Computational Geosciences
https://doi.org/10.1007/s10596-023-10219-0

ORIGINAL PAPER

Adaptive mesh refinement in locally conservative level set methods for multiphase fluid displacements in porous media

Deepak Singh[1] · Helmer André Friis[1] · Espen Jettestuen[2] · Johan Olav Helland[3]

Received: 6 December 2022 / Accepted: 4 May 2023

Abstract
Multiphase flow in porous media often occurs with the formation and coalescence of fluid ganglia. Accurate predictions of such mechanisms in complex pore geometries require simulation models with local mass conservation and with the option to improve resolution in areas of interest. In this work, we incorporate patch-based, structured adaptive mesh refinement capabilities into a method for local volume conservation that describes the behaviour of disconnected fluid ganglia during level set simulations of capillary-controlled displacement in porous media. We validate the model against analytical solutions for three-phase fluid configurations in idealized pores containing gas, oil, and water, by modelling the intermediate-wet oil layers as separate domains with their volumes preserved. Both the pressures and volumes of disconnected ganglia converge to analytical values with increased refinement levels of the adaptive mesh. Favourable results from strong and weak scaling tests emphasize that the number of patches per processor and the total number of patches are important parameters for efficient parallel simulations with adaptive mesh refinement. Simulations of two-phase imbibition and three-phase gas invasion on segmented 3D images of water-wet sandstone show that adaptive mesh refinement has the highest impact on three-phase displacements, especially concerning the behaviour of the conserved, intermediate-wet phase.

Keywords Local volume conservation · Adaptive mesh refinement · Level set method · Pore-scale simulation · High performance computing

1 Introduction

Multiphase flow in porous media occurs in a wide range of engineered applications in the subsurface, including enhanced oil recovery, permanent CO_2 storage, temporary gas storage, and remediation of non-aqueous pollutants. For all these applications it is important to determine the extent of bypassed or trapped fluids after invasion of another fluid. This requires investigations on the pore scale where the mechanisms for the formation of fluid ganglia by detachment from the flowing phases can be described. During the flow process, such ganglia may be displaced, split, reconnect with other ganglia, or eventually get capillary trapped [50, 54, 64]. Pore geometry, rock-fluid and fluid-fluid interactions, and fluid flow mechanics control such trapping. In three-phase flow, capillary forces can lead to double displacements at the pore scale where the continuous phase displaces a disconnected second phase that displaces the third phase [18, 42, 54]. These double displacements can lead to significant fluid redistribution in the pore space and mobilization of the trapped phase. Hence, modelling ganglion behaviour in two- and three-phase systems at the pore scale is required to estimate the extent of capillary trapping at larger scales. This calls for pore-scale methods capable of preserving mass locally during ganglion splitting and merging. Such three-dimensional simulations are numerically demanding for accurate results. But, adaptive gridding can improve computational performance since higher grid resolutions are only needed near the interfaces.

Helmer André Friis, Espen Jettestuen and Johan Olav Helland contributed equally to this work.

Supplementary information The supplementary information file provided with this article contains additional results from scaling tests, close-up of three-phase configurations in different corners of the triangular pore, equilibrium states for two cylindrical pores, and evolution videos of two-phase and three-phase displacement in the Castlegate sandstone geometry.

✉ Deepak Singh
deepak.singh@uis.no

Extended author information available on the last page of the article

Published online: 13 July 2023

Computational fluid dynamics (CFD) techniques for modelling multiphase flow at the pore scale include lattice-Boltzmann methods [29, 30, 35, 49, 57, 59, 67, 71], smoothed particle hydrodynamics methods [32, 65], and phase-field methods [31, 60, 72]. More conventional CFD methods typically combine the Navier-Stokes equations with a technique for interface capturing, like the volume-of-fluid (VoF) method [23, 48, 56] or the level set (LS) method [44, 55]. These interface-tracking methods are implicit, so that they can handle topological changes (like ganglia splitting and merging) in a natural manner. The VoF method describes the phase locations with an indicator function, and its motion is governed by a conservation law [23]. Thus, the VoF method has intrinsic volume/mass conservative properties, but it requires advanced numerical methods to handle the sharp changes of the indicator function at interfaces and for calculating interface properties like curvature. The LS method is based on advection of a smooth LS function typically given as the signed distance to the interface. As such, the LS method is not conservative, but interface properties like curvature is easily calculated directly from the LS function. Several techniques has been developed to improve conservation of LS methods, including the implementation of higher order numerical schemes [40], particle LS methods [9], combined LS and VoF methods [61], the conservative LS method which adopts a modified LS function [41], as well as various interface-correction methods [13, 37].

Using the above methods for direct simulation of multiphase flow on segmented images of the pore structure, while accounting for capillary, viscous, and gravitational forces, is computationally costly. However, for multiphase flow at the pore scale in reservoir rock, capillary forces normally dominate over viscous forces and gravity [21]. Thus, it is adequate to approximate the flow as multiphase displacements controlled by capillary pressure, which makes for more efficient simulations. These displacements are generally modelled as a series of quasi-static equilibrium states for different prescribed capillary pressures [17, 69]. Such quasi-static simulations generate capillary pressure-saturation curves for drainage and imbibition that can be correlated against experimental data. Methods in this category include pore-network modelling based on invasion percolation [2, 43, 45, 73], pore-morphology methods [15, 22, 39, 66], and LS methods [18, 27, 47]. Within interface-capturing methods, we also note that the MBO method [38] (named after its authors) and related variants [51, 62], which solves a diffusion equation and updates an indicator function in alternate steps, can be used to simulate curvature-driven motion in multiphase systems. Common to all these approaches is that they need to be supplied with methods for local conservation of mass (or volume, for incompressible fluids) to accurately describe the behaviour of fluid ganglia that detach from the continuous phases.

Jettestuen et al. [28] presented a new method for local volume conservation (LVC) of fluid ganglia within the setting of capillary-controlled displacement, using LS methods for two-phase systems [27] and the multiphase level set method (MLS) for three-phase systems [18], abbreviated as LS-LVC and MLS-LVC methods, respectively. The LS/MLS-LVC methods have the option to enforce conservation on selected phases, and it calculates surface area, volume, and pressure of individual fluid ganglia of the conserved phases during displacement. A volume redistribution algorithm handles volume conservation during splitting and coalescence of isolated domains. Jettestuen et al. [28] applied the LS/MLS-LVC methods to various 2D and 3D geometries, where it showed a quadratic volume convergence with respect to resolution of the uniform grid. The model was used to simulate gas invasion after primary drainage by conserving only the oil phase in one case and both oil and water phases in another. It demonstrated that local volume conservation of both phases predicts a higher residual oil saturation than only oil conservation. However, we expect finer grids or adaptive mesh refinement (AMR) to improve the resolution of this problem, especially with respect to oil layers in three-phase systems and small fluid ganglia. Consequently, the improved model should generate more realistic capillary pressure curves and distributions of trapped fluid ganglia, allowing a more accurate ganglion characterization. Thus, the objective of this work is to extend the LS/MLS-LVC methods such that they can utilize AMR capabilities.

AMR is a computational technique that can help in the dynamic focussing of refinement to resolve local features in the computational domain without incurring time penalties associated with global refinement. It is an effective tool for improving the accuracy of numerical simulations while maintaining reasonable computational efficiency [10]. In general, there are two main categories for AMR implementation, the patch-based approach [4, 11, 14] and the tree-based approach [46, 63, 70]. The patch-based approach requires tagging unresolved regions using special refinement criteria and then grouping these grid cells into patches with a finer mesh [14]. On the other hand, a quadtree approach works on the principle of recursive decomposition, where a quadtree stores the data [63]. Although the second method is computationally fast and efficiently uses storage memory, its application can require complicated algorithms and interpolations to manage hanging nodes. Jettestuen et al. [28] implemented the LS/MLS-LVC methods for uniform grids using the software framework SAMRAI [3, 24, 25], which is a C++ library for parallelism and AMR that uses ghost cells and a patch-based approach to provide excellent parallel scalability while handling a large number of processors [14]. Ghost cells of a patch are extensions into the neighbouring patches where the values on the grid are generated through coarsening and refinement of values on the neighbouring patches.

Coarsening operations transfer data from finer grids to coarser grids, and refining operations transfer data from coarser grids to finer grids.

In this work, we extend the work of [28] by including AMR capabilities for the LS/MLS-LVC methods and integrating it in the parallel framework of SAMRAI. While SAMRAI provides AMR capabilities for real numbers (e.g., to describe scalar fields), the challenge in our LVC application is handling integers on a map that represents isolated domains. The paper is organized as follows: First, Section 2 briefly describes the LS/MLS-LVC models, while Section 3 presents the methodology and related algorithms for incorporating AMR in the LVC method. In Section 4 we validate the MLS-LVC implementation with AMR against analytical solutions for three-phase configurations in a triangular pore. We discuss the computational efficiency obtained by using AMR rather than a corresponding uniform fine grid in the simulations. Then, we provide weak and strong scaling test results to demonstrate the parallel scalability of the new algorithm. Subsequently, we investigate the impact of AMR on simulations of two-phase imbibition and three-phase gas invasion with disconnected oil ganglia on 3D segmented pore-space images of sandstone. Finally, Section 5 summarizes the results.

2 Methods

We investigate capillary-controlled multiphase systems of gas (g), oil (o) and water (w) at the pore scale, in which a series of capillary equilibrium states for different imposed capillary pressures describes the fluid displacement. At these equilibrium states, the capillary pressure across the fluid/fluid interfaces in the pore space satisfies the Young-Laplace equation:

$$P_{\alpha\beta} = p_\alpha - p_\beta = \sigma_{\alpha\beta} C_{\alpha\beta}, \quad \alpha\beta = go, ow, gw, \tag{1}$$

where $P_{\alpha\beta}$ [Pa] is the capillary pressure, $\sigma_{\alpha\beta}$ [Nm^{-1}] is the interfacial tension between phases α and β, p_α [Pa] is the pressure of phase α, and $C_{\alpha\beta}$ [m^{-1}] is the interface curvature. The fluid/fluid interface meets the pore/solid interface at a contact angle $\theta_{\alpha\beta}$, which is measured through the denser fluid phase β and satisfies Young's equation:

$$\sigma_{\alpha\beta} \cos\theta_{\alpha\beta} = \sigma_{s\alpha} - \sigma_{s\beta}, \quad \alpha\beta = go, ow, gw, \tag{2}$$

where $\sigma_{s\alpha}$ and $\sigma_{s\beta}$ are the solid/fluid interfacial tensions. By combining Young's equations for all three fluid pairs, we obtain the Bartell-Osterhof equation [e.g.,8],

$$\sigma_{gw} \cos\theta_{gw} = \sigma_{go} \cos\theta_{go} + \sigma_{ow} \cos\theta_{ow}, \tag{3}$$

which constrains the interfacial tensions and contact angles in three-phase systems at thermodynamic equilibrium.

In this section, we briefly describe the LS [27] and MLS [18] methods for capillary-controlled displacement, as well as the technique used for incorporating local volume conservation (LVC) in these methods [28].

2.1 Level set method for two-phase capillary-controlled displacement

The level set (LS) method is an Eulerian approach to model propagation of interfaces under the influence of a velocity functional [1, 44, 52]. In this method, a level set function ϕ, normally given by the signed distance function, describes the interfaces as $\phi = 0$, so that ϕ takes different signs on the inside and the outside of a fluid. The equation of motion for an evolving level set function ϕ under action of a normal velocity F_N and an advection velocity $\mathbf{F}_A$ is given by Adalsteinsson and Sethian [1]:

$$\phi_t + F_N |\nabla\phi| + \mathbf{F}_A \cdot \nabla\phi = 0. \tag{4}$$

During evolution with Eq (4), the level set function ϕ gets distorted and requires repeated reinitialisation to maintain its signed distance nature. The reinitialisation is carried out using the following equation [44]:

$$\phi_t + S(\phi)(|\nabla\phi| - 1) = 0, \tag{5}$$

where $S(\cdot)$ is a smoothed sign function given by

$$S(\phi) = \frac{\phi}{\sqrt{\phi^2 + |\nabla\phi|^2 (\Delta x)^2}}, \tag{6}$$

and Δx is the grid spacing.

LS methods have previously been used to model capillary-controlled displacement at the pore scale [27, 47]. Following Jettestuen et al. [27] the LS function ϕ describes the interface between fluid phases α and β as the zero contour $\phi = 0$, where $\phi < 0$ in phase α and $\phi > 0$ in phase β. Another LS function ψ is static and describes the pore geometry, so that $\psi > 0$ in pore space, $\psi < 0$ in solid, and $\psi = 0$ on the pore/solid interface. A sharp Heaviside function $H(\cdot)$ separates the velocity term in the pore space, which governs the capillary-controlled displacement, from the velocity term in the solid space, that forms the contact angle θ on the pore walls. The evolution equation (4) takes the following form [27]:

$$(\phi)_t + H(\psi)\,(C - \kappa)\,|\nabla\phi| + \frac{H(-\psi)}{\Delta x} S(\psi)\,(\nabla\phi \cdot \nabla\psi - |\nabla\phi||\nabla\psi| \cos\zeta) = 0. \tag{7}$$

Here, $C = P_{\alpha\beta}/\sigma_{\alpha\beta}$ is a prescribed interface curvature, $\kappa = \nabla \cdot \nabla\phi/|\nabla\phi|$ is the curvature field of the LS function ϕ, Δx

is grid spacing, $S(\cdot)$ is the sign function, and ζ is a fluid/solid intersection angle which is related to the contact angle by $\zeta = 180^\circ - \theta_{\alpha\beta}$. H is the sharp Heaviside function given by:

$$H(\psi) = \begin{cases} 0 & \psi < 0, \\ 1 & \psi \geq 0, \end{cases} \tag{8}$$

At capillary equilibrium ($\phi_t = 0$) the velocity terms in equation (7) are zero in vicinity of the interfaces. Then we obtain $C = \kappa$ (Young-Laplace equation) in the pore space, while in the solid space and on the pore walls, the intersection angle ζ satisfies

$$\cos\zeta = \mathbf{n}_\phi \cdot \mathbf{n}_\psi = \frac{\nabla\phi}{|\nabla\phi|} \cdot \frac{\nabla\psi}{|\nabla\psi|}, \tag{9}$$

where $\mathbf{n}_\phi$ and $\mathbf{n}_\psi$ are unit normal vectors of ϕ and ψ, respectively. After each calculated equilibrium state using equation (7), we can increase or decrease C stepwise to simulate drainage or imbibition, respectively. A more detailed description of the LS method with numerical validation and applications are provided by Jettestuen et al. [27], Helland et al. [17], and Friis et al. [11].

