{1/3} E_u^A \right\|_{L^3}^2 - C \left\| \frac{\partial F(\phi_0)}{\sqrt{K_0}} \right\|_{L^\infty} \| E_u^A \|_{L^3}^3.
\]

We now turn to estimate the terms on the right-hand side of (88). Using Cauchy-Schwarz inequality, we obtain the estimates

\[
\left( \frac{\partial \phi}{\partial t}, E_p^A \right) \leq \frac{1}{2\varepsilon} \left\| \frac{\partial \phi}{\partial t} \right\|^2 + \frac{1}{2} \| E_p^A \|^2,
\]

\[
\left( \frac{\rho \phi \partial u}{\phi_h \partial t}, E_u^A \right) \leq \frac{1}{2} \left\| \frac{\partial u}{\partial t} \right\|^2_{L^\infty} \| E_u^A \|^2 + \frac{1}{2} \| E_u^A \|^2.
\]

It is derived from the boundedness of \( \phi_h \) and \( K(\phi_h)^{-1} \) that

\[
\left( \frac{\rho \phi \partial E_u^I}{\phi_h \partial t}, E_u^A \right) + \left( \frac{\mu}{K(\phi_h)^2} E_u^I, E_u^A \right) \leq C \left\| \frac{\partial E_u^I}{\partial t} \right\|^2 + C \| E_u^I \|^2 + \| E_u^A \|^2.
\]

Using the uniform Lipschitz continuity of \( K(\cdot)^{-1} \) and \( \frac{F(\cdot)}{\sqrt{K(\cdot)}} \), we get

\[
\left( \left( \frac{\mu}{K(\phi_h)} - \frac{\mu}{K(\phi)} \right) u, E_u^A \right) + \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} - \frac{\rho F(\phi)}{\sqrt{K(\phi)}} \right) u | u, E_u^A \right) \leq C \| E_u^A \|^2 + \| E_u^A \|^2.
\]

Combining the above estimates gives

\[
\frac{\varepsilon}{2} \frac{\partial}{\partial t} \| E_p^A \|^2 + \frac{1}{2} \frac{\partial}{\partial t} \left( \frac{\rho \phi_h E_u^A, E_u^A} {\phi_h} \right) + \mu \| K(\phi_h)^{-1/2} E_u^A \|^2 + \frac{1}{2} \left\| \left( \frac{\rho F(\phi_h)}{\sqrt{K(\phi_h)}} \right)^{1/3} E_u^A \right\|_{L^3}^3 \leq C \left\| \frac{\partial E_u^I}{\partial t} \right\|^2 + C \| E_u^I \|^2 + C \| E_u^I \|^3 + \frac{\varepsilon}{2} \| E_p^A \|^2 + C \| E_u^A \|^2
\]

\[
+ C \left\| \frac{\partial E_u^I}{\partial t} \right\|^2 + C \| E_u^A \|^2.
\]

(89)
Integrating (89) in time from 0 to $t$ ($t \in J$) and using Lemma 4 and the approximation results, we reach the estimate (85). □

**Lemma 6.** The auxiliary errors of the discrete concentration and auxiliary flux variable satisfy

$$\|E_c^A\|^2(t) + \int_0^t \|(\phi_h D_e(u_h^M))^{-1/2}E_q^A\|^2 \leq C \int_0^t \left(\|E_c^A\|^2 + \|E_q^A\|^2 + \|E_\phi\|^2 \right) + Ch^{2r+2}, \quad (90)$$

where $t \in (0,T]$ and the values of different $C$ depend on $T$ and the exact solutions, but independent of $h$.

**Proof.** Firstly, it is easy to know that for the solute transport equation,

$$\left(\frac{\partial (\phi_h E_c^A)}{\partial t}, \psi \right) + (\nabla \cdot E_q^A, \psi) + \frac{k_c k_s}{k_c + k_s} (a_v(\phi_h) E_c^A, \psi)$$

$$= - \left(\frac{\partial (E_\phi C_f)}{\partial t}, \psi \right) + \frac{k_c k_s}{k_c + k_s} (a_v(\phi_h) E_c^A, \psi)$$

$$+ \frac{k_s}{k_c + k_s} \left( (a_v(\phi_h) - a_v(\phi)) C_f, \psi \right) + (f_p E_c^A, \psi), \quad (91)$$

$$((\phi_h D_e(u_h^M))^{-1} E_q^A, w) = (E_c^A, \nabla \cdot w) + (\phi_h^{-1} M_e(u_h) E_c^A, w)$$

$$+ ((\phi_h D_e(u_h^M))^{-1} E_q^A, w) + ((\phi_h D_e(u_h^M))^{-1} - (\phi D_e(u))^{-1} q, w)$$

$$+ \left( (\phi^{-1} M_e(u) - \phi_h^{-1} M_e(u_h)) C_f, w \right). \quad (92)$$

Take $\psi = E_c^A$ in (91) and $w = E_q^A$ in (92), respectively, and then sum these two equations

$$\left(\frac{\partial (\phi_h E_c^A)}{\partial t}, E_c^A \right) + \|(\phi_h D_e(u_h^M))^{-1/2}E_q^A\|^2 \leq \frac{k_c k_s}{k_c + k_s} (a_v(\phi_h) E_c^A, E_c^A)$$
\[
= \left( \frac{\partial (\phi_h E^I_c)}{\partial t}, E^A_c \right) - \left( \frac{\partial (E^A_c C_f)}{\partial t}, E^A_c \right) + \frac{k_c k_s}{k_c + k_s} (a_v(\phi_h) E^I_c, E^A_c) \\
+ \frac{k_c k_s}{k_c + k_s} \left( (a_v(\phi_h) - a_v(\phi)) C_f, E^A_c \right) + \left( \phi_h^{-1} M_e(u_h) E_c, E^A_q \right) \\
+ \left( (\phi_h D_e(u_h^M))^{-1} E^I_q, E^A_q \right) + \left( ((\phi_h D_e(u_h^M))^{-1} - (\phi D_e(u))^{-1}) q, E^A_q \right) \\
+ \left( (\phi_1^{-1} M_e(u) - \phi_h^{-1} M_e(u_h) \right) C_f, E^A_q) + (f_p E_c, E^A_c). \tag{93}
\]