2.2 Multiphase level set method for three-phase capillary-controlled displacement

The MLS method describes each fluid phase α by its own unique LS function ϕ_α, so that $\phi_\alpha < 0$ inside phase α, $\phi_\alpha > 0$ on the outside, and $\phi_\alpha = 0$ on the boundary of α to the other phases. While this representation can describe an arbitrary number of phases, we focus here on three-phase displacement where $\alpha = g, o, w$. For each fluid phase, we also specify a phase pressure p_α, surface tension γ_α, and a fluid/solid intersection angle ζ_α which is measured through phase α and defined by $\cos\zeta_\alpha = \mathbf{n}_{\phi_\alpha} \cdot \mathbf{n}_\psi$. The surface tensions and intersection angles satisfy the following equations:

$$\sigma_{\alpha\beta} = \gamma_\alpha + \gamma_\beta, \tag{10}$$

$$\sigma_{\alpha s} = \gamma_s - \gamma_\alpha \cos\zeta_\alpha, \tag{11}$$

$$\cos\theta_{\alpha\beta} = \frac{\gamma_\beta \cos\zeta_\beta - \gamma_\alpha \cos\zeta_\alpha}{\gamma_\beta + \gamma_\alpha}, \tag{12}$$

where $\alpha\beta = go, ow, gw$, and γ_s is an imaginary surface tension for the solid phase that ensures $\sigma_{\alpha s}$ is always positive. Hence, equation (12) is the same as Young's equation (2). Following [18], the intersection angles are taken as

$$\zeta_w = \theta_{ow}, \quad \zeta_o = 180^\circ - \theta_{ow} \quad \text{and } \zeta_g = 180^\circ, \tag{13}$$

which, by equation (12), leads to a set of contact angles that fulfills equation (3) for a given set of γ_α.

The evolution equations for the fluid LS functions ϕ_α are given by [16, 18]:

$$\begin{aligned} &(\phi_\alpha)_t + H(\psi)\,(p_\alpha - \gamma_\alpha\kappa_\alpha)\,|\nabla\phi_\alpha| + \\ &\frac{H(-\psi)}{\Delta x} S(\psi)\gamma_\alpha\,(\nabla\phi_\alpha \cdot \nabla\psi - |\nabla\phi_\alpha||\nabla\psi|\cos\zeta_\alpha) = 0, \\ &\alpha = g, o, w. \end{aligned} \tag{14}$$

Here, γ and ζ remain constant throughout the simulation. On the other hand, p_α is constant, as specified by the boundary conditions, for continuous phases whereas it varies with time for isolated domains. The calculation of isolated domain pressures is detailed in the next section.

As these evolution equations are uncoupled, the three LS functions ϕ_α will develop overlap and void regions around the interfaces during evolution. Consequently, in an interface region of two phases α and β, the intersection of ϕ_α and ϕ_β occurs at a nonzero contour (i.e., $\phi_\alpha = \phi_\beta > 0$ in case of void region or $\phi_\alpha = \phi_\beta < 0$ in case of overlap region) rather than at the zero contour ($\phi_\alpha = \phi_\beta = 0$) which would correspond to a unique representation of the $\alpha\beta$-interface. This problem is solved using the projection method of Lossaso et al. [36] where, in every grid cell, the average of the two smallest ϕ_α is subtracted from all ϕ_α at the end of each time-iteration step. This step moves all intersections of the different LS functions to zero contours, creating a unique delineation of the phases in the computational domain. When the multiphase system has relaxed to a steady state, i.e., $(\phi_\alpha)_t = 0, \alpha = g, o, w$, the solution of equation (14) (with projection step included) satisfies Young-Laplace equation in the pore space and Young's equation on the pore walls [16, 18]:

$$P_{\alpha\beta} = p_\alpha - p_\beta = (\gamma_\alpha + \gamma_\beta)\kappa_\alpha, \tag{15}$$

$$\cos\theta_{\alpha\beta} = \frac{\gamma_\beta \cos\zeta_\beta - \gamma_\alpha \cos\zeta_\alpha}{\gamma_\beta + \gamma_\alpha} = \frac{\nabla\phi_\beta}{|\nabla\phi_\beta|} \cdot \frac{\nabla\psi}{|\nabla\psi|}, \tag{16}$$

since $\phi_\alpha = -\phi_\beta$ in the vicinity of the $\alpha\beta$ interfaces, which implies $\nabla\phi_\alpha = -\nabla\phi_\beta$ and $\kappa_\alpha = -\kappa_\beta$.

The MLS method uses non-dimensional parameters for length, surface tension and pressure, L, γ, and p, respectively. These parameters are derived from actual values of length L^{actual} [m], surface tension γ^{actual} [Nm^{-1}] and pressure p^{actual} [Pa], using characteristic properties for the physical system:

$$L = \frac{L^{actual}}{L^*}, \ \gamma = \frac{\gamma^{actual}}{\gamma^*}, \ p = \frac{p^{actual}}{p^*} \text{ and } p^* = \frac{\gamma^*}{L^*}, \tag{17}$$

where L^* [m], γ^* [Nm^{-1}], and p^* [Pa] are the characteristic parameters.

In both LS and MLS methods, the numerical discretization approximates the normal and advective velocity terms using a weighted essentially non-oscillatory (WENO) scheme with appropriate upwinding techniques, while the curvature terms use central differences. The iteration-time discretization uses a third-order Runge-Kutta method with a time step determined from a standard Courant-Friedrichs-Lewy (CFL) condition [44]. In this work, the CFL coefficient is 0.6. Faster temporal discretization methods exist for steady-state problems on structured mesh [33, 68], but it remains to investigate their applicability to capillary-controlled displacement on complex pore geometries. The convergence in the MLS method is declared when the difference of ϕ_α between two reinitializations, denoted by m and n, in a small band around the zero contours of the LS functions in pore space is lower than a specified tolerance value:

$$\max\left\{\frac{\sum_\Omega H_\epsilon(\psi+\epsilon)H_\epsilon(\phi_\alpha^n+\lambda)H_\epsilon(-\phi_\alpha^n+\lambda)|\phi_\alpha^n-\phi_\alpha^m|}{\sum_\Omega H_\epsilon(\psi+\epsilon)H_\epsilon(\phi_\alpha^n+\lambda)H_\epsilon(-\phi_\alpha^n+\lambda)}\right\} < c\Delta x,\ \forall\,\alpha=g,o,w \tag{18}$$

where H_ϵ is a smoothed Heaviside function and $\epsilon = 1.5\times\Delta x$ is the smoothening parameter. In equation (18) $\lambda = b\times\epsilon$ is the width of the convergence band. In this work, we set $b = 5$ and $c = 0.001$.

Note that the interface position can slightly shift after a reinitialization step. This effect is most significant in MLS simulations of triple junctions between the three fluids where rounded features in the zero contours could form and result in small overlap/void regions. This impact of reinitialization is most severe when the angle at the triple junction through one of the fluids is small (as determined by the interfacial tensions). In our experience this region is about the size of one to two grid cells, so the impact is small. The projection step [36] in the MLS iteration after the reinitialization removes these small regions.

2.3 Local volume conservation method

Jettestuen et al. [28] developed a technique for local volume conservation (LVC) of fluid ganglia in the LS and MLS methods. Here, we describe the LVC method for a selected conserved phase α in the MLS setting. The main idea is to determine a spatially and temporal phase pressure $p_\alpha(\mathbf{x},t)$ for use in equation (14) that conserves the volumes of disconnected ganglia of phase α during level set evolution. To this end, the method first identifies the disconnected domains of phase α using a grassfire method [47] that is based on Chebyshev distance (i.e., L_∞-norm) as metric with 8-connectivity (in 2D) and 26-connectivity (in 3D). Note that the Chebyshev distance from the gridcell to these connected blocks is 1 unit. We denote the identified domains at iteration time t as $\Omega_a(t)$, where $a\in I_\alpha(t)$, and $I_\alpha(t)$ is a numbered list of all ganglia of phase α at time t. Then, we construct *extended* domains $\Omega_a^{\text{ext}}(t)$ that completely encloses domains $\Omega_a(t)$, so that $\Omega_a^{\text{ext}}(t)\supset\Omega_a(t)$. The extended domains include all points nearest to the boundary of the corresponding domains as determined by a distance map, using city-block distance (i.e., L_1-norm) as metric. Note that city-block distance (also referred to as Manhattan distance or L_1-norm) is the minimum number of grid cells traversed laterally to reach a point from another point. The extended domains are needed to distribute volumes correctly from one iteration time step to another, but they also ensure accuracy in the numerical discretization of equation (14) across interfaces. These extended regions partition the computational domain Ω completely:

$$\Omega = \left(\cup_{a\in I_\alpha(t)}\Omega_a^{\text{ext}}(t)\right)\cup\Omega_{\alpha,\text{bnd}}^{\text{ext}}(t). \tag{19}$$

Here, we have distinguished the extended disconnected regions $\Omega_a^{\text{ext}}(t)$ from the extensions $\Omega_{\alpha,\text{bnd}}^{\text{ext}}(t)$ of all regions $\Omega_{\alpha,\text{bnd}}$ that are connected to inlet or outlet boundaries and referred to as the continuous phase. The continuous phase changes volume as phase α enters or leaves the computational domain.

Then, the phase pressure for a conserved phase α is given by Saye and Sethian [52]:

$$p_\alpha(\mathbf{x},t)=\begin{cases}\dfrac{V_a^{(0)}(t)-V_a(t)}{\Delta t\,A_a(t)}, & \mathbf{x}\in\Omega_a^{\text{ext}}(t), a\in I_\alpha(t),\\ p_{\alpha,\text{bnd}}, & \mathbf{x}\in\Omega_{\alpha,\text{bnd}}^{\text{ext}}(t).\end{cases} \tag{20}$$

Here, $p_{\alpha,\text{bnd}}$ is the continuous-phase pressure assigned to regions connected to inlet or outlet boundaries. If phase α is the invading phase, stepwise increments of $p_{\alpha,\text{bnd}}$ after every capillary equilibrium state drives the quasi-static displacement. Further, $V_a^{(0)}(t)$ is the target volume of a disconnected region Ω_a, while $V_a(t)$ and $A_a(t)$ are volume and surface area of the region, respectively, calculated at each iteration-time t:

$$V_a(t) = \int_{\Omega_a^{\text{ext}}(t)} H_\epsilon(\psi)H_\epsilon(-\phi_\alpha)\mathrm{d}V, \tag{21}$$

$$A_a(t) = \int_{\Omega_a^{\text{ext}}(t)} H_\epsilon(\psi)\delta_\epsilon(\phi_\alpha)|\nabla\phi_\alpha|\mathrm{d}V, \tag{22}$$

where δ_ϵ is a smoothed Delta function. Calculation of global saturation uses equation (21) with integral over the entire computational domain Ω. These smoothed Heaviside and Delta functions are calculated as:

$$H_\epsilon(\phi_\alpha)=\begin{cases}0 & \phi_\alpha<-\epsilon,\\ \frac{1}{2}+\frac{\phi_\alpha}{2\epsilon}+\frac{1}{2\pi}\sin\left(\frac{\pi\phi_\alpha}{\epsilon}\right) & -\epsilon\le\phi_\alpha\le\epsilon,\\ 1 & \phi_\alpha>\epsilon,\end{cases} \tag{23}$$

and,

$$\delta_\epsilon(\phi_\alpha) = \begin{cases} 0 & \phi_\alpha < -\epsilon, \\ \frac{1}{2\epsilon} + \frac{1}{2\epsilon}\cos\left(\frac{\pi\phi_\alpha}{\epsilon}\right) & -\epsilon \le \phi_\alpha \le \epsilon, \\ 0 & \phi_\alpha > \epsilon. \end{cases} \tag{24}$$

A conservative volume redistribution technique assigns volumes to disconnected domains at the next time step based on the volume of domains at the previous time step. The method accounts for potential ganglia splitting and coalescence. Assume we have identified the sets of domains and extended domains at iteration time t_1, $\{\Omega_a(t_1) : a \in I_\alpha(t_1)\}$ and $\{\Omega_a^{\text{ext}}(t_1) : a \in I_\alpha(t_1)\}$, and similarly, at time $t_2 = t_1 + \Delta t$ these sets are $\{\Omega_b(t_2) : b \in I_\alpha(t_2)\}$ and $\{\Omega_b^{\text{ext}}(t_2) : b \in I_\alpha(t_2)\}$. The intersection between the domain $\Omega_a(t_1)$ and the extended domain $\Omega_b^{\text{ext}}(t_2)$ is

$$\Delta\Omega_{ab}(t_1, t_2) = \Omega_a(t_1) \cap \Omega_b^{\text{ext}}(t_2). \tag{25}$$

The target volume assigned to domain $\Omega_b(t_2)$ is given as

$$V_b^{(0)}(t_2) = \sum_{a \in I_\alpha(t_1)} \frac{|\Delta\Omega_{ab}(t_1, t_2)|}{|\Omega_a(t_1)|} V_a^{(0)}(t_1), \tag{26}$$

where $|\cdot|$ denotes the volume measured in number of voxels. Equation (26) uses the relative volume of overlap between regions at t_1 and t_2 as a weight for the volume redistribution. The set of extended regions cover the entire computational domain, see equation (19), which ensures volume conservation during the redistribution. Further, restrictions on time step in the MLS method guarantees small interface changes within an iteration step, so that target domain volumes are redistributed correctly. We remark that continuous-phase domains $\Omega_{\alpha,\text{bnd}}^{\text{ext}}(t)$ are not part of the redistribution equations as their pressure is prescribed. Their role is merely to limit the extensions of nearby isolated domains. When new regions form by detachment from the connected phase, their target volumes $V_a^{(0)}(t)$ are calculated from equation (21). Thus, equation (20) implies that the pressures of such regions will be zero in the first time iteration step. However, their temporary volumes $V_a(t)$, also calculated by equation (21), will generally differ in subsequent iterations, resulting in nonzero domain pressure.

The phase-by-phase representation of the MLS method allows for applying LVC to one, two or three phases in a three-phase system. The LS method, which instead uses one LS function to describe interfaces in a two-phase system, will also require a similar phase-by-phase representation to account for LVC of both phases. However, conservation of one phase follows the same approach as described above. The main difference is that the LS method constructs a spatial and temporal interface curvature $C(\mathbf{x}, t)$ that replaces pressure (equation (20)):

$$C(\mathbf{x}, t) = \begin{cases} \Upsilon \frac{V_a{}^{(0)}(t) - V_a(t)}{\Delta t A_a(t)}, & \mathbf{x} \in \Omega_a^{\text{ext}}(t), a \in I(t), \\ C_{\text{bnd}}, & \mathbf{x} \in \Omega_{\text{bnd}}^{\text{ext}}(t), \end{cases} \tag{27}$$

where $\Upsilon = 1$ if $\phi < 0$ represents the conserved phase, and $\Upsilon = -1$ if instead $\phi > 0$ represents the conserved phase.

3 LVC methodology for AMR and parallelism

Here we introduce the methodology and algorithms for parallel AMR implementation of MLS-LVC method. Apart from the differences described in the previous section, the AMR implementation for LS-LVC method will be similar. An important part of the LVC algorithm presented by Jettestuen et al. [28] is the identification of isolated fluid domains. The software implementation uses a patch-based approach, so the identification must first be performed on each patch. However, since a domain usually crosses (often several) patch borders, unique global labelling (using integers) is necessary. As discussed in [28], the corresponding computational algorithm requires considerable communication in its parallel version, which is implemented using Message Passing Interface (MPI). The LVC algorithm constructs extended domains by extending the global labels outward into the computational grid from the interface of each isolated domain. In the parallel setting, patch boundaries limit these extensions, so that equation (19) does not generally hold. Thus, additional extension procedures are carried out across patch boundaries to construct sufficiently large extended domains. As mentioned in Section 2.3, the role of the extended domains are as follows: (i) to conserve domain volumes during evolution, (ii) to redistribute volume during domain splitting and coalescence, (iii) to calculate volume and surface area of the domains, and (iv) to distribute the calculated phase pressure on the computational grid (see also sections 3 to 5 in [28] for details on LVC application for uniform grids).