Because \( \frac{\partial \phi_h}{\partial t} \geq 0 \), the first term on the left-hand side of (93) is estimated as
\[
\left( \frac{\partial (\phi_h E^A_c)}{\partial t}, E^A_c \right) = \frac{1}{2} \left( \frac{\partial}{\partial t} \phi_h E^A_c, E^A_c \right) + \left( \frac{\partial \phi_h}{\partial t} E^I_c, E^A_c \right) \geq \frac{1}{2} \frac{\partial}{\partial t} (\phi_h E^A_c, E^A_c).
\]

Because \( k_c, k_s \) and \( a_v(\phi_h) \) are all nonnegative, the third term on the left-hand side of (93) is also nonnegative. The first term on the right-hand side of (93) is estimated as
\[
\left( \frac{\partial (\phi_h E^I_c)}{\partial t}, E^A_c \right) = \left( \frac{\partial \phi_h}{\partial t} E^I_c, E^A_c \right) + (\phi_h E^I_c, E^A_c) \\
\leq \|E^A_c\|^2 + \frac{1}{2} R^* \|E^I_c\|^2 + \frac{1}{2} \left\| \frac{\partial E^I_c}{\partial t} \right\|^2,
\]

where we have used the boundedness of \( \phi_h \) and \( \frac{\partial \phi_h}{\partial t} \). The second term on the right-hand side of (93) is estimated as
\[
- \left( \frac{\partial (E^A_c C_f)}{\partial t}, E^A_c \right) = - \left( C_f \frac{\partial E^A_c}{\partial t}, E^A_c \right) - \left( E^A_c \frac{\partial C_f}{\partial t}, E^A_c \right) \\
\leq \|E^A_c\|^2 + \frac{1}{2} \left\| \frac{\partial C_f}{\partial t} \right\|^2 \|E^A_c\|^2 + \frac{1}{2} \left\| C_f \right\|^2 \left\| \frac{\partial E^A_c}{\partial t} \right\|^2.
\]

Noticing that \( 0 \leq a_v(\phi_h) \leq a_0 \) and \( |a_v(\phi_h) - a_v(\phi)| \leq \frac{a_0}{1 - \phi_0} \|E^A_\phi\| \), we obtain the estimates
\[
\frac{k_c k_s}{k_c + k_s} (a_v(\phi_h) E^I_c, E^A_c) \leq \frac{1}{2} \|E^A_c\|^2 + \frac{1}{2} \left( \frac{k_c k_s a_0}{k_c + k_s} \right)^2 \|E^I_c\|^2,
\]
\[
\frac{k_c k_s}{k_c + k_s} (a_v(\phi_h) - a_v(\phi)) C_f, E_c^A \leq \frac{1}{2} \|E_c^A\|^2
\]
\[
+ \frac{1}{2} \frac{(k_c k_s a_0^*)^2}{(k_c + k_s)^2(1 - \phi_0^*)^2} \|C_f\|_{L^\infty}^2 \|E_{\phi}\|^2.
\]

Using the boundedness of \( M_e(u_h) \) and \( D_e(u_h^M) \), we get
\[
(\phi_h^{-1} M_e(u_h) E_c, E_q^A) + ((\phi_h D_e(u_h^M))^{-1} E_q^I, E_q^A)
\]
\[
\leq \frac{1}{4} \left\| \left(\phi_h D_e(u_h^M)\right)^{-\frac{1}{2}} E_q^A \right\|^2 + C \|E_c\|^2 + C \|E_q^I\|^2.
\]

From the hypothesis regarding the choice of \( M \), we have \(|u_h^M - u| \leq |u_h - u|\) pointwise, so using the Lipschitz continuity of \( M_e(\cdot) \) and \( D_e(\cdot) \), we get
\[
\left( ((\phi_h D_e(u_h^M))^{-1} - (\phi D_e(u))^{-1}) q, E_q^A \right) + \left( (\phi^{-1} M_e(u) - \phi_h^{-1} M_e(u_h)) C_f, E_q^A \right)
\]
\[
\leq \frac{1}{4} \left\| \left(\phi_h D_e(u_h^M)\right)^{-\frac{1}{2}} E_q^A \right\|^2 + C \left( \|C_f\|_{L^\infty}^2 + \|q\|_{L^\infty}^2 \right) \left( \|E_u\|^2 + \|E_{\phi}\|^2 \right).
\]

Because \( f_p \leq 0 \), the last term is estimated as
\[
(f_p E_c, E_c^A) = (f_p E_c^A, E_c^A) - (f_p E_c^I, E_c^A)
\]
\[
\leq -(f_p E_c^I, E_c^A) \leq \frac{1}{2} \left( \|f_p\|_{L^\infty}^2 \|E_c^I\|^2 + \|E_c^A\|^2 \right).
\]

We summarize the above estimates and get
\[
\frac{1}{2} \frac{\partial}{\partial t} (\phi_h E_c^A, E_c^A) + \frac{1}{2} \left\| (\phi_h D_e(u_h^M))^{-1/2} E_q^A \right\|^2 \leq C \|E_c^A\|^2 + C \|E_c^I\|^2
\]
\[
+ \frac{1}{2} \left\| \frac{\partial E_c^I}{\partial t} \right\|^2 + C \|E_q^I\|^2 + C \|E_u\|^2 + C \|E_{\phi}\|^2 + C \left\| \frac{\partial E_{\phi}}{\partial t} \right\|^2. \tag{94}
\]

Finally, the estimate (90) is obtained by integrating (94) in time from 0 to \( t (t \in J) \), using Lemma 4 and the approximation results. □

Based on the above lemmas, we are now able to prove the main error estimates as below.
Theorem 3. The MFEM solutions of (59) - (63) satisfy the following error estimate
\[
\|E\phi\|_{L^\infty(J;L^2)}^2 + \|E_p\|_{L^\infty(J;L^2)}^2 + \|E_u\|_{L^\infty(J;L^2)}^2 + \|E_c\|_{L^\infty(J;L^2)}^2 \\
+ \left\| \frac{\partial E\phi}{\partial t} \right\|_{L^2(J;L^2)}^2 + \left\| K(\phi_h)^{-1/2} E_u \right\|_{L^2(J;L^2)}^2 + \left\| (\rho F(\phi_h))^{1/3} E_u \right\|_{L^3(J;L^3)}^3 \\
+ \left\| (\phi_h D_e(u_h^M))^{-1/2} E_q \right\|_{L^2(J;L^2)}^2 \leq C h^{2r+2},
\]
where \( J = (0, T) \) and the value of \( C \) depends on \( T \) and the exact solutions, but independent of \( h \).

Proof. It is followed from Lemmas 4 – 6 that
\[
\|E\phi\|_t^2 + \|E_p^A\|_t^2 + \|E_u^A\|_t^2 + \|E_c^A\|_t^2 + \int_0^t \left\| \frac{\partial E\phi}{\partial t} \right\|_t^2 \\
+ \int_0^t \left| K(\phi_h)^{-1/2} E_u^A \right|_t^2 + \int_0^t \left| (\rho F(\phi_h))^{1/3} E_u^A \right|_t^3 + \int_0^t \left\| (\phi_h D_e(u_h^M))^{-1/2} E_q^A \right\|_t^2 \\
\leq C \int_0^t \left( \|E\phi\|_t^2 + \|E_p^A\|_t^2 + \|E_u^A\|_t^2 + \|E_c^A\|_t^2 \right) + Ch^{2r+2},
\]
where \( t \in (0, T] \), and the values of different \( C \) depend on \( T \) and the exact solutions, but are independent of \( h \). Thus, the estimate (95) is obtained by applying Gronwall’s lemma to (96) and then using the triangle rule. \( \square \)

In many practical applications, the recovery rate, denoted by \( \theta \), is an important indicator in oil production, which is defined as
\[
\theta = 1 - \frac{\int_{\Omega} C_f(x,t) dx}{\int_{\Omega} C_f^0 dx},
\]
and the discrete recovery rate \( \theta_h \) as
\[
\theta_h = 1 - \frac{\int_{\Omega} C_h(x,t) dx}{\int_{\Omega} C_f^0 dx}.
\]
From Theorem 3, we immediately deduce the following approximate error of the discrete recovery rate.

**Corollary 1.** The error between the exact and discrete recovery rates is bounded as

$$|\theta_h - \theta| \leq C h^v,$$

where $v = r + 1$ and $C$ is independent of $h$.

### 7.5 Upwind Quadrature Rule and Time Discretization Scheme

In the numerical implement of the finite element methods, the quadrature rule is usually necessary for the integral calculations, but the convenient quadrature rule lacks upwind mechanism to characterize the convection term. Notice that the functions in $\mathcal{W}$ may be discontinuous, we introduce the following upwind quadrature rule for $C_h$ as

$$\left( \phi_h^{-1} M_e (u_h) C_h, w \right) \approx \sum_{i=1}^{Q_d} \omega_i \phi_h^{-1} (x_i) M_e (u_h (x_i)) \cdot w(x_i) C_h^*(x_i),$$

where $E$ is an element, $Q_d$ is the number of quadrature points and $\omega_i$ are the quadrature weights. If the point $x_i \in e$, where $e \in \Gamma_h$ is shared by the two elements $E^1_e$ and $E^2_e$ with the unit normal vector $n_e$ exterior to $E^1_e$, the upwind value of $C_h^*$ at $x_i$ is defined as

$$C_h^*(x_i) = \begin{cases} C_h(x_i)|_{E^1_e}, & \text{if } u_h(x_i) \cdot n_e \geq 0, \\ C_h(x_i)|_{E^2_e}, & \text{if } u_h(x_i) \cdot n_e < 0. \end{cases}$$

In numerical tests, we employ the lowest order Raviart-Thomas space ($RT_0$) on a rectangular mesh. Furthermore, we apply the upwind trapezoid quadrature rule to

$$\left( \phi_h^{-1} M_e (u_h) C_h, w \right)$$

and apply the conventional trapezoid quadrature rule to
\((\phi_h D_e (u^N_h))^{-1} q_h, w\), respectively for decoupling the system and getting explicit formulas for each individual diffusive flux [143].

We now present the time discretization of the mixed finite element forms used in the numerical experiment. Let the total time interval \( J = (0, T] \), where \( T > 0 \). Divide \( J \) into \( N \) subintervals \( I_n = (t_n, t_{n+1}] \), where \( t_0 = 0 \) and \( t_N = T \), and denote \( \delta t_n = t_{n+1} - t_n \).

For any scalar \( v(t) \) or vector \( u(t) \), we denote by \( v^n \) or \( u^n \) its approximation at the time \( t_n \).

Suppose that \( p^n_h, c^n_h \in \mathcal{W} \) and \( u^n_h \in \mathcal{X}_0 \), \( q^n_h \in \mathcal{Y}_0(0 \leq n \leq N - 1) \) are given or calculated, the implicit-explicit MFEM formulation is stated as below: find \( p^{n+1}_h, c^{n+1}_h \in \mathcal{W} \) and \( u^{n+1}_h \in \mathcal{X}_0 \), \( q^{n+1}_h \in \mathcal{Y}_0 \) such that

\[
\varepsilon \left( \frac{p^{n+1}_h - p^n_h}{\delta t_n}, \phi \right) + \left( \frac{\phi^{n+1}_h - \phi^n_h}{\delta t_n}, \phi \right) + (\nabla \cdot u^{n+1}_h, \phi) = (f(x, t^{n+1}), \phi), \phi \in \mathcal{W},
\]

\[
\left( \frac{\rho}{\phi^{n+1}_h} \frac{u^{n+1}_h - u^n_h}{\delta t_n}, v \right) + \left( \frac{\mu}{K(\phi^{n+1}_h)} u^{n+1}_h, v \right) + \left( \rho F(\phi^{n+1}_h) \frac{|u^n_h| u^{n+1}_h}{\sqrt{K(\phi^{n+1}_h)}}, v \right)
\]