A major challenge in applying local volume conservation on a uniform grid is creating a computationally efficient way of identifying isolated domains and populating appropriate domain pressures on the geometry. Furthermore, these pressures should be suitably extended from the interfaces so that the size of the numerical stencil does not affect the level set evolution. We handle this requirement by linking pressures to the extended regions and using city-block distances for modifying extended regions to suitable distances from the interfaces across patches. However, the case for AMR grids

is more complex than for uniform grids. Firstly, AMR grids require additional attention in handling integers during data communication between patches of different sizes. Secondly, the city-block distance between patches with different grid spacings will not correspond to each other. Finally, AMR application requires regular regridding (changing the grid structure) to maintain accuracy and be computationally efficient. Consequently, additional efforts are needed to ensure that various variables involved keep the same values before and after regridding.

The MLS-LVC model is implemented in C++ programming language using the SAMRAI software framework. SAMRAI allows flexibility in cell refinement criteria and frequency of regridding. The gridding and regridding process is accomplished in a linear order time [14]. We have utilised the framework to generate patches with finer mesh and communicate relevant values between patches through coarsening and refining variables in the ghost cells. In SAMRAI, adaptive grids are a layered structure where finer grids are successively placed on top of the coarsest grid (see Fig. 1). The calculations are done on the topmost layer or mesh (i.e., the finest grid) for each overlapping location. When needed, the values are transferred to other meshes by coarsening or refining operations.

In the following subsections, we provide the LVC algorithm for AMR grids followed by a discussion on the necessary modifications made to the original (uniform grid) algorithm for AMR compatibility.

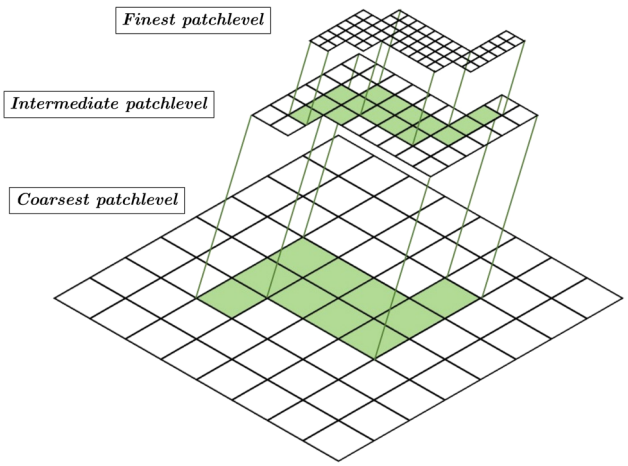

Fig. 1 AMR grid structure in SAMRAI framework with three patch levels. The regions tagged for refinement are shown in *green*. The illustration shows a case where each tagged grid cell is refined into four equally-sized grid cells. The calculations at any spatial location are done on the highest available patch level and then transferred to lower levels through ghost cells. For details, see Anderson et al. [3]

3.1 MLS-LVC algorithm with AMR

The LVC algorithm for AMR grids is a slight modification of the parallel algorithm for uniform grids from Jettestuen et al. [28](Section 5.1). The improvements reduce the memory space requirement for uniform grids while adding the capability to handle AMR grids. The steps in the modified algorithm are as follows, assuming α is a conserved phase:

1. Initialize the system at initial iterative time t_0:
   (a) Isolated domain identification (Algorithm 1, Section 3.2): Construct domains $\{\Omega_a : \phi_\alpha \leq 0$ and $\|\Omega_a - \Omega_b\|_\infty > 1, \forall\ a \neq b,\ a$ and $b \in I_\alpha(t_0)\}$. Here, $\|\Omega_a - \Omega_b\|_\infty$ represents the Chebyshev distance between the boundaries of domains Ω_a and Ω_b. Figure 2 (*top row*) depicts isolated domains identified under different levels of mesh refinement.
   (b) Generate a set of extended domains $\{\Omega_a^{\text{ext}} : a \in I_\alpha(t_0)\}$ (Algorithm 2, Section 3.2). Figure 2 (*middle row*) shows the result after extension with different levels of refinement.
   (c) Modify extended domains (Algorithm 3, Section 3.2): Modify the set of extended regions $\{\Omega_a^{\text{ext}} : a \in I_\alpha(t_0)\}$ by extending the domain labels across patches. The modification ensures that the numerical stencils for grid cells near patch borders and fluid/fluid interfaces use the correct pressures in the numerical scheme. Figure 2 (*bottom row*) shows the result after modification of extension maps with different levels of refinement.
   (d) Set $t_n = t_0$:
   Loop over iteration time (from t_n to t_{n+1}):
2. Set pressure of conserved phase, $p_\alpha(\mathbf{x}, t_n)$. For continuous-phase domains, assign phase pressure $p_{\alpha,\text{bnd}}$ to $\Omega_{\alpha,\text{bnd}}^{\text{ext}}(t_n)$. For isolated domains, set pressures in $\Omega_a^{\text{ext}}(t_n), a \in I_\alpha(t_n)$, using equation (20) (enforcing volume conservation) as follows:
   (a) On each patch: Perform computations according to equations (21) and (22).
   (b) On each processor: Aggregate the volume and area information and communicate it to the root processor.
   (c) On the master processor: Calculate pressure of isolated domains using equation (20) and communicate relevant information back to all processors and patches.
3. Iterate the multiphase system from t_n to t_{n+1} with equation (14) using the phase pressure from the previous step and then enforcing the projection step [36]. This is done in parallel using the patch-based parallelism in SAM-

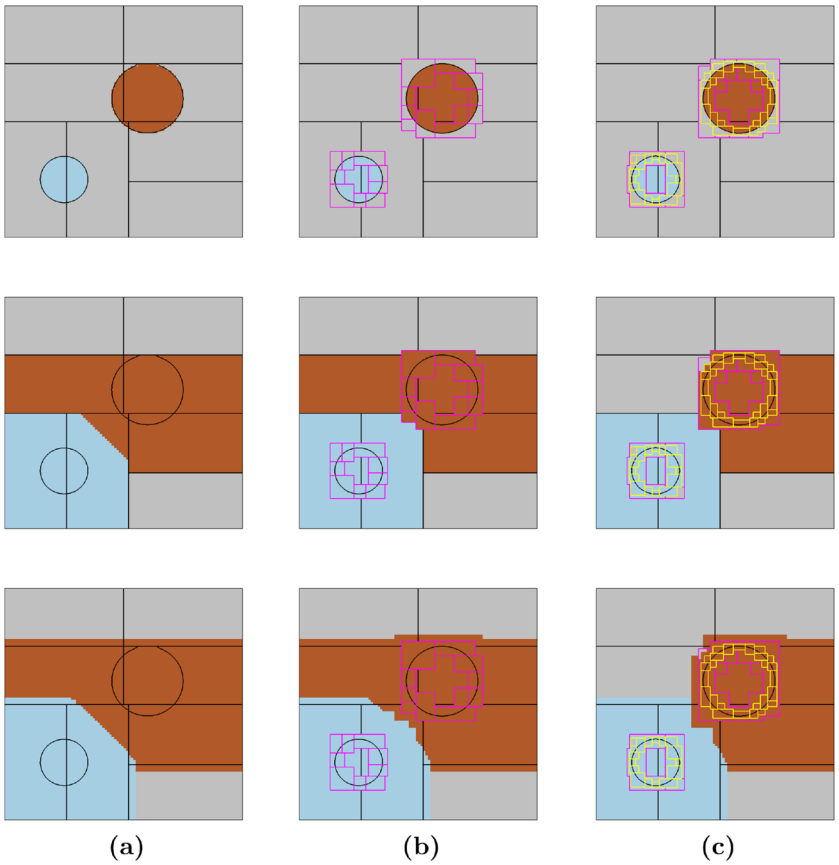

Fig. 2 Example with two isolated circular regions that illustrates the identification (*top*), extension (*middle*), and correction (*bottom*) of domain integer values on patch for (a) uniform coarse grid, (b) one level of grid refinement, and (c) two levels of grid refinement. The figure highlights the boundaries of coarsest patches (*black*), the boundaries of patches with one level of refinement (*purple*) and two levels of refinement (*yellow*)

RAI. Here, we also periodically reinitialise level sets and check for convergence.

4. Repartition the system at time step t_{n+1}, i.e., construct isolated domains $\{\Omega_{a'} : \phi_\alpha \leq 0$ and $\|\Omega_{a'} - \Omega_{b'}\|_\infty > 1, \forall\, a' \neq b', a'$ and $b' \in I_\alpha(t_{n+1})\}$ in a similar manner as in step 1 (a) above. Then, extend the isolated domains to obtain the set $\{\Omega^{\text{ext}}_{a'}(t_{n+1})\}$ in a similar manner as in step 1 (b) and (c) above.
5. Redistribute volumes according to equations (25) and (26) while excluding continuous phase domains ($\Omega^{\text{ext}}_{\alpha,\text{bnd}}$).
6. Identify new regions detached from the continuous phase and calculate their region target volumes based on equation (21).
7. Regrid at regular intervals $\{n \mod Q = 0 : Q$ is the regridding interval$\}$ (Algorithm 4, Section 3.2).
8. Go to step 2 with $n \longleftarrow n + 1$.

3.2 Considerations for adaptive mesh refinement

Transforming the LVC algorithm for compatibility with an adaptive grid necessitates reconciling variables between meshes with different grid spacings. Compared to integers, handling real numbers is easy; the values from a grid of one size can be easily transferred onto a grid of another size through interpolation, extrapolation, or averaging. SAMRAI provides operators that carry out these operations for real numbers. However, averaging integers can produce undesired results for our purpose. For example, if four adjacent

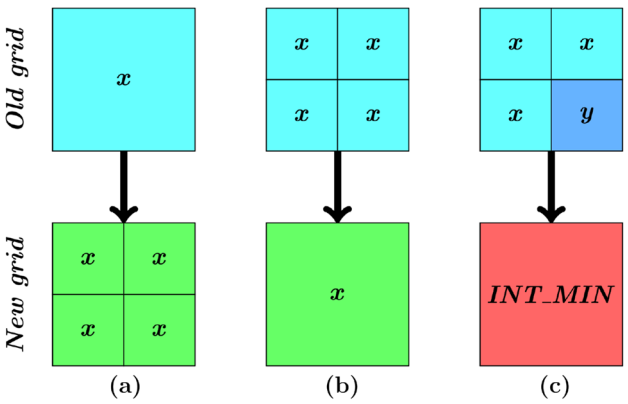

Fig. 3 Integer operators where x and y are integer values and *INT_MIN* is the minimum integer limit in C++. (a) Refining operator, (b) coarsening operator 1 for equal values, and (c) coarsening operator 1 for different values

fine grid cells have values 1, 2, 4 and 5 in the extended domain map, then the coarser grid cell below them will be wrongly filled with 3 after averaging. Thus, we define our own operators for coarsening and refining of the integer data type (see Fig. 3).

Refining is easy since it creates copies of the coarser value on the finer grid (see Fig. 3 (a)). Coarsening is more challenging since the finer grid cells over a coarser grid cell can either have equal values or different values. So, we create an operator that copies the value on to the coarser grid cell if all the finer grid cells over it have the same value (see Fig. 3 (b)), otherwise it returns the minimum possible C++ integer value, *INT_MIN* (see Fig. 3 (c)). This unique value on the coarser grid allows us to check if the overlaying finer grid cells have the same values or not. We use this operator on patch variables M and E that store the isolated domains map and the extended domains map, respectively. This operator helps in preserving most of the original values on the coarsened regions and the rest can be recovered by extending the map values on the new grid using Algorithm 2 and 3. We also create another coarsening operator that always returns *INT_MIN*. This second operator is used for dynamic identification of refined domains on the coarser patch throughout the simulation. In this way the various computations (e.g., volume computations) can be performed with good parallel efficiency. We use this second operator on patch variable P that stores the processor rank. Since Open MPI library provides only positive integers as processor identification number, *INT_MIN* acts as a distinct value that will not be provided by the MPI library.

In our model, integers are used for identifying isolated domains Ω_a and their extended domains Ω_a^{ext}, $a \in I_\alpha(t)$. Algorithm 1 describes the steps followed in identifying isolated domains of a particular phase on the computational grid in the modified algorithm. We identify isolated domains on a given patch using the grassfire algorithm with ϕ_α as the input. The grassfire algorithm (referred to as grassfire transform in image processing) can be described as 'setting fire' to the zero level set contour. It finds the closest region to each grid cell in the patch. The algorithm requires alternate sweeps of the patch from any two diagonally opposite corners while measuring backward distances from the area already swept. In this patch-level identification, we use *INT_MAX* (the maximum integer limit in C++) to represent domains connected to the computational grid boundary. We label the isolated domains found on each processor sequentially (starting from 1) but the uniquely labelled isolated domains on a processor are not necessarily disconnected from each other. Thus, we use the processor number and the local domain number to identify each domain uniquely.

Algorithm 1 Identify domains.

Require: Patch variables: Phase level set value (ϕ_α), processor rank (P), domain map (M), and number of grid cells in z-direction (n_z), y-direction (n_y), and x-direction (n_x)

Ensure: ($\Omega_a : \phi_\alpha \leq 0$) **and** ($\|\Omega_{\alpha,bnd} - \Omega_a\|_\infty > 1$) **and** ($\|\Omega_b - \Omega_a\|_\infty > 1,\ \forall\ a \neq b$)

1: Initialise patch variable M to zero.
2: Create an empty list L_l on each processor where l is the rank of processor in *MPI_COMM_WORLD*. Also, create an empty list L_{agg} on root processor.
3: Update P to the rank of processor. Then, update P and M on the refined regions of a coarse patch to *INT_MIN*
4: Identify domains on a patch where $\phi_\alpha \leq 0$ using grassfire algorithm. Update M by labelling the domains uniquely using incremental tags on each processor. Identify domains connected to boundary with *INT_MAX*
5: Fill ghost cell data
6: **for** each gridcell $\{i, j, k\}$ on each patch boundary **do**
7: $Local = \{i, j, k\}$ and $Neighbour = adjacent\ ghostcell$
8: Identify neighbours in a tuple with four elements, $N = \{P_{Local}, M_{Local}, P_{Neighbour}, M_{Neighbour}\}$
9: **if** ($P_{Local} \geq 0$) **and** ($P_{Neighbour} \geq 0$) **and** ($M_{Local} > 0$) **and** ($M_{Neighbour} > 0$) **then**
10: **if** ($P_{Local} < P_{Neighbour}$) **or** ($M_{Local} = INT\_MAX$) **then**
11: Save neighbours tuple N in list L_l
12: **else if** ($P_{Local} = P_{Neighbour}$) **and** ($M_{Local} < M_{Neighbour}$) **then**
13: Save neighbours tuple N in list L_l
14: **else**
15: Ignore the neighbour pair
16: **end if**
17: **end if**
18: **end for**
19: On root processor, $L_{agg} \supset L_l, \forall\ l \in MPI\_COMM\_WORLD$
20: On root processor, $\forall\ N \in L_{agg}$, use P as an indicator to reidentify patch-level unique M to globally sequential values M^*.
21: **while** on root processor and there exists $N \in L_{agg} : M^*_{Local} \neq M^*_{Neighbour}$ **do**
22: Set both $M^* \in N$ to $\min\{M^*_{Local}, M^*_{Neighbour}\}$
23: For the M^* value changed in the last step, update all the corresponding M^* in L_{agg}
24: **end while**
25: Reidentify all unique M^* sequentially to M_{global}
26: Send relevant M_{global} to each processor
27: Update M on the patches to M_{global}

Steps 6 to 18 of Algorithm 1 check if two identified isolated domains are adjacent to each other. Since step 4 prevents such an identification inside the patch, it is sufficient to look at patch boundary grid cells and the ghost cell adjacent to them to find such pairs. Steps 9 to 17 of Algorithm 1 ensure that for each neighbouring pair spanning adjacent patches, one patch identifies the pair and the other ignores it. We analyzed and compared the computational efficiency of sending the complete initial dataset of identified neighbours to the root processor vs. sending a reduced dataset to the root processor after identifying unique isolated domains on each proces-

Algorithm 2 Define extended domains

Require: Patch variables: Isolated domain label (M), extended domain label (E), distance (D), ratio of patch grid spacing w.r.t. finest patch grid spacing (R), and number of grid cells in z-direction (n_z), y-direction (n_y), and x-direction (n_x)

Ensure: $E_{i,j,k} = a$ **if** $(\|(i,j,k)-\Omega_a\|_1 < \|(i,j,k)-\Omega_b\|_1)$, $\forall a, b \in M$

1: Update ghost cell data
2: Initialise E and D: $E = 0$, $D = -1 \times n_x \times n_y \times n_z \times R$
3: **if** Isolated domain is present on the patch **then**
4: **for** $k \in [0, n_z)$, $j \in [0, n_y)$, $i \in [0, n_x)$ **do**
5: **if** $M_{i,j,k} > 0$ **then**
6: $D_{i,j,k} = 0$
7: $E_{i,j,k} = M_{i,j,k}$
8: **else**
9: $D_{i,j,k} = \max\{D_{i,j,k}, \max\{D_{i-1,j,k}, D_{i,j-1,k}, D_{i,j,k-1}\}+R\}$
10: Update $E_{i,j,k}$ to E of grid cell that modified $D_{i,j,k}$ above
11: **end if**
12: **end for**
13: **for** $k \in (n_z, 0]$, $j \in (n_y, 0]$, $i \in (n_x, 0]$ **do**
14: **if** $M_{i,j,k} < 0$ **then**
15: $D_{i,j,k} = \max\{D_{i,j,k}, \max\{D_{i+1,j,k}, D_{i,j+1,k}, D_{i,j,k+1}\} + R\}$
16: Update $E_{i,j,k}$ to E of grid cell that modified $D_{i,j,k}$ above
17: **end if**
18: **end for**
19: In case of AMR simulations, repeat steps 4 to 12
20: **end if**

sor. Our analysis shows that sending the original dataset to the root processor provides faster results. This study is detailed in Appendix A. On the root processor, steps 21 to 25 in Algorithm 1 provide globally unique labels to isolated domains found on the patch (for details refer [28], Appendix A). Figure 2 (*top row*) depicts an example of isolated domain identification with different levels of refinement.

Algorithm 3 Modify extended domains across patches

Require: Patch variables: Processor rank (P), extended domain label (E), distance (D), ratio of patch grid spacing w.r.t. finest patch grid spacing (R), modification distance threshold (d_c, in city-blocks), and number of grid cells in z-direction (n_z), y-direction (n_y) and x-direction (n_x)

1: **if** region with values from refined areas exists (identified by $P = INT\_MIN$) **then**
2: **for** $k \in [0, n_z)$, $j \in [0, n_y)$, $i \in [0, n_x)$ **do**
3: $d_{finer} = \|(i,j,k) - Finer\ patch\ boundary\|_1 \times R$
4: **if** $(d_{finer} < d_c$ **and** $D_{i,j,k} > D_{finer-ghostcell} + d_{finer})$ **or** $E_{i,j,k} = 0$ **then**
5: $E_{i,j,k} = E_{finer-ghostcell}$
6: **end if**
7: **end for**
8: **end if**
9: **for** $k \in [0, n_z)$, $j \in [0, n_y)$, $i \in [0, n_x)$ **do**
10: $d = \|(i,j,k) - Patch\ boundary\|_1$
11: **if** $(d < d_c$ **and** $D > D_{ghostcell} + d)$ **or** $E_{i,j,k} = 0$ **then**
12: $E_{i,j,k} = E_{ghostcell}$
13: **end if**
14: **end for**

Algorithm 4 Restructure variables during regridding

Require: Patch variables: Phase level set value (ϕ_α), Processor rank (P), domain label (M), extended domain label (E), distance (D), ratio of patch grid spacing w.r.t. finest patch grid spacing (R), modification distance threshold (d_c, in city-blocks), and number of grid cells in z-direction (n_z), y-direction (n_y) and x-direction (n_x). Array variables: original volume ($V_\alpha^{(0)}$), temporary volume (V_α), and temporary area (A_α) to store $V_a^{(0)}$, V_a, and A_a (where $a \in M$), respectively for use in equation (20).

Ensure: New variables are sized according to the new grid geometry

1: Reinitialize the phase level set values to signed distance function
2: Tag cells for refinement and regrid
3: Refine and coarsen ghost cell data
4: Update the extended domains using Algorithm 2
5: Modify the extended map using Algorithm 3
6: Create new variables for volume and area, $V_\alpha^{(0)*}$, V_α^*, and A_α^*, sized according to new grid geometry
7: Fill the new variables ($V_\alpha^{(0)*}$, V_α^*, and A_α^*) using old variables ($V_\alpha^{(0)}$, V_α, and A_α) such that $E_{new} = E_{old}$
8: Free memory used by variables ($V_\alpha^{(0)}$, V_α, and A_α) based on old grid geometry and use the new variables for further pressure calculations

We generate the set of extended domains Ω_a^{ext} using Algorithm 2 on each patch. On uniform grids, we accomplish it by carrying out two sweeps of the patch for each grassfire run (steps 4 to 18). But for AMR grids, the extended regions are generated by running three sweeps of the patch. With AMR, some parts on coarser patches have *INT_MIN* values since finer patches cover them. Generally, in our model, the finer patches are not distributed in such a complicated manner that multiple grassfire run sweeps are required to distribute the pressure values effectively. In such scenarios, the additional sweep of the patch (step 19, which is a rerun of the first sweep) provides the correct minimum distance needed in the grassfire algorithm. Figure 2 (*middle row*) shows the result after extension with different levels of refinement.

These extended regions on the patch are further extended into neighbouring patches by using distances from interfaces and patch neighbourhood information (see Algorithm 3). As mentioned earlier, the city-block distances are measurements where one unit distance is equal to one grid cell. However, the city-block distance on patches with different grid spacing will represent different actual distances. To ensure that this difference does not lead to insufficient domain extensions across borders of patches with different grid spacings, we use the ratio of patch grid spacing to make the distances proportionate such that the finest grid cell equals one unit of distance. For example, suppose the coarse grid spacing is two times bigger than the finest grid spacing. In that case, one fine-patch grid cell equals one unit of distance, and one coarse-patch grid cell equals two units of distance. Another point of importance is the order in which these extensions are carried out from one patch to another. For uniform grids,

the order of extension is inconsequential. However, on AMR grids, the order is important to ensure that the level set velocities near zero contour are independent of the patch structure. So, we first carry out extensions from finer patches to coarser patches below them (steps 1 to 8 of Algorithm 3). This first run ensures that if an interface lies only on the finest patch, its effect extends into all the neighbouring coarse patches. Then, we carry out a second extension (as for uniform grid algorithm in Jettestuen et al. [28]) on all the patches (steps 9 to 14). Since the minimum required depth of modification depends on the numerical stencil size, an optimal way to implement this algorithm is to find the necessary patch boundary grid cells. In steps 1 to 8, this boundary is the boundary of a finer patch lying inside the coarse patch, and the distance of the ghost cell underlying the neighbouring finer patch from the closest domain on that finer patch is $D_{finer-ghostcell}$. While for steps 9 to 14, it is the boundary with a patch on the same or lower level, and the distance of the ghost cell from the closest isolated domain on that neighbouring patch is $D_{ghostcell}$. Once the requisite boundary grid cell has been found, we can check whether the extended domain value on the grid cells within the required distance from this grid cell needs to be modified. Figure 2 (*bottom row*) shows the result after modification of extension maps with different levels of refinement.

In our model, we store the original volume, temporary volume, and temporary area (required in equation (20)) in 1D arrays based on the patch grid structure. We number the patches on each processor sequentially and store these variables for extended domain labels present on that patch. For example, if patch number 1 on processor 0 has extended domain labels 2 and 3, then our array starts with the original volume, temporary volume, and temporary area for domains 2 and 3, respectively. If patch number 2 on processor 0, has extended domain labels 1, we append the original volume, temporary volume, and temporary area for domain 1 to these arrays. This array shape helps in limiting the communication between root processors and other processors since we can use $MPI\_Gatherv$ and $MPI\_Scatterv$ to share the arrays efficiently. We update these variables after each regridding operation using Algorithm 4. During regridding, we create similar variables based on the newer grid configuration. We refine and coarsen all patch variables using integer operators (described previously) and real operators (provided by SAMRAI) before any other operation. Then, we extend the domains using Algorithms 2 and 3. We utilize the extended domain labels to link the variables based on the previous grid structure to those based on the new grid structure. Then, we erase the older variables from memory space and use the newer ones in their place for further calculations.

4 Results and discussion

Here, we present simulation results with the LS/MLS-LVC models using the new AMR capabilities. We start by validating the model against analytical results for three-phase configurations in a 2D triangular pore and a 3D cylindrical pore, followed by analysing the parallel performance of the AMR implementation through scaling tests. Then, we apply the LS/MLS-LVC models on segmented micro-CT images of a sandstone to explore the significance of AMR in simulations of two-phase and three-phase capillary displacements with disconnected fluid ganglia.

In this section, the phrase '1-level' or 'AMR-1' describes simulations using AMR with one level of refinement, while '2-levels' or 'AMR-2' describes simulations using AMR with two levels of refinement. The phrase 'LVC-1' and 'LVC-2' refer to local volume conservation of one and two phases, respectively. For visualization purposes, we use the software VisIt developed by LLNL [6]. To visualize the fluid configuration inside a pore geometry, we confine the zero contours of ϕ_α to the pore space using $\phi_\alpha = \max(\phi_\alpha, -\psi), \alpha = g, o, w$.

4.1 Validation against three-phase fluid configurations in a 2D triangular pore

We consider three-phase fluid configurations in a 2D triangular pore where water is the wetting phase residing in pore corners, oil is present as intermediate-wet layers, and gas is the non-wetting phase occupying the central portion of the triangle. This configuration is similar to the one theoretically studied by Hui and Blunt [26]. We will assume the oil layers are disconnected from each other with different pressures and their individual volumes (areas in 2D) conserved. This situation is different from our previous investigations on triangular pores using MLS method where global oil volume conservation led to a common pressure for all oil layers [18]. For stepwise increments of gas pressure at constant water pressure we simulate the displacement of oil layers toward pore corners, applying LVC on the oil phase only. Our objective is to show that higher levels of AMR improve the accuracy in simulations based on a comparison with analytical solutions and that AMR simulations provide similar results as simulations on a corresponding uniform, fine grid.

Geometrically, an oil layer can exist in a corner if the contact angles and interface curvatures, $\theta_{\alpha\beta}$ and $C_{\alpha\beta}$, $\alpha\beta = go, ow$, satisfy the relations [26]:

$$\theta_{\alpha\beta} < \pi/2 - \Theta_k, \quad \alpha\beta = go, ow, \tag{28}$$

and

$$\frac{C_{go}}{C_{ow}} < \begin{cases} \dfrac{\cos\theta_{go} - \sin\Theta_k}{\cos\theta_{ow} - \sin\Theta_k}, & \theta_{go} < \theta_{ow}, \\ \dfrac{\cos\left(\theta_{go} + \Theta_k\right)}{\cos\left(\theta_{ow} + \Theta_k\right)}, & \theta_{go} \geq \theta_{ow}, \end{cases} \tag{29}$$

where Θ_k is the half-angle of corner k. Here we assume that an oil layer ceases to exist when the gas-oil and oil-water interfaces form a triple junction on the pore walls or at the corner bisector. The volume V_o of an oil layer in corner k, sandwiched by corner water and bulk gas, is [18]

$$V_o = \frac{B_{go}}{C_{go}^2} - \frac{B_{ow}}{C_{ow}^2}, \tag{30}$$

where

$$B_{\alpha\beta} = \cos\theta_{\alpha\beta}\left(\frac{\cos\theta_{\alpha\beta}}{\tan\Theta_k} - \sin\theta_{\alpha\beta}\right) + \theta_{\alpha\beta} + \left(\Theta_k - \frac{\pi}{2}\right), \quad \alpha\beta = go, ow. \tag{31}$$

From equations (30) and (31) we find C_{go} as a function of C_{ow}:

$$C_{go} = \sqrt{\frac{B_{go}C_{ow}^2}{V_o C_{ow}^2 + B_{ow}}} \tag{32}$$

The horizontal asymptote for the above equation is

$$C_{go} = \sqrt{\frac{B_{go}}{V_o}}, \tag{33}$$

which corresponds to the situation where all the water has been displaced from the corner.

For the validation, we use a triangle with baseline 10 μm and corner angles 30°, 60° and 90°, located on a spatial domain of size $140L^* \times 95L^*$. The characteristic parameters for the MLS-LVC method is $L^* = 10^{-7}$ m, $\gamma^* = 10^{-4}$ Nm^{-1}, and $P^* = 10^3$ Pa. We model a slightly non-spreading fluid system with interfacial tensions $\sigma_{ow} = 20\times10^{-3}$ Nm^{-1}, $\sigma_{go} = 12 \times 10^{-3}$ Nm^{-1} and $\sigma_{gw} = 30 \times 10^{-3}$ Nm^{-1}. The contact angles are $\theta_{ow} = 50°, \theta_{go} = 30.3°$ and $\theta_{gw} = 39.3°$, representing a water-wet state in which water is absent in the 90°-corner by equation (28). Following [18], we initialize the zero contours of the fluid LS-functions as concentric circles that intersect the pore walls. Then, $\phi_w < 0$ outside the largest circle, $\phi_g < 0$ inside the smallest circle, and $\phi_o < 0$ in the annulus (with thickness 0.5 μm). We initialize the gas and water phase pressures to 110 kPa and 100 kPa, respectively. The isolated oil layer pressures will be calculated from equation (20). So, we start from a gas-water capillary pressure of $P_{gw} = 10$ kPa, and thereafter simulate capillary equilibrium states for each gas-pressure increment of 1 kPa until $P_{gw} = 20$ kPa.

The simulation on the coarse, uniform grid uses grid-cell spacing $\Delta x = 1$, reinitialization frequency once every 5 MLS iterations, and $c = 0.001$ as MLS convergence criterion in equation (18). Now, halving the grid spacing will reduce the CFL-determined time-step four times. So, to make simulations comparable between meshes with different grid spacing, we use a reinitialization frequency of once every 20 iterations for AMR-1 and once every 80 iterations for AMR-2. To maintain comparable convergence criteria, we also set $c = 0.0005$ for AMR-1 and $c = 0.00025$ for AMR-2 in equation (18). Simulations on the corresponding uniform, fine grids with $\Delta x = 0.5$ and $\Delta x = 0.25$ follow the same procedure. In the AMR simulations we tag cells for refinement around interfaces in the pore space according to the criterion $|\phi_\alpha| < \Delta x$ and $\psi > 0$ for $\alpha = g, o, w$. We perform regridding every 1000 iterations for AMR-1 and every 2000 iterations for AMR-2.