\[
= (p^{n+1}_h, \nabla \cdot v) + (\rho g, v), \quad v \in \mathcal{X}_0,
\]

\[
\left( \frac{\phi^{n+1}_h c^{n+1}_h - \phi^n_c c^n_h}{\delta t_n}, \psi \right) + (\nabla \cdot q^{n+1}_h, \psi) + \frac{k_c k_s}{k_c + k_s} (a_\nu(\phi^{n+1}_h) c^{n+1}_h, \psi)
\]

\[-(f_p(x, t^{n+1}) c^{n+1}_h, \psi) = (f_i(x, t^{n+1}) c_i(x, t^{n+1}), \psi), \quad \psi \in \mathcal{W},
\]

\[
((\phi^{n+1}_h D_e (u^{n+1,M}_h))^{-1} q^{n+1}_h, w)
\]

\[
= ((\phi^{n+1}_h)^{-1} \mathcal{M}_e (u^{n+1}_h) c^{n+1}_h, w) + (c^{n+1}_h, \nabla \cdot w), \quad w \in \mathcal{Y}_0,
\]

where \( u^{n+1,M}_h = \mathcal{M}(u^{n+1}_h) \).

For the calculation of the discrete porosity, we use an implicit-explicit scheme as
\[
\frac{\phi_h^{n+1} - \phi_h^n}{\delta t_n} = \frac{R\phi(1-\phi_h^{n+1})}{1-\phi_0} \tilde{C}_h^n,
\] (97)

where \( \tilde{C}_h^n = \max(0, \min(C_h^n, 1)) \). Denote

\[
\theta^n = \delta t_n \frac{R\phi}{1-\phi_0} \tilde{C}_h^n.
\]

From (97), the porosity \( \phi_h^{n+1} \) can be calculated as

\[
\phi_h^{n+1} = \frac{\theta^n}{1+\theta^n} + \frac{\phi_h^n}{1+\theta^n}.
\] (98)

Note that the porosity \( \phi_h^{n+1} \) computed by this method satisfies the maximum principle for any time step size. Indeed, suppose that \( 0 \leq \phi_h^n \leq 1 \), then we deduce from (98) that \( 0 < \phi_h^n \leq \phi_h^{n+1} < 1 \) holds for any time step size. By recurrence relation, we obtain \( 0 < \phi_0 \leq \phi_h^n < 1 \) for \( n \geq 1 \).

7.6 Numerical Error Analysis

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viscosity of the acid flow (( \mu ))</td>
<td>( 10^{-2} ) Pa • s</td>
</tr>
<tr>
<td>Density of the acid flow (( \rho ))</td>
<td>( 10^3 ) kg/m(^2)</td>
</tr>
<tr>
<td>Longitudinal dispersion coefficient ( (D_L) )</td>
<td>( 10^{-7} )</td>
</tr>
<tr>
<td>Transverse dispersion coefficient ( (D_T) )</td>
<td>( 10^{-9} )</td>
</tr>
<tr>
<td>Molecular diffusion coefficient ( (D_M) )</td>
<td>( 10^{-7} )</td>
</tr>
<tr>
<td>Surface reaction rate constant ( (k_r) )</td>
<td>( 10^{-4} ) m/s</td>
</tr>
<tr>
<td>Local mass-transfer coefficient ( (k_l) )</td>
<td>( 10^{-4} ) m/s</td>
</tr>
<tr>
<td>Dissolving power of acid (( \alpha ))</td>
<td>0.1 kg/mole</td>
</tr>
<tr>
<td>Density of the rock (( \rho_s ))</td>
<td>2500 kg/m(^2)</td>
</tr>
<tr>
<td>Initial interfacial area ( (a_0) )</td>
<td>0.5 m(^2)</td>
</tr>
<tr>
<td>Initial concentration of the acid flow ( (C_f^0) )</td>
<td>( 10^3 ) mole/m(^2)</td>
</tr>
</tbody>
</table>

In this example, we test a simple problem to verify the convergence rate of the propose scheme. Suppose there is a 0.2-meter by 0.2-meter porous medium. The initial porosity of the porous medium is 0.2 everywhere and the initial permeability of the porous medium is \( 10^{-12} \) m\(^2\) everywhere. A type of acid flow is injected from the left boundary and the
velocity is $10^{-6}$ m/s. A constant pressure of $10^5$ Pa is imposed on the right boundary of the porous medium. The upper and lower boundaries of the porous medium are closed. The initial pressure inside the porous medium is also $10^5$ Pa. Two high-porosity entries are set on the left boundary of the porous medium, one 0.05 meter above the bottom with a porosity of 0.6, and the other 0.1 meter above the bottom with a porosity of 0.5. The permeabilities of the two entries are $10^{-10}$ m$^2$ and $10^{-11}$ m$^2$ respectively. The other parameters can be observed in Table 12. The parameter $\varepsilon$ is set to 0. Each time step is 100 seconds. There are four sizes of grid in the experiment and they have 20$^2$, 40$^2$, 80$^2$ and 160$^2$ cells, respectively. The errors of physical parameters such as the porosity, concentration, pressure and velocity in the grids of 20$^2$, 40$^2$, 80$^2$ cells are calculated by comparing them with the corresponding values in the grid of 160$^2$ cells. The data of fine grid of 160$^2$ cells is linearly interpolated to the coarse grid. We calculate the errors at the end of the 2500th second, 5000th second, 7500th second and 10000th second, respectively. The listed results in Table 13 - Table 16 are the relative errors which can be calculated as

$$\|E_\phi\|_{rel} = \left\| \frac{E_\phi}{\|\phi_h\|} \right\|,$$

$$\|E_p\|_{rel} = \left\| \frac{E_p}{\|p_h\|} \right\|,$$

$$\|E_c\|_{rel} = \left\| \frac{E_c}{\|C_h\|} \right\|,$$

$$\|E_u\|_{rel} = \left\| \frac{E_u}{\|u_h\|} \right\|.$$
It is observed from the present numerical error results that the convergence rate of all MFEM approximations is almost first order and the porosity convergence depends on the concentration. This is in agreement with the theoretical analysis results.