Figure 4 shows three-phase fluid configurations for $P_{gw} = 10$ and 20 kPa from the simulation on the finest uniform grid ($\Delta x = 0.25$) and the AMR-1 and AMR-2 simulations. As the AMR simulations form grids with higher resolution only around the fluid/fluid interfaces, the pore geometry is resolved with coarse grid spacing ($\Delta x = 1$) in areas away from the interfaces, leading to rounded pore corners in case of AMR-1 and AMR-2. Such differences in uniform (or initial) grid spacings also lead to minor differences in the input interface location on the computational grids, which contribute to slightly different original volume of the oil layers in the corners. This difference is larger for the 30° corner in comparison to the 60° corner. The supplementary information provides zoomed-in views of the configurations in the different pore corners for $P_{gw} = 10$, 14, and 20 kPa.

A comparison of the results from the AMR-1 simulation with a simulation on the corresponding uniform, fine grid ($\Delta x = 0.5$) shows that the locations of the zero contours of ϕ_α from the two simulations are nearly overlaying each other, see Fig. 5 (a) and (b). The deviation between the two simulations is largest for the oil/water interface in the 30°-corner where the zero contours are separated by at most one coarse grid spacing ($\Delta x = 1$). This separation can be attributed to the slightly different initial oil-layer volumes, which leads to a more pronounced separation as the oil is squeezed deeper into the corner. The overlapping $C_{go}(C_{ow})$-curves shown in Fig. 5 (c) and (d) confirm that the AMR simulation is able to mimic the results expected from the corresponding uniform fine grid.

Figure 6 presents a similar comparison between the AMR-2 simulation and the simulation on the corresponding uniform, fine grid with grid spacing $\Delta x = 0.25$. As for the previous case, the zero contours of ϕ_α are nearly overlapping,

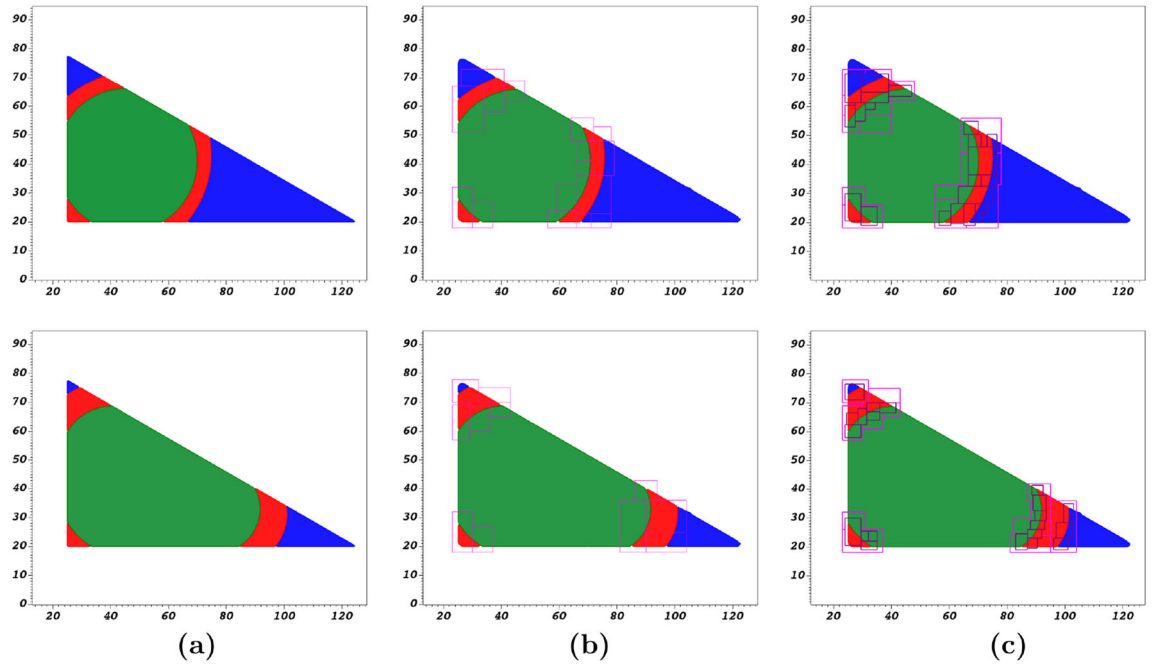

Fig. 4 Three-phase fluid configurations of gas (*green*), oil (*red*) and water (*blue*) for gas-water capillary pressures $P_{gw} = 10$ kPa (*top row*) and $P_{gw} = 20$ kPa (*bottom row*) from simulations in the triangular pore with (a) uniform fine grid ($\Delta x = 0.25$), (b) AMR-1, and (c) AMR-2. (b, c) The results from AMR simulations also show the rectangular patches with finer grid around the interfaces. The coarse grid spacing in the AMR simulations is $\Delta x = 1$. For visualization purposes, we confined the zero contours of ϕ_α to the pore space using $\phi_\alpha = \max(\phi_\alpha, -\psi)$, $\alpha = g, o, w$

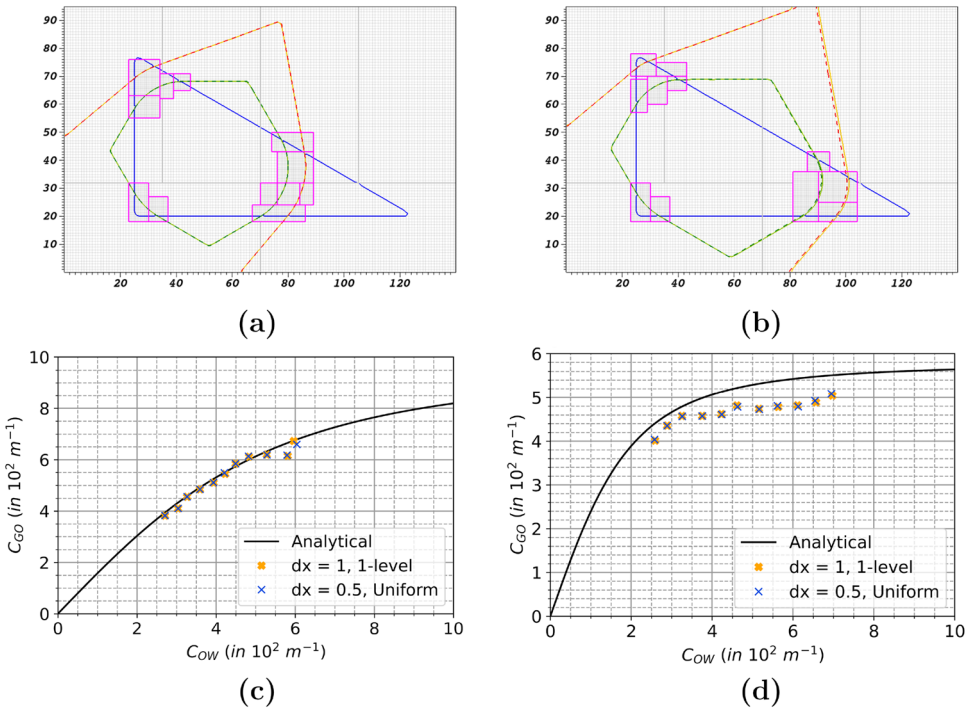

Fig. 5 (a, b) Stable zero-contour configurations of the fluid-phase LS functions from simulations with AMR-1 and uniform grid ($\Delta x = 0.5$) in the triangular pore for (a) $P_{gw} = 14$ kPa and (b) $P_{gw} = 20$ kPa. The figures show $\phi_g = 0$ (*solid, light green* curve (AMR-1) and *dashed, dark green* curve (uniform grid)) and $\phi_w = 0$ (*solid yellow* curve (AMR-1) and *dashed red* curve (uniform grid)). We also plot the fluid/fluid and fluid/solid boundaries of all fluid phases in the pore geometry in *blue* colour by means of the zero contours of $\phi_\alpha = \max(\phi_\alpha, -\psi)$, $\alpha = g, o, w$ from AMR-1 results. For the AMR-1 results, we also show the rectangular patches with finer grid around the interfaces. (c, d) Relationships between interface curvatures C_{go} and C_{ow} for (c) 30°-corner and (d) 60°-corner

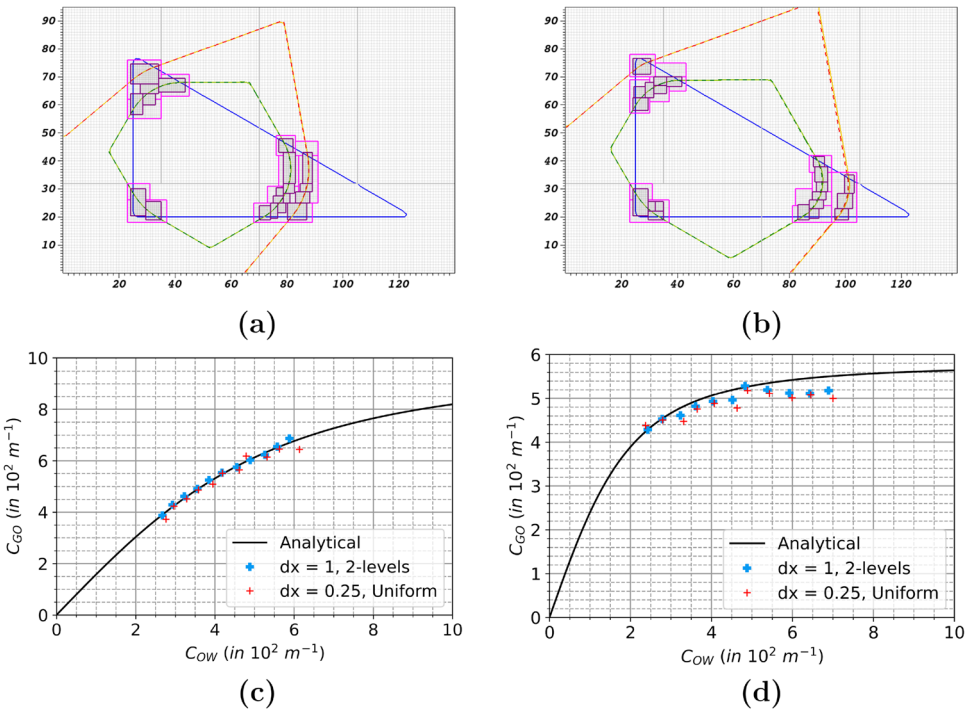

Fig. 6 (a, b) Stable zero-contour configurations of the fluid-phase LS functions from simulations with AMR-2 and uniform grid ($\Delta x = 0.25$) in the triangular pore for (a) $P_{gw} = 14$ kPa and (b) $P_{gw} = 20$ kPa. The figures show $\phi_g = 0$ (*solid, light green* curve (AMR-2) and *dashed, dark green* curve (uniform grid)) and $\phi_w = 0$ (*solid yellow* curve (AMR-2) and *dashed red* curve (uniform grid)). We also plot the fluid/fluid and fluid/solid boundaries of all fluid phases in the pore geometry in *blue* colour by means of the zero contours of $\phi_\alpha = \max(\phi_\alpha, -\psi)$, $\alpha = g, o, w$ from AMR-2 results. The AMR-2 results also display the rectangular patches with the two different levels of finer grid around the interfaces. (c, d) Relationships between interface curvatures C_{go} and C_{ow} for (c) 30°-corner and (d) 60°-corner

see Fig. 6 (a) and (b). The largest difference occurs in the 30° corner around the oil/water interface for $P_{gw} = 20$ kPa, in which the zero contours of ϕ_o this time are separated by at most half a coarse grid spacing ($\Delta x = 1$). The better performance of AMR-2 compared to AMR-1 in providing zero level-set locations closer to the results on the uniform finer grid is due to the improvement in initial oil volume with further refinement of the grid. Figure 6 (c) and (d) show that the AMR-2 simulation also replicates the $C_{go}(C_{ow})$-curves from the simulation on the uniform fine grid, which validates the AMR implementation in the MLS-LVC model.

Figure 7 presents $C_{go}(C_{ow})$-curves from the AMR-1 and AMR-2 simulations along with the results from the simulation on the coarse uniform grid ($\Delta x = 1$). Clearly, the coarse-grid simulation reasonably resolves the interface curvatures in the 30°-corner, while AMR improves accuracy of the solution further (see Fig. 7 (a)). The resolution of oil-water interface is more accurate in the 30°-corner compared to the 60°-corner. So, the most significant improvements with AMR occurs in the 60°-corner, where the results on the coarse, uniform grid noticeably differs from the analytic solution (see Fig. 7 (b)). These results support the application of AMR in complex 3D geometry to improve the accuracy of results. Figure 8 explores convergence behavior of the oil-layer volumes based on these three simulations. We present results from the equilibrium states obtained at every 2 kPa increase of P_{gw} and estimate the relative error $E[V]$ of the oil volumes calculated during the MLS iterations with respect to the initial (target) volume in each corner. In the case of the 90° corner, the relative error for different P_{gw} almost coincide because the absence of water renders the gas-oil interface static. All results show that the application of AMR in the MLS-LVC model yields quadratic volume convergence with respect to grid spacing.

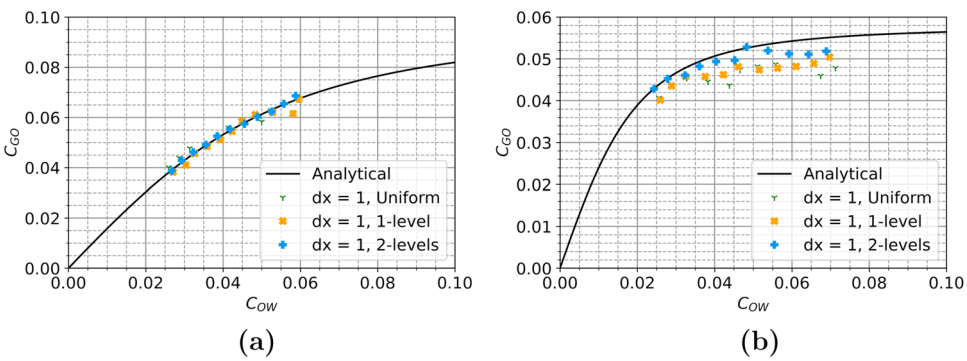

Fig. 7 Analytical solutions for gas-oil curvature vs. oil-water curvature in the triangular pore compared with simulation results from uniform coarse grid, AMR-1, and AMR-2: (a) 30° corner, and (b) 60° corner

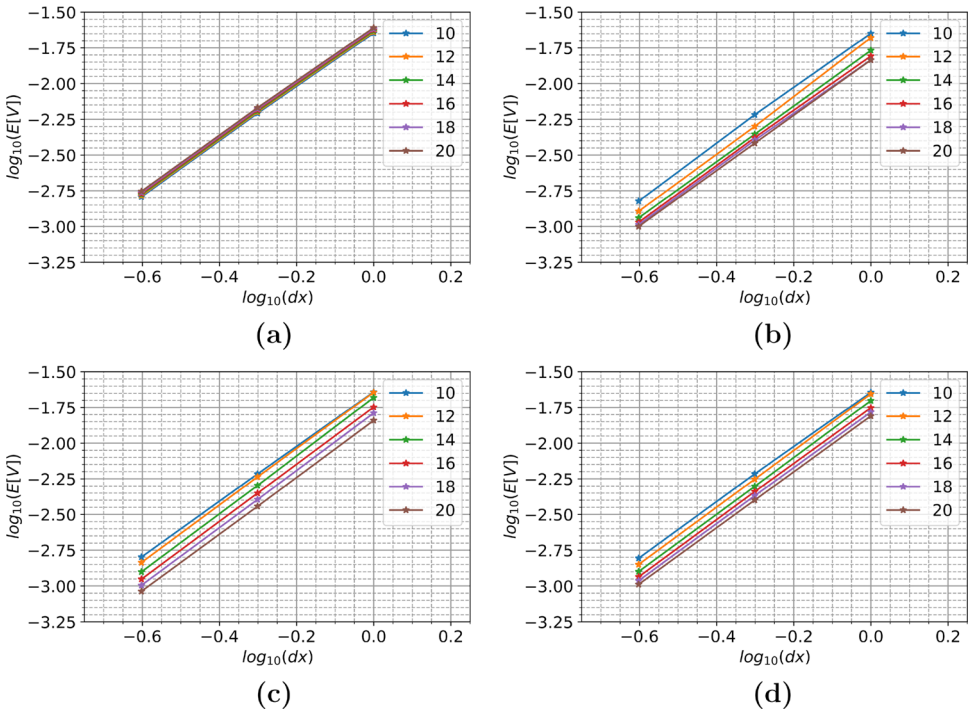

Fig. 8 Convergence of conserved oil-layer volume with respect to grid spacing for different gas-water capillary pressures in the triangular pore: (a) 90° corner, (b) 60° corner, (c) 30° corner, and (d) total oil volume. $E[V]$ is relative error. The data is from uniform grid simulation ($\Delta x = 1$), AMR-1, and AMR-2 simulation. The x-axis represents the grid spacing on the finest grid in the simulation