Table 13 Numerical errors of porosities in different rectangular meshes at different time steps.

<table>
<thead>
<tr>
<th>Porosity</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>20*20</td>
<td>0.046493</td>
<td>0.06579</td>
<td>0.066958</td>
<td>0.065767</td>
</tr>
<tr>
<td>40*40</td>
<td>0.029612</td>
<td>0.03473</td>
<td>0.032795</td>
<td>0.032507</td>
</tr>
<tr>
<td>80*80</td>
<td>0.011837</td>
<td>0.012807</td>
<td>0.012462</td>
<td>0.013261</td>
</tr>
</tbody>
</table>

Table 14 Numerical errors of concentrations in different rectangular meshes at different time steps.

<table>
<thead>
<tr>
<th>Concentration</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>20*20</td>
<td>0.26962</td>
<td>0.22998</td>
<td>0.21524</td>
<td>0.20962</td>
</tr>
<tr>
<td>40*40</td>
<td>0.15376</td>
<td>0.11824</td>
<td>0.10478</td>
<td>0.10284</td>
</tr>
<tr>
<td>80*80</td>
<td>0.057591</td>
<td>0.041013</td>
<td>0.036898</td>
<td>0.03921</td>
</tr>
</tbody>
</table>

Table 15 Numerical errors of pressures in different rectangular meshes at different time steps.

<table>
<thead>
<tr>
<th>Pressure</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>20*20</td>
<td>0.00038845</td>
<td>0.0010679</td>
<td>0.0015855</td>
<td>0.001986</td>
</tr>
<tr>
<td>40*40</td>
<td>0.00020249</td>
<td>0.00048864</td>
<td>0.00066225</td>
<td>0.0008296</td>
</tr>
<tr>
<td>80*80</td>
<td>6.0701e-05</td>
<td>0.00014087</td>
<td>0.00019473</td>
<td>0.00026732</td>
</tr>
</tbody>
</table>

Table 16 Numerical errors of velocities in different rectangular meshes at different time steps.

<table>
<thead>
<tr>
<th>Velocity</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>20*20</td>
<td>0.4557</td>
<td>0.6968</td>
<td>0.7424</td>
<td>0.7506</td>
</tr>
<tr>
<td>40*40</td>
<td>0.3077</td>
<td>0.3876</td>
<td>0.4488</td>
<td>0.5181</td>
</tr>
<tr>
<td>80*80</td>
<td>0.1517</td>
<td>0.2000</td>
<td>0.2489</td>
<td>0.3110</td>
</tr>
</tbody>
</table>
8. Conclusions

In this work, a parallel two-phase compositional flow code is developed. The parallel code achieves satisfactory scalability, and a multilevel solver in Hypre performs well. Then, a sparse grid surrogate is applied to the flash calculation in the two-phase compositional flow simulation. A series of experiments are conducted to verify the accuracy and effectiveness of this approach. In the small case study, the applicability of two surrogate models is checked. One is based on Cartesian grids, and the other one is based on sparse grids. Both give acceptable results for a two-component test case, with the sparse grid surrogate performing only slightly worse than the full grid surrogate. But sparse grid surrogate has a drastically reduced computing effort with only about 10% of the memory requirements of full grid surrogate, for the most unfavorable test setup.

When applying the approach to an example larger problem, we find that after the application of the surrogates, the computing speed of the flash calculation is increased by more than 1600 times, followed by the speedup of the whole two-phase compositional flow simulation by more than 10 times for the best case. This demonstrates that the bottleneck of flash calculations has been removed by the application of sparse grids. To fully exploit the advantages of sparse grids, applying the proposed method for more than two components will be part of future work.

In the wormhole simulation, 2D and 3D wormholes are simulated with the Darcy and DBF frameworks. In the experiments, by comparing the results from the two frameworks,
we conclude that the DBF framework simulates ramified wormholes, while the Darcy framework simulates facial dissolution. The parallel code achieves good scalability when tested on Shaheen.

An MFEM scheme is also proposed to simulate the wormhole propagation. MFEM is used not only for the DF equation, but also for the solute transport equation by introducing an auxiliary flux variable. Based on the cut-off operators and an auxiliary function of velocity, we prove the stability and a priori error estimates for the velocity, pressure, concentration and porosity in different norms by using the coupled analysis approach.

In the future, more properties simulated by the two frameworks such as the optimal injection rate and the breakthrough time will be compared. Moreover, the iterative solvers in Hypre cannot solve some of the linear systems in the DBF framework, which impedes the application of the parallel code. Thus, a new solver should be developed to solve the problem. Finally, the proposed MFEM scheme will be expanded to the DBF equation.
Appendices

A. Selected Publications and Software


Y. Wu, S. Sun, A. Salama, Parallelization of Two-phase Compositional Flow Simulations: Combination of Sparse Grids and State of the Art Solvers, minor revision required by *Computational Geosciences*.


Reservoir Simulation Toolbox: a piece of parallel software for the 3D two-phase compositional flow simulation. (Independently developed by myself)

Wormhole Simulator: a piece of parallel software for the 3D wormhole simulation. (Independently developed by myself)

B. Finite Difference Method

In the 2D domain, suppose the pressure function is \( p(x_1, x_2, t) \) with \((x_1, x_2)\) being the coordinate of the point and \( t \) being the time. The first derivative of \( p \) with respect to \( x_1 \) can be derived in the following three methods. The first method is the forward finite difference method

\[
\frac{\partial p}{\partial x_1}(x_1, x_2, t) = \frac{p(x_1 + h_1, x_2, t) - p(x_1, x_2, t)}{h_1} - \frac{\partial^2 p}{\partial x_1^2}(x_1^*, x_2, t) \frac{h_1}{2},
\]

where \( x_1 \leq x_1^* \leq x_1 + h_1 \) and \( h_1 > 0 \) are fixed numbers. The second method is the backward finite difference method

\[
\frac{\partial p}{\partial x_1}(x_1, x_2, t) = \frac{p(x_1, x_2, t) - p(x_1 - h_1, x_2, t)}{h_1} - \frac{\partial^2 p}{\partial x_1^2}(x_1^{**}, x_2, t) \frac{h_1}{2},
\]

where \( x_1 - h_1 \leq x_1^{**} \leq x_1 \) is also a fixed number. The third method is CCFD method

\[
\frac{\partial p}{\partial x_1}(x_1, x_2, t) = \frac{p(x_1 + h_1, x_2, t) - p(x_1, x_2, t)}{2h_1} - \left( \frac{\partial^3 p}{\partial x_1^3}(x_1^*, x_2, t) + \frac{\partial^3 p}{\partial x_1^3}(x_1^{**}, x_2, t) \right) \frac{h_1^2}{12}.
\]

All the three expressions above can be applied on \( x_2 \) similarly. When using the difference of \( p \) to estimate the differential of \( p \) with respect to \( x_1 \) or \( x_2 \), the method
has the error of the second order in \( h_1 \), which has the finest accuracy in all the three methods. Thus, in the discretization of this work, CCFD method is used.

C. Procedure to Call Structured-Grid System Interfaces of Hypre

There are four steps to set up the linear system by calling the structured-grid system interface of Hypre: setting up the grid, the stencil, the matrix and the right-hand side vector. The basic component of a grid is a box, which is a collection of cells owned by the processors. Each processor has only one box. According to the domain decomposition strategy shown in Figure 3(b), each box is a rectangle. To locate the box in the grid, the indices of the left down cell and right up cell of the boxes have to be set. The index is a 2-tuple with the first number representing the \( x \)-direction global index and the second number representing the \( y \)-direction global index. In the experiment, the index begins with one. For example, for Processor 3 in Figure 3(b), the left down cell index is (21,21), and the right up cell index is (40,40). There are some routines to set up the struct grid. The HYPRE_StructGridCreate() routine creates an empty 2D grid object that lives on the MPI_COMM_WORLD communicator. The HYPRE_StructGridSetExtents() routine adds a new box to the grid. The HYPRE_StructGridAssemble() routine is a collective call and finalizes the grid assembly, making the grid ready to be used by the solver. Figure 55 is a FORTRAN90 code snippet to demonstrate the grid setting. In the code snippet, “grid” stands for the grid, “mpi_comm” stands for the communicator, “ierr” is the error code,
and “ilower” and “iupper” stand for the left down index and right up index of the box, respectively.

```fortran
integer(kind=8) :: grid
call HYPRE_StructGridCreate(mpi_comm, 2, grid, ierr)
call HYPRE_StructGridSetExtents(grid, ilower, iupper, ierr)
call HYPRE_StructGridAssemble(grid, ierr)
```

Figure 55 FORTRAN90 code snippet to set up the grid.

Furthermore, a five-point stencil is used to describe the relative offset from a cell in the grid. The five stencil entries are numbered from 0 to 4. The 2-tuples of (0,0), (-1,0), (1,0), (0,-1) and (0,1) are assigned to them, respectively. Thus, the 0th entry stands for the coefficient of the central pressure, the 1st entry stands for the coefficient of the left pressure among others. The `HYPRE_StructStencilCreate()` routine creates an empty 2D, five-point stencil object. The `HYPRE_StructStencilSetElement()` routine defines the geometry of the stencil and assigns the stencil numbers for each of the stencil entries. Neither of the calls are collective calls. Figure 56 is the code snippet to set up the stencil.

In the code snippet, the variable “myentry” stands for the entry, the array “offsets” stands for the five offset indices, and “stencil_indices” is the five-point stencil.

```fortran
integer :: myentry
integer :: offsets(5,2) = reshape([0,-1,1,0,0,0,0,0,-1,1], [5,2])
integer :: stencil_indices(5) = (/0, 1, 2, 3, 4/)
call HYPRE_StructStencilCreate(2, 5, stencil, ierr)
do myentry = 0, 4
   call HYPRE_StructStencilSetElement(stencil, myentry, offsets(myentry+1,1:2), ierr)
end do
```

Figure 56 FORTRAN90 code snippet to set up the stencil.
integer(kind=8) :: A
call HYPRE_StructMatrixCreate(mpi_comm, grid, stencil, A, ierr)
call HYPRE_StructMatrixInitialize(A, ierr)
! The code to fill in the entries of A, omitted here.
call HYPRE_StructMatrixSetBoxValues(A, ilower, iupper, 5, stencil_indices, values, ierr)
call HYPRE_StructMatrixAssemble(A, ierr)

Figure 57 FORTRAN90 code snippet to set up the matrix.

In the step of setting up the matrix, it is necessary to set up the coefficients of all the stencils. For the pressure equation imposed on each cell, there are five coefficients if the cell lies inside the grid and three or four coefficients if the cell lies on the border of the grid. The matrix $A$ in the linear system is set up in terms of the grid and stencil objects described before. The coefficients associated with each stencil entry vary from grid point to grid point. The HYPRE_StructMatrixCreate() routine creates an empty matrix object. The HYPRE_StructMatrixInitialize() routine indicates that the matrix coefficients are ready to be set. The HYPRE_StructMatrixSetBoxValues() routine sets the matrix coefficients of one processor at one time. The coefficients are all stored in an array in the order of rows. In each row, the coefficients are stored according to the entry order in the stencil. The HYPRE_StructMatrixAssemble() routine is a collective call and finalizes the matrix assembly, making the matrix ready to be used. The code snippet to set up the matrix can be observed in Figure 57. In the code snippet, “A” stands for the matrix, and “values” is the array to store the coefficient values.