Finally, Fig. 9 depicts various aspects of the computational performance of the MLS-LVC model with AMR. The total number of grid cells decreases during the AMR simulations because fewer grid cells are tagged for refinement when the interfaces approach the corners, see Fig. 9 (a) and (b). We also note that even the AMR-2 simulation forms a lower number of grid cells than the uniform-grid simulation with grid spacing $\Delta x = 0.5$. Similarly, Fig. 9 (c) shows that the numbers of

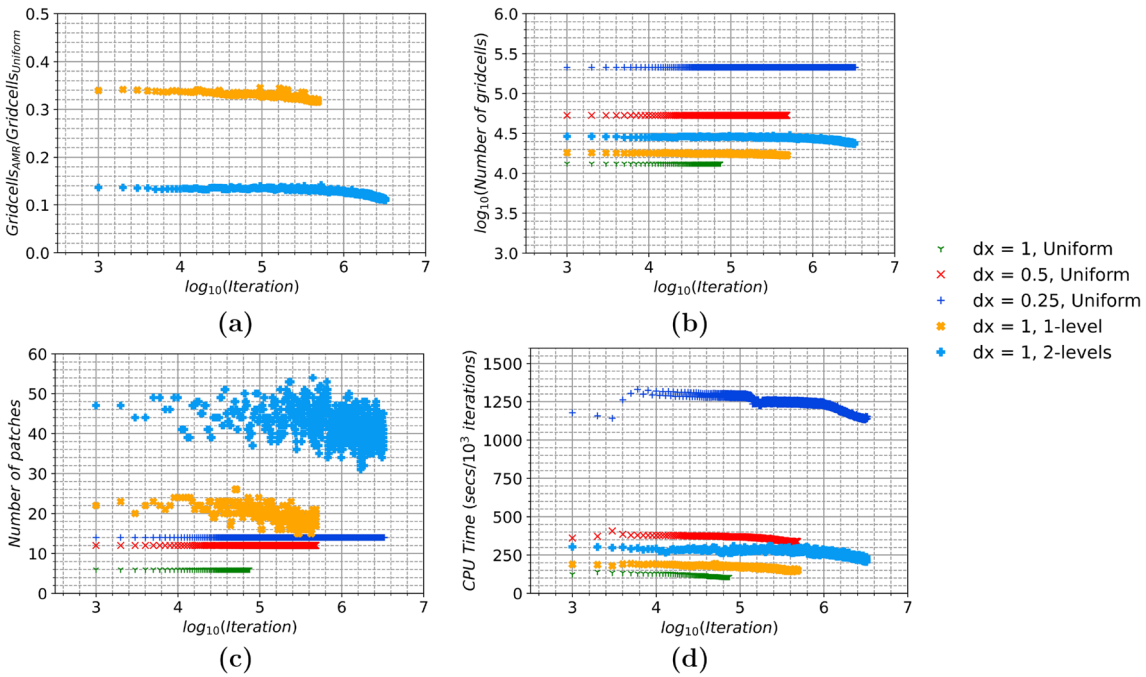

Fig. 9 Performance of simulations with AMR and uniform grids as function of the number of MLS iteration steps. (a) Ratio between the numbers of grid cells for AMR and corresponding uniform, fine grid, (b) number of grid cells (on a logarithmic scale), (c) number of patches, and (d) CPU time taken per 1000 iterations (in seconds)

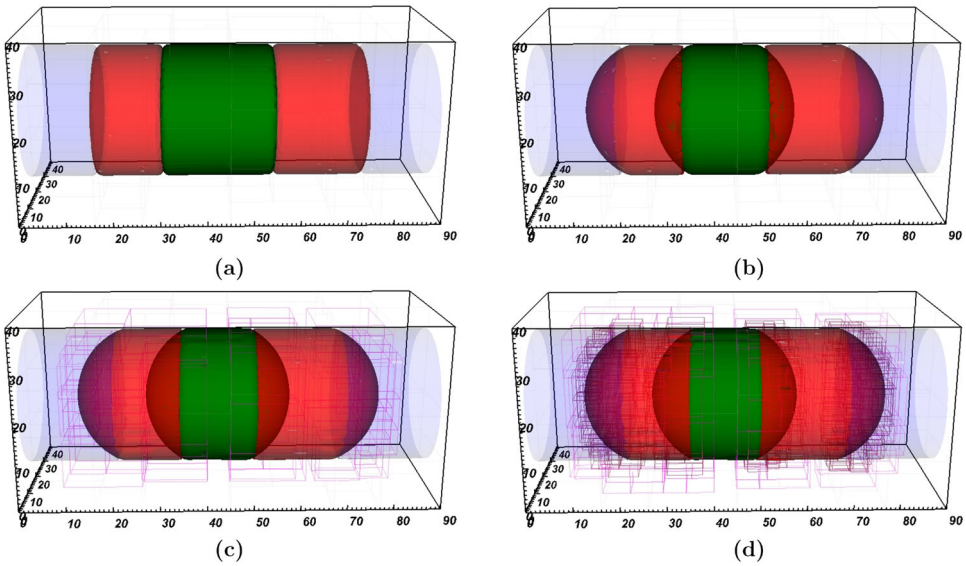

Fig. 10 Three-phase fluid configurations of gas (*green*), oil (*red*) and water (*blue*) in a 3D cylindrical pore with pore radius $= 15\mu$m. (a) Initial configuration, (b) uniform coarse grid ($\Delta x = 1$), (c) AMR-1, and (d) AMR-2. The coarse grid spacing in the AMR simulations is $\Delta x = 1$. The AMR results display the rectangular patches from the refined grid with the first level of refinement in *light purple* and the second level of refinement in *dark purple*

patches decrease during the AMR simulations, even though they remain higher than the constant patch numbers from the simulations with uniform grid.

Figure 9 (d) compares CPU time spent per 1000 iterations in the simulations. The coarse-grid simulation ($\Delta x = 1$) is fastest, while the AMR simulations are faster than the corresponding uniform fine-grid simulations. The number of iterations required to reach the final stable state is the same for the corresponding simulations with AMR and uniform fine grid. All the simulations become progressively faster as the oil layers move deeper into the corners. This is mainly because the periodic reinitializations of ϕ_α require less iterations in the late stage of these simulations. In general, a dynamic CPU time will also arise when the number of patches spanned by conserved domains changes significantly during a simulation, leading to varying communication loads between processors in the LVC method. Specific to the AMR simulations on the triangular pore is that both the numbers of grid cells and patches decrease, although a generally higher number of patches in the cases with AMR can maintain a significant processor communication in the LVC method and lead to more modest declines of CPU time. Still, Fig. 9 (d) shows that, on an average AMR-1 is twice as fast, and AMR-2 more than four times as fast, as the computation on a corresponding uniform fine grid. Combining this efficiency gain with the nearly overlapping results from Fig. 5 (c) and (d)

Table 1 Volume [10^{-15}m^3] and pressure [kPa] of gas bubble and oil droplets. Oil drop -1 refers to the smaller drop

	Gas bubble		Oil drop - 1		Oil drop - 2	
	Pressure	Volume	Pressure	Volume	Pressure	Volume
Pore radius 10 μm						
Uniform	7.606	7.824	5.663	4.688	5.664	6.260
AMR-1	7.705	7.847	5.721	4.706	5.713	6.278
AMR-2	7.741	7.854	5.742	4.711	5.736	6.283
Analytical	7.765	7.856	5.759	4.713	5.759	6.285
Pore radius 12 μm						
Uniform	6.718	11.383	5.084	6.756	5.084	9.019
AMR-1	6.774	11.413	5.115	6.780	5.111	9.043
AMR-2	6.803	11.421	5.131	6.786	5.126	9.049
Analytical	6.804	11.424	5.132	6.789	5.132	9.051
Pore radius 15 μm						
Uniform	5.769	17.622	4.466	10.561	4.466	14.096
AMR-1	5.814	17.662	4.487	10.594	4.484	14.129
AMR-2	5.830	17.672	4.496	10.602	4.490	14.137
Analytical	5.843	17.676	4.506	10.605	4.506	14.140

and Fig. 6 (c) and (d), we can conclude that AMR is computationally important for higher accuracy in large geometries.

4.2 Validation against three-phase fluid configurations in a 3D cylindrical pore

We consider three-phase fluid configurations in a 3D cylindrical pore where water is the wetting phase, oil is the intermediate-wetting phase, and gas is the non-wetting phase. Water is present at the two ends, while two oil droplets of different sizes surround a single gas bubble (see Fig. 10 (a)). We conserve the volume of the gas bubble and the two oil droplets, while we treat the water as a continuous phase with a prescribed pressure. We provide the initial configuration as cylindrical regions (fluid/fluid interfaces are parallel to x-y plane) and allow the gas-oil and oil-water curvatures to develop during simulation based on the fluid phase properties (see Fig. 10 (a)).

For a given water pressure P_w, the oil pressure P_o, and gas pressure P_g can be calculated analytically from Young-Laplace equation as:

$$\begin{aligned} P_o &= P_w + \frac{(2\sigma_{ow}\cos\theta_{ow})}{r}, \\ P_g &= P_w + \frac{(2\sigma_{ow}\cos\theta_{ow}) + (2\sigma_{go}\cos\theta_{go})}{r}, \end{aligned} \tag{34}$$

where r is the pore radius. Note that two oil droplets with different (but sufficiently large) volumes will have identical pressures in such straight tubes.

For the validation, we study the fluid configurations in three pores with length 90 μm, and pore radii 10 μm, 12 μm, and 15 μm. The spatial domain size is $40L^* \times 40L^* \times 90L^*$. The characteristic parameters for the MLS-LVC method are $L^* = 10^{-6}$ m, $\gamma^* = 10^{-4}$ Nm^{-1}, and $P^* = 100$ Pa. Hence, the coarse grid spacing is $\Delta x = 1$. We model a slightly non-spreading fluid system with interfacial tensions $\sigma_{ow} = 20 \times 10^{-3}$ Nm^{-1}, $\sigma_{go} = 11 \times 10^{-3}$ Nm^{-1} and $\sigma_{gw} = 30 \times 10^{-3}$ Nm^{-1}. The contact angles are $\theta_{ow} = 20°$, $\theta_{go} = 24.24°$ and $\theta_{gw} = 16.1°$. We prescribe the water drop pressures to 2 kPa. All other simulation parameters are the same as in the last section. During simulation, the pressures of the two oil droplets and the gas bubble will be calculated from equation (20). Figure 10 (b-d) shows the equilibrium fluid interfaces for pore radius 15 μm. The supplementary information contains images of the equilibrium fluid configurations for the other pore radii.

Table 1 lists the pressure and volume of the gas bubble and the two oil droplets from our simulations and their analytical values. Figure 11 compares analytical and simulated gas and oil pressures, while Fig. 12 shows the convergence of the gas and oil volumes with respect to the grid spacing. Figure 12 confirms a quadratic volume convergence for the

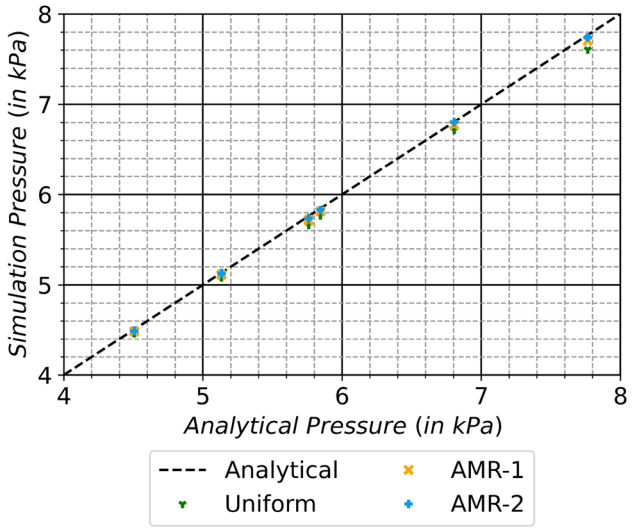

Fig. 11 Comparison of simulation results vs. analytical solutions for oil droplet and gas bubble pressures in the three cylindrical pores of different sizes. In each simulation, the two oil droplet pressures are nearly identical and appear as one data point in the figure. The three leftmost data sets represent oil droplet pressures, and the three rightmost data sets represent gas bubble pressures

different pore sizes. Table 2 lists the computational data for these simulations. It can be seen that the number of iterations required for convergence increases with pore size. The computational time is also affected by the changes in the number of grid cells and patches in the geometry (computational performance analysis is discussed in detail in the next section). The results show that increasing the refinement level improves the accuracy of the pressure and volume results.

4.3 Scaling tests

In this section, we will discuss the parallel performance and scalability of the MLS-LVC method with AMR. For uniform grids, Jettestuen et al. [28] demonstrated that the algorithm performance depends on the number of preserved domains. Here, we investigate the parallel performance with one and two levels of AMR, which are the most relevant cases for application of the method on segmented micro-CT images of rock samples. We carry out weak scaling and strong scaling tests for uniform coarse grid, AMR-1, and AMR-2, while conserving domains of one phase (LVC-1) and two phases (LVC-2).

In the scaling tests, we consider a pore space occupied by an equal number of disconnected oil and gas domains (see Fig. 13). Continuous water occupies the remaining part of the computational domain (solid is absent). Pressures of continuous phases are set equal, while pressures of disconnected domains are calculated from equation (20). The simulations consist of 24 level set iterations for each of the three fluid phases, including the projection step calculations as well as

the LVC calculations for selected phase(s). Three reinitialization solves (each consisting of 10 iterations) are carried out at the 10th and 20th level set iteration step. In AMR simulations we refine all grid cells that satisfy $|\phi_\alpha| < \Delta x, \alpha = g, o, w$. The simulation time in the scaling tests is the duration between the 4th iteration step and the 24th iteration step and thus we exclude gridding computation times. We refer to Gunney and Anderson [14] for an investigation of the parallel performance of AMR regridding within the SAMRAI framework. Even though the scaling performance of SAMRAI is linear for processor clusters much larger than the 512 processors used in our test, the scaling test performance presented in this section should diminish with the frequency of regridding during the test. Overall, the scaling tests capture $60(= 20 \times 3)$ level set iteration steps and $60(= 2 \times 10 \times 3)$ reinitialization steps. All the computations are performed on the supercomputer *Fram* provided by UNINETT Sigma2 - the National Infrastructure for High-Performance Computing and Data Storage in Norway.

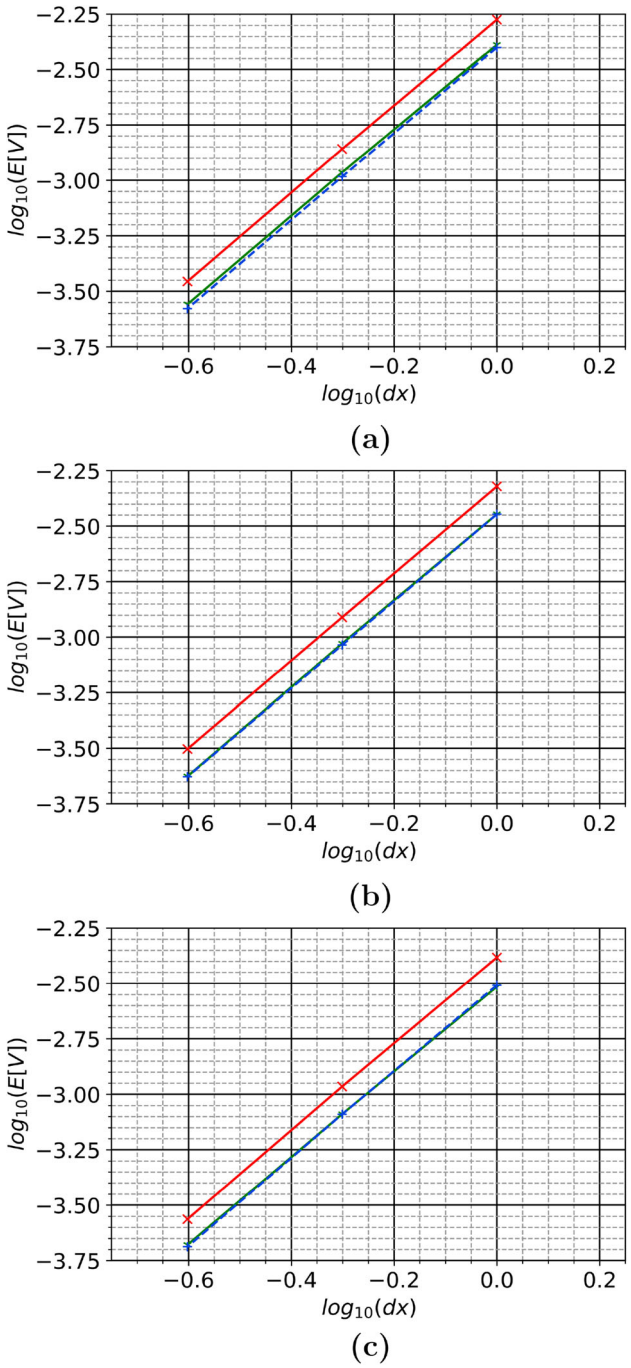

Fig. 12 Volume convergence of oil droplets and gas bubble with respect to the grid spacing in the cylindrical pore: (a) $r = 10\mu$m, (b) $r = 12\mu$m, and (c) $r = 15\mu$m. $E[V]$ is the relative error, and the x-axis represents the grid spacing on the finest grid in the simulation. Gas in *green-solid*, smaller oil drop in *red-solid*, and larger oil drop in *blue-dashed*

Table 2 Simulation performance in cylindrical pore for different pore sizes. Time in CPU hours

	Iterations	Patches	Gridcells	Time
Pore radius 10 μm				
Uniform	1680	45	144000	1.59
AMR-1	6660	109	403632	15.86
AMR-2	25920	257	1322296	156.01
Pore radius 12 μm				
Uniform	2115	45	144000	2.02
AMR-1	8560	106	495336	23.59
AMR-2	34720	351	1733792	282.51
Pore radius 15 μm				
Uniform	2545	45	144000	2.48
AMR-1	11280	135	661336	41.60
AMR-2	49600	512	2619456	623.01

In the case of strong scaling, we consider a computational domain with $320 \times 320 \times 320$ coarse grid cells for the uniform-grid and AMR-1 simulations (see Fig. 13 (a)). As this domain size leads to memory overflow for AMR-2 with less than 512 processors, we use a computational domain with $320 \times 320 \times 160$ coarse grid cells for the strong scaling tests with AMR-2 (see Fig. 13 (b)). In both cases the geometry contains a total of 216 $(= 6^3)$ equally sized isolated fluid domains. Figure 14 shows the results from the strong scaling tests, and Tables 3 and 4 detail the speed-up and number of patches data. The tests show that AMR has a limited effect on the speed-up behaviour of the MLS-LVC method. In general, LVC-1 provides better scalability compared to LVC-2. Compared to the results on the uniform coarse grid, the scalability dips for AMR-1 but becomes equal or better for AMR-2. The number of patches per processor are nearly constant for the simulations on the uniform coarse grid, whereas in the AMR runs, the number of patches per processor decreases as the number of processors increases. The non-monotonic performance is due to a more significant decrease in the number of patches per processor for AMR-2 compared to AMR-1.

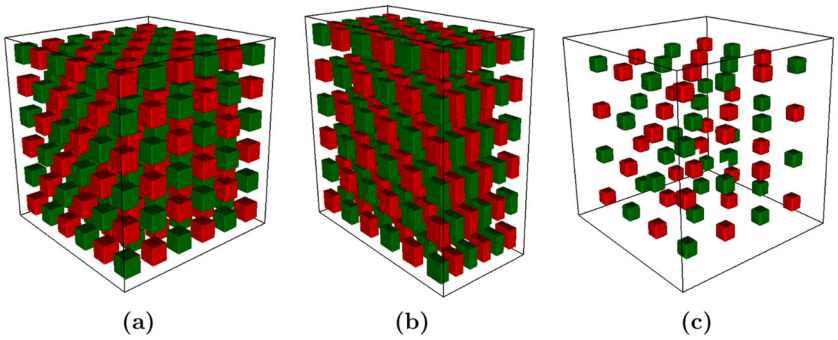

Fig. 13 Geometries with gas (*green*) and oil (*red*) configurations used in the scaling tests (the surrounding water phase is not shown): (a) Strong scaling tests for uniform grid and AMR-1 ($320 \times 320 \times 320$ coarse grid cells and $6^3 = 216$ isolated fluid domains), (b) strong scaling tests for AMR-2 ($320 \times 320 \times 160$ coarse grid cells and $6^3 = 216$ isolated domains), and (c) weak scaling tests (geometries of different sizes containing $4^3 = 64$ isolated fluid domains)

Such a decrease reduces the load on each processor, and it also reduces the communication requirement and root processor computations involved in identifying unique isolated domains in the LVC algorithm.

In case of weak scaling, we consider a cubic domain with $320 \times 320 \times 320$ coarse grid cells for 512 processors and $64(= 4^3)$ isolated fluid domains (see Fig. 13 (c)). Then we proportionately reduce the number of coarse grid cells and the number of processors by a factor of eight until we obtain $40 \times 40 \times 40$ coarse grid cells for one processor, while the number of isolated fluid domains (64) remains constant. Tables 5 to 7 lists the results of the weak scaling tests. For the smallest and least resolved geometry (using one processor), the first level of refinement in the AMR simulations covers the complete geometry instead of being limited to the region around the interfaces. This leads to a longer computational time than for the AMR simulations on the larger geometries with 8, 64, and 512 processors.

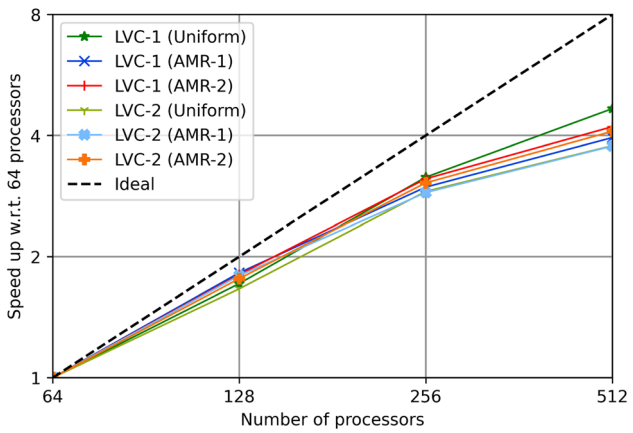

Fig. 14 Strong scaling results for uniform coarse grid, AMR-1, and AMR-2, with conservation of one phase (108 domains) and two phases (216 domains)

Another trend is that the computational time for the AMR runs is least for 64 processors, see Table 5. This trend results from two parameters: the number of patches per processor and the *total* number of patches. Like the strong scaling test, the reduction in the number of patches per processor can increase computational speed. However, an increase in the total number of patches increases communication time and computational time on the root processor. We illustrate the impact of the total number of patches by comparing computation times for AMR-2 runs with 64 and 512 processors (Table 5) against corresponding patch numbers in Table 6: Even though the number of patches per processor is significantly lower for the 512-processor runs than for the 64-processor runs, the 512-processor runs are slower as they form a nearly five times higher number of patches in total. These findings demonstrate that weak scaling results are sensitive to the total number of patches in the system, as reported by Gunney and Anderson [14]. The SAMRAI framework has been designed such that, for given grid refinement criteria, the scalability of the regridding process is of a linear order, but there is no control on the total number of patches that will be formed in the system.

An outlier in the weak scaling results is the 512-processor runs for AMR-1 which is slower than the 8-processor runs. In this case, the number of patches per processor has increased in comparison to the 64-processor case. The number of patches for 512 processors is approximately 24 times higher than for 8 processors. Based on the reasons mentioned above, there is a compounded decrease in performance which makes the overall computation slower. Table 7 highlights the reduction in the number of grid cells in AMR computations compared to a corresponding uniform fine grid. These values confirm the importance of using AMR to reduce the size of a large computational grid and using a large number of processors. Additional results from the scaling tests are provided in the supplementary information.

Table 3 Speed-up (w.r.t. 64 processors) in strong scaling test

Processors	LVC-1			LVC-2		
	Uniform	AMR-1	AMR-2	Uniform	AMR-1	AMR-2
64	1	1	1	1	1	1
128	1.715	1.824	1.805	1.664	1.795	1.764
256	3.150	2.976	3.121	2.903	2.885	3.049
512	4.663	3.951	4.205	3.772	3.764	4.096

Table 4 Number of patches per processor in strong scaling test

Processors	LVC-1			LVC-2		
	Uniform	AMR-1	AMR-2	Uniform	AMR-1	AMR-2
64	1.094	21.516	40.266	1.094	21.516	40.266
128	1.563	11.563	21.148	1.563	11.563	21.148
256	1.277	6.852	11.656	1.277	6.848	11.656
512	1.592	4.330	7.623	1.592	4.336	7.625

Table 5 Computation time in weak scaling test

Processors	LVC-1			LVC-2		
	Uniform	AMR-1	AMR-2	Uniform	AMR-1	AMR-2
1	11.923	118.777	398.994	12.462	128.688	432.552
8	12.771	47.366	146.219	13.548	51.350	158.142
64	17.571	33.106	90.371	19.130	36.517	101.186
512	19.817	71.723	147.780	21.908	76.648	156.282

Table 6 Number of patches per processor in weak scaling test

Processors	LVC-1			LVC-2		
	Uniform	AMR-1	AMR-2	Uniform	AMR-1	AMR-2
1	1	9	73	1	9	73
8	1	9	17	1	9	17
64	1.109	2.109	8.109	1.109	2.109	8.109
512	1.592	3.395	4.910	1.592	3.395	4.914

Table 7 Ratio of the numbers of grid cells in simulations with AMR and corresponding uniform grid in weak scaling test

Processors	LVC-1		LVC-2	
	AMR-1	AMR-2	AMR-1	AMR-2
1	1.125	0.357	1.125	0.357
8	0.341	0.134	0.341	0.134
64	0.216	0.068	0.216	0.068
512	0.166	0.041	0.166	0.041

The SAMRAI framework provides flexibility in patch generation by taking user inputs for the largest and smallest patch sizes during simulation. In our model, we use the computational domain size as the largest patch size and twice our numerical stencil size on each axis as the smallest patch size. This choice allows us to prevent a large number of small patches that can hinder computational speed. An important factor impacting the computational speed is the number of isolated ganglia in the system, which is apparent by the longer computation time of LVC-2 simulations compared to LVC-1 simulations.

4.4 3D LS/MLS-LVC simulations with AMR on sandstone pore geometries

We use 3D pore geometries of sandstone to explore the impact of AMR in simulations of capillary-controlled two-phase and three-phase displacements where disconnected fluid ganglia form. For this purpose, we have extracted subvolumes from a data set of segmented microtomography images of Castlegate sandstone with voxel length 5.6 μm [58]. We use LS-LVC method for two-phase simulations and MLS-LVC method for three-phase simulations. In both cases we compare the results produced with one level of refinement (AMR-1) against the results from the simulation on the coarse, uniform grid (represented by voxels). We set the MLS characteristic parameters for the simulations to $L^* = 5.6 \times 10^{-6}$ m, $\gamma^* = 10^{-4}$ Nm^{-1}, and $P^* = 17.9$ Pa.

The number of conserved domains doubles in LVC-2 simulations compared to LVC-1 simulations. For example, in the weak scaling test, there are 32 and 64 locally conserved domains in LVC-1 and LVC-2 simulations, respectively. In contrast, the number of patches per processor typically remains equal since we refine grid cells around all fluid/fluid interfaces in the pore space (see Tables 4 and 6). The results from the strong and weak scaling tests with 512 processors present four different numbers of conserved domains on the $320 \times 320 \times 320$ geometry. The results show that the factor by which the computational time increases with a growing number of locally conserved domains is lower than the growth in the number of such domains (see supplementary information for details).

4.4.1 Two-phase imbibition

We explore two-phase oil/water displacements during imbibition on a Castlegate rock geometry with volume $200L^* \times 200L^* \times 200L^*$ and porosity 22.7%. The rock is considered to be water-wet with contact angle $\theta_{ow} = 20°$, and interfacial tension is $\sigma_{ow} = 20 \times 10^{-3}$ Nm^{-1}. For this case, Helland et al. [17] performed LS simulations of primary drainage and imbibition without LVC, by increasing (drainage) and then decreasing (imbibition) the interface curvature $C_{ow} = \sigma_{ow}/P_{ow}$ stepwise after each equilibrium state achieved with equation (7). Here, we simulate imbibition with LVC applied to oil, for both AMR-1 and coarse uniform grid, starting from a reversal point on the primary drainage $P_{ow}(S_w)$-curve where $S_w = 0.13$ and $P_{ow} = 2.68$ kPa.