Finally, all the other terms except the pressure terms are shifted to the right-hand side of the equations to be the right-hand side vector. The right-hand side vector $b$ in the linear
system is set up in the way similarly to the matrix setting up step. The main difference is that there is no stencil here. The HYPRE_StructVectorCreate() routine creates an empty vector object. The HYPRE_StructVectorInitialize() routine indicates that the vector coefficients are ready to be set. This routine follows the same rules as its corresponding matrix routine. The HYPRE_StructVectorSetBoxValues() routine sets the vector coefficients over the grid points in the box, and again, follows the same rules as its corresponding matrix routine. The HYPRE_StructVectorAssemble() routine is a collective call and finalizes the vector assembly, making the vector ready to be used by the solver. Figure 58 shows the code snippet to set up the right-hand side vector. “b” stands for the right-hand side vector, and “rhs_values” is the array to store the right-hand side vector values.

```fortran
integer(kind=8) :: b
call HYPRE_StructVectorCreate(mpi_comm, grid, b, ierr)
call HYPRE_StructVectorInitialize(b, ierr)
! The code to fill in the entries of b, omitted here.
call HYPRE_StructVectorSetBoxValues(b, ilower, iupper, rhs_values, ierr)
call HYPRE_StructVectorAssemble(b, ierr)
```

Figure 58 FORTRAN90 code snippet to set up the right-hand side vector.

### D. Derivation of Degrees of Freedom and Accuracy of Sparse Grids

**Lemma 7.** For purposes of summation over all grid points \( x_{l,i} \) with corresponding basis functions \( \phi_{l,i} \notin V_n^{(1)} \), we obtain for arbitrary \( s \in \mathbb{N} \),

\[
\sum_{|l|_1 > n+d-1} 2^{-s|l|_1} = 2^{-s_n} \cdot 2^{-s_d} \sum_{i=0}^{\infty} 2^{-s_i} \cdot \binom{n+i+d-1}{d-1}.
\]
\[ \leq 2^{-s} n \cdot 2^{-s} d \cdot 2 \cdot A(d, n). \]

**Proof.** We have

\[
\sum_{|\mathbf{l}|_1 > n + d - 1} 2^{-s|\mathbf{l}|_1} = \sum_{i=n+d}^{\infty} 2^{-s} \cdot \sum_{|\mathbf{l}|_1 = i}^{\infty} \sum_{i=n+d}^{\infty} 2^{-s} i \cdot \left( \begin{array}{c} i-1 \\ d-1 \end{array} \right)
\]

\[
= 2^{-s} n \cdot 2^{-s} d \cdot \sum_{i=0}^{\infty} 2^{-s} i \cdot \left( \begin{array}{c} n + i + d - 1 \\ d - 1 \end{array} \right).
\]

Because

\[
\sum_{i=0}^{\infty} x^i \cdot \left( \begin{array}{c} n + i + d - 1 \\ d - 1 \end{array} \right) = \frac{x^{-n}}{(d-1)!} \cdot \left( \sum_{i=0}^{\infty} x^{n+i+d-1} \right)^{(d-1)}
\]

\[
= \frac{x^{-n}}{(d-1)!} \cdot \left( x^{n+d-1} \cdot \frac{1}{1-x} \right)^{(d-1)}
\]

\[
= \frac{x^{-n}}{(d-1)!} \cdot \sum_{k=0}^{d-1} \left( \begin{array}{c} d - 1 \\ k \end{array} \right) \cdot \frac{1}{x^{n+d-1}} \cdot \left( \frac{1}{1-x} \right)^{(d-1-k)}
\]

\[
= \sum_{k=0}^{d-1} \left( \begin{array}{c} n + d - 1 \\ k \end{array} \right) \cdot \frac{x^{d-1-k}}{(1-x)^{d-1-k}} \cdot \frac{1}{1-x}.
\]

We get with \( x = 2^{-s} \),

\[
\sum_{i=0}^{\infty} 2^{-s} i \cdot \left( \begin{array}{c} n + i + d - 1 \\ d - 1 \end{array} \right) \leq 2 \cdot \sum_{k=0}^{d-1} \left( \begin{array}{c} n + d - 1 \\ k \end{array} \right) = 2 \cdot A(d, n).
\]

Thus,

\[
\sum_{|\mathbf{l}|_1 > n + d - 1} 2^{-s|\mathbf{l}|_1} = 2^{-s} n \cdot 2^{-s} d \cdot \sum_{i=0}^{\infty} 2^{-s} i \cdot \left( \begin{array}{c} n + i + d - 1 \\ d - 1 \end{array} \right)
\]

\[
\leq 2^{-s} n \cdot 2^{-s} d \cdot 2 \cdot A(d, n).
\]

Firstly we have to define the discrete approximation spaces and sets of basis functions that span these discrete spaces. Suppose we have the multi-index \( \mathbf{l} = (l_1, \ldots, l_d) \in N^d \)
which indicates the level in a multivariate sense, we consider the family of \( d \)-dimensional standard rectangular grids
\[
\{ \Omega_l : l \in N^d \},
\]
with mesh size
\[
h_l = (h_{l1}, ..., h_{ld}) = 2^{-l}.
\]
That is, the grid \( \Omega_l \) is equidistant with respect to each individual coordinate direction, but in general may have different mesh sizes in the different coordinate directions. The grid points \( x_{l,i} \) of grid \( \Omega_l \) are just the points
\[
x_{l,i} = (x_{l1,i1}, ..., x_{ld,idi}) = i * h_l, 0 \leq i \leq 2^l.
\]
In a piecewise linear setting, the simplest choice of a 1D basis function is the standard hat function
\[
\phi(x) = \begin{cases} 
1 - |x|, & \text{if } x \in [-1,1], \\
0, & \text{otherwise}.
\end{cases}
\]
This mother of all piecewise multi-linear basis functions can be used to generate an arbitrary \( \phi_{l,j}(x_j) \) with support \([x_{l,i} - h_{l,j}, x_{l,i} + h_{l,j}] = [(i_j - 1)h_{l,j}, (i_j + 1)h_{l,j}]\) by dilation and translation, which is
\[
\phi_{l,j}(x_j) = \phi \left( \frac{x_j - i_j * h_{l,j}}{h_{l,j}} \right).
\]
The above 1D basis functions are the input of the tensor product construction which provides a suitable piecewise \( d \)-linear basis function in each grid point \( x_{l,i} \),
\[
\phi_{l}(x) = \prod_{j=1}^{d} \phi_{l,j}(x_j).
\]
Then the discrete approximation spaces can be defined as
\[ V_l = \text{span}\{\phi_{l,i}: 1 \leq i \leq 2^l - 1\}. \]

Additionally, we introduce the hierarchical increments \( W_l \)
\[ W_l = \text{span}\{\phi_{l,i}: 1 \leq i \leq 2^l - 1, i_j \text{ odd for all } 1 \leq j \leq d\}, \]
for which the relation
\[ V_l = \bigoplus_{k \leq l} W_k, \]
is easily observed. Note that the supports of all basis functions \( \phi_{l,i} \) spanning \( W_l \) are mutually disjoint, thus with the index set
\[ I_l = \{i \in \mathbb{N}^d: 1 \leq i \leq 2^l - 1, i_j \text{ odd for all } 1 \leq j \leq d\}, \]
we get another basis of \( V_l \), the hierarchical basis
\[ \{\phi_{k,i}: i \in I_k, k \leq l\}. \]

Suppose \( n \) stands for the maximum level in the hierarchical basis, we define a new discrete approximation space \( V_n^{(1)} \)
\[ V_n^{(1)} = \bigoplus_{|l|_1 \leq n + d - 1} W_l, \]
then the grids that correspond to the spaces \( V_n^{(1)} \) are just the standard sparse grids. The dimension of the space \( V_n^{(1)} \), i.e., the number of degrees of freedom or inner grid points, is given by
\[
\left| V_n^{(1)} \right| = \sum_{|l|_1 \leq n + d - 1} 2^{|l|-1}_1 = \sum_{i=d}^{n+d-1} 2^{i-d} \cdot \sum_{|l|_1=l} 1 = \sum_{i=d}^{n+d-1} 2^{i-d} \cdot \binom{i-1}{d-1} \\
= \sum_{i=0}^{n-1} 2^i \cdot \binom{d+i-1}{d-1} \\
= \frac{1}{(d-1)!} \cdot \sum_{i=0}^{n-1} (x^{i+d-1})^{(d-1)}|_{x=2} = \frac{1}{(d-1)!} \cdot \frac{x^{d-1} - x^n}{1-x}|_{x=2}
\]
\[ = \frac{1}{(d-1)!} \sum_{i=0}^{d-1} \binom{d-1}{i} \cdot (x^{d-1} - x^{n+d-1})^{(i)} \cdot \left( \frac{1}{1-x} \right)^{(d-1-i)} \bigg|_{x=2} \]

\[ = (-1)^d + 2^n \sum_{i=0}^{d-1} \binom{n+d-1}{i} \cdot (-2)^{d-1-i} = 2^n \cdot \left( \frac{n^d-1}{(d-1)!} + O(n^{d-2}) \right) \]

\[ = O(h_n^{-1} \cdot \log_2 h_n \cdot d^{-1}) = O(N \cdot (\log N)^{d-1}). \]

The left question to be discussed concerns the interpolation accuracy that can be obtained on sparse grids. For that, we look at the interpolation error \( u - u_n^{(1)} \) of the sparse grid interpolant \( u_n^{(1)} \in V_n^{(1)} \), which can be written as

\[ u - u_n^{(1)} = \sum_l u_l - \sum_{|l|_1 \leq n+d-1} u_l = \sum_{|l|_1 > n+d-1} u_l. \]

Therefore, for any norm \( \| \cdot \| \), we have

\[ \| u - u_n^{(1)} \| \leq \sum_{|l|_1 > n+d-1} \| u_l \|. \]

The following lemma provides a prerequisite for the estimates of the interpolation error with respect to the different norms we are interested in. For \( d, n \in N \), we define

\[ A(d, n) = \sum_{k=0}^{d-1} \binom{n+d-1}{k} = \frac{n^d-1}{(d-1)!} + O(n^{d-2}). \]

With Lemma 1 and set \( s = 2 \), we get

\[ \| u - u_n^{(1)} \|_\infty \leq \sum_{|l|_1 > n+d-1} \| u_l \|_\infty \leq \frac{|u|_{2,\infty}}{2d} \cdot \sum_{|l|_1 > n+d-1} 2^{-2||l||_1} \]

\[ \leq \frac{2 \cdot |u|_{2,\infty}}{8d} \cdot 2^{-2n} \cdot A(d, n) \]

\[ = O(h_n^2 \cdot n^{d-1}) = O(N^{-2} \cdot (\log N)^{d-1}). \]

Analogously, we have the result for the \( L_2 \)-norm.
\[ \| u - u_n^{(1)} \|_2 = O(N^{-2} \ast (\log N)^{d-1}). \]

Concerning the first bound with respect to the energy norm, we have

\[
\| u - u_n^{(1)} \|_E \leq \sum_{|l|_1 > n + d - 1} \| u_l \|_E \leq \frac{|u|_{2,\infty}}{2 \cdot 12^\frac{d-1}{2}} \cdot \sum_{|l|_1 > n + d - 1} 4^{-|l|_1} \cdot \left( \sum_{j=1}^{d} 4^{|l|_1 j} \right)^{\frac{1}{2}}
\]

\[
= \frac{|u|_{2,\infty}}{2 \cdot 12^\frac{d-1}{2}} \cdot \sum_{i=n+d}^{\infty} 4^{-i} \cdot \sum_{|l|_1=i} \left( \sum_{j=1}^{d} 4^{|l|_1 j} \right)^{\frac{1}{2}} \leq \frac{|u|_{2,\infty}}{2 \cdot 12^\frac{d-1}{2}} \cdot \sum_{i=n+d}^{\infty} d \cdot 2^{-i}
\]

\[
= \frac{d \cdot |u|_{2,\infty}}{2 \cdot 3^\frac{d-1}{2} \cdot 4^{d-1}} \cdot 2^{-n} = O(h_n) = O(N^{-1}).
\]
References


[112]. http://hpc.kaust.edu.sa/


[127]. http://petrochemical.gronerth.com/flash_distillation_program_explorer.htm


198


[142]. http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics