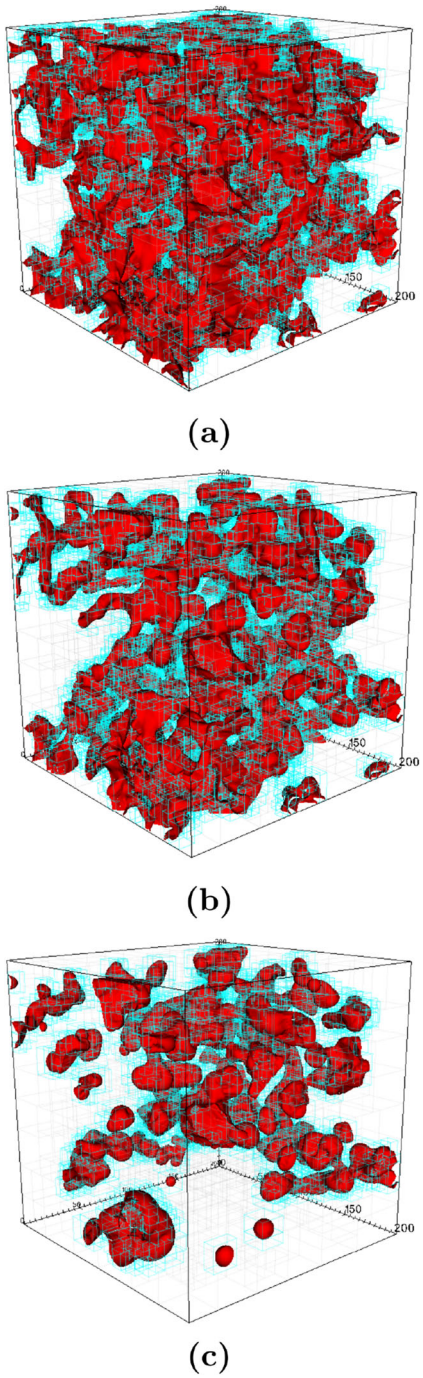

Fig. 15 Location of oil phase (*red*) during AMR simulation of imbibition in Castlegate sandstone for (a) $S_w = 0.13$ and $P_{ow} = 2.68$ kPa (initial state), (b) $S_w = 0.39$ and $P_{ow} = 1.07$ kPa, and (c) $S_w = 0.73$ and $P_{ow} = -0.54$ kPa (final state). Patches with finer grid are shown in *light blue*

We treat the top face of the rock as inlet boundary where we place a water-wet porous plate with thickness of two coarse grid cells, which allows water to imbibe into the sample. Oil withdrawal occurs only through bottom face (outlet) while the side walls are sealed. The level set functions use reflective conditions at all boundaries. We terminate the imbibition simulations when all the oil is capillary trapped as disconnected ganglia. The coarse, uniform simulation uses grid spacing $\Delta x = 1$, while periodic reinitialization of ϕ occurs every fifth LS iteration. The tolerance value in equation (18) is $c = 1.5 \times 10^{-3}$. Based on the discussions in section 4.1, we let the AMR simulation perform reinitializations every 20th LS iteration and set $c = 7.5 \times 10^{-4}$. We tag cells for refinement that satisfy $|\phi| < \Delta x, \psi > 0$ and regrid every 2000 iterations.

Figure 15 shows the location of oil phase and finer grid patches at three different capillary pressures during the AMR simulation. A video of the AMR simulation is provided as supplementary information. During the initial stages of imbibition, the oil phase is continuous while water layers in crevices along the pore walls grow in thickness [17]. Even-

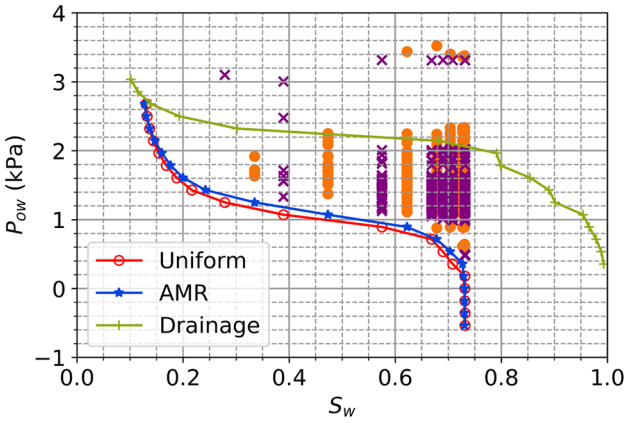

Fig. 17 Capillary pressure between continuous oil and water as a function of water saturation for primary drainage (coarse, uniform grid) and imbibition (coarse, uniform grid and AMR-1). Capillary pressures of disconnected oil ganglia that form during imbibition for AMR-1 (*orange circles*) and coarse, uniform grid (*purple crosses*) are also shown

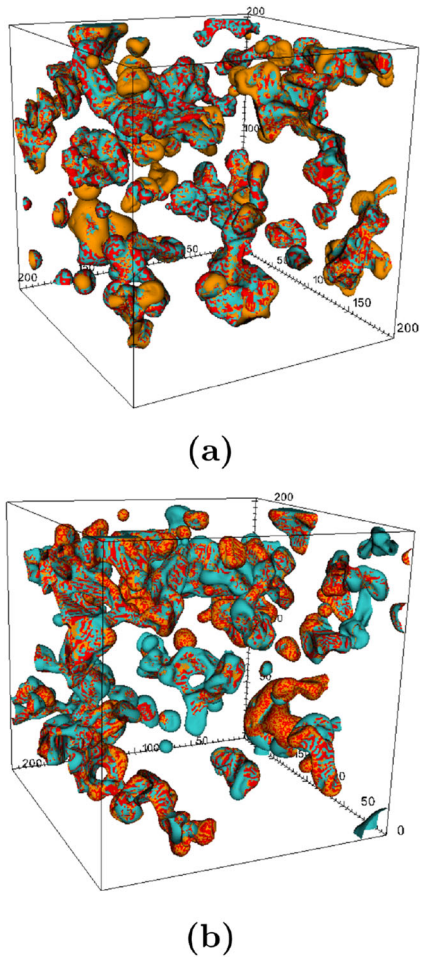

Fig. 16 Residual oil configurations after imbibition in Castlegate sandstone from AMR-1 and uniform, coarse-grid simulations. Images (a) and (b) show oil-phase locations from two diagonally opposite views for AMR-1 (*orange*), coarse uniform grid (*cyan*), and the common locations from both simulations (*red*)

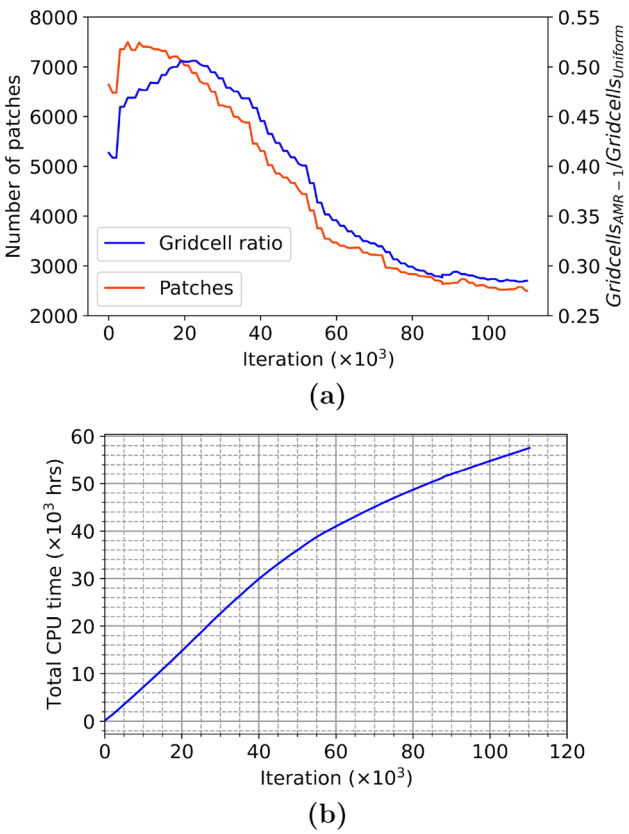

Fig. 18 Model performance during the AMR-1 simulation of two-phase imbibition on Castlegate sandstone. (a) Evolution of the number of patches and the ratio of the number of grid cells between AMR-1 and a corresponding uniform fine grid. (b) Total CPU usage curve

tually, water snaps off oil and invades narrow pore throats while disconnected oil ganglia form and relax to stable states in nearby pore openings. Figure 16 shows that the location of oil ganglia changes slightly due to refinement. However, the residual oil saturation after imbibition is $S_{or} = 0.27$ for both the uniform coarse grid and AMR-1 runs.

Figure 17 reveals that the capillary pressure curves from the two simulations are almost identical and that most oil ganglia form when $S_w > 0.4$. An important point of note is that interface relaxation creates stable states in which the capillary pressures of oil ganglia differ from the prescribed capillary pressure between the continuous phases at the incident of ganglia disconnection. The oil ganglia create a wide range of capillary pressures, in which the majority of the data are located between the continuous $P_{ow}(S_w)$-curves for drainage and imbibition. Such behaviour are also observed in pore-scale experiments based on analysis of two-phase micro-CT images [12, 20, 34]. The AMR simulation differentiates the ganglia capillary pressures more distinctly as it creates a wider spread among the majority of the data than the coarse simulation.

Figure 18 (a) shows that the number of patches remains nearly constant while the refined area increases during the initial stages of the imbibition. The increase in refined area is

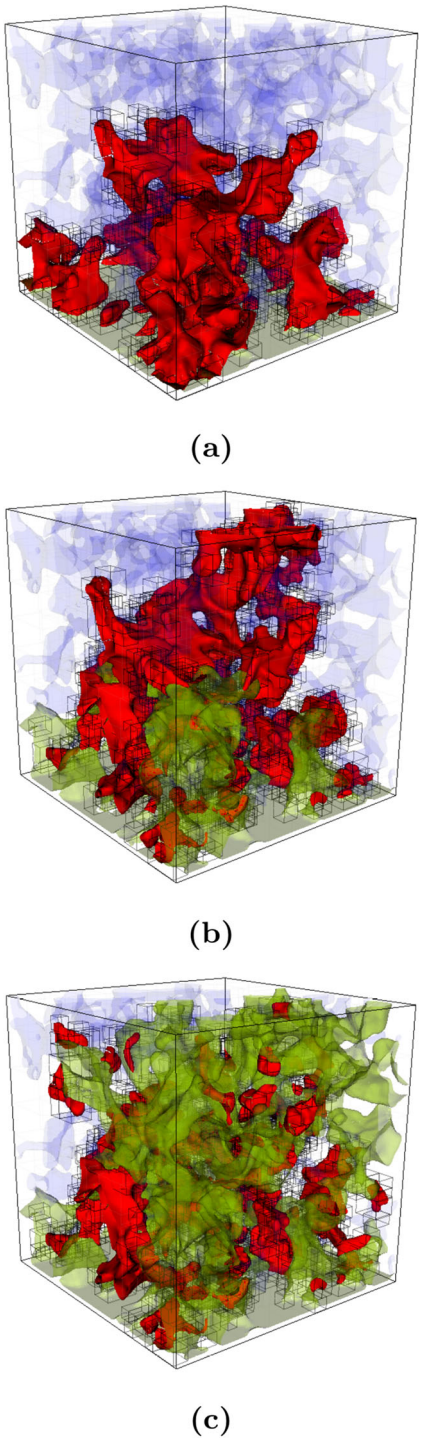

Fig. 19 Three-phase fluid configurations of water (*blue*), oil (*red*) and gas (*green*) from AMR simulation of gas invasion in Castlegate sandstone for (a) $P_{gw} = 1.9$ kPa (initial state), (b) $P_{gw} = 2.77$ kPa, and (c) $P_{gw} = 3.44$ kPa (final state). Patches with fine grid are shown in *black*

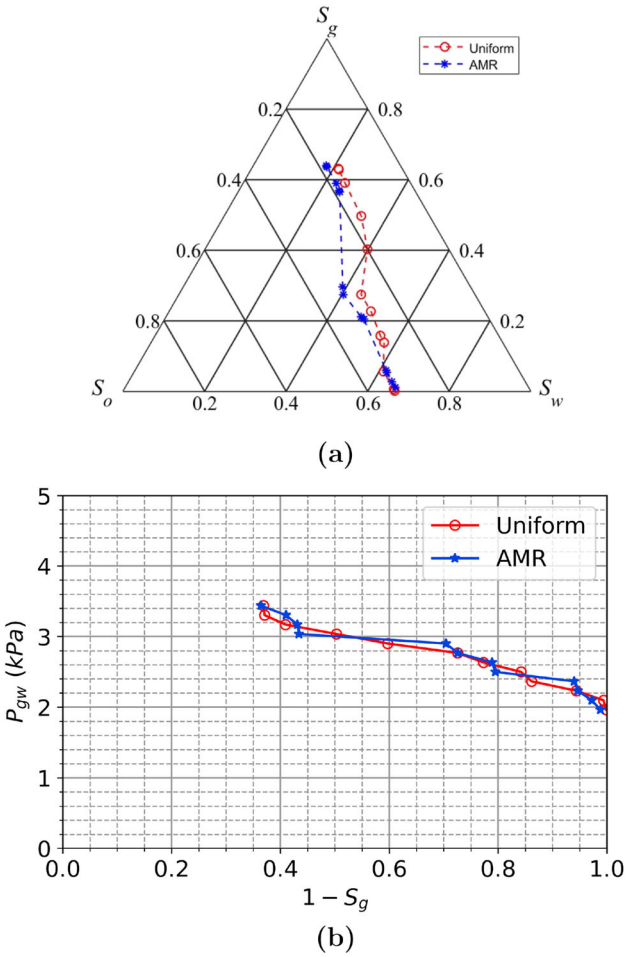

Fig. 20 Simulation results from gas invasion on Castlegate sandstone using AMR and coarse, uniform grid: (a) Saturation paths, and (b) gas-water capillary pressure curves

due to the swelling of water layers which increases interfacial area between oil and water [11, 17]. At later stages, both the number of grid cells and the number of patches in the computational geometry decrease as oil is displaced. This speeds up the simulation and results in decreased slope of the total CPU usage curve, as shown in Fig. 18 (b). The total CPU time is the product of simulation time and number of computing cores.

4.4.2 Three-phase gas invasion

To investigate the impact of AMR on simulations of three-phase displacements we use a Castlegate sandstone geometry with size $128L^* \times 128L^* \times 128L^*$ and porosity 20.8%. We consider a water-wet state with a consistent set of contact angles given by $\theta_{go} = 23.1°, \theta_{gw} = 32°$ and $\theta_{ow} = 40°$, and a fluid system with interfacial tensions $\sigma_{go} = 11 \times 10^{-3}$ Nm^{-1}, $\sigma_{gw} = 30 \times 10^{-3}$ Nm^{-1} and $\sigma_{ow} = 20 \times 10^{-3}$ Nm^{-1}, representing a slightly non-spreading oil. For this case Jettestuen et al. [28] performed a coarse-grid simulation ($\Delta x = 1$) of gas invasion after primary drainage using MLS-LVC method with conservation of the oil phase (LVC-1).

In the present study, we repeat the simulation of gas invasion with one level of grid refinement (AMR-1). The inlet is located at the bottom face of the cubic domain where we add a two-voxel thick layer of pore space filled with invading phase. With respect to LVC, the side walls are sealed and top face is outlet. The level set functions use linear extrapolation at inlet and reflective conditions at the other faces as boundary conditions. We initiate gas invasion from a high water saturation ($S_w = 0.67$) before oil breakthrough occurs in primary drainage. Thus, all the oil is disconnected and subjected to LVC in the beginning of the three-phase displacement. We simulate gas invasion by increasing gas pressure stepwise while the phase pressures of continuous oil and water are held constant. The simulations are terminated when the oil saturation no longer decreases. The coarse-grid simulation performed reinitialization of level set functions ϕ_α every 10th MLS iteration and used $c = 1.5 \times 10^{-3}$ as tolerance value. As we are mainly interested in the behaviour of the disconnected phase in the AMR simulation, we only refine cells around oil boundaries according to $|\phi_o| < \Delta x$ and $\psi > 0$. Gas/water interfaces will then be located on the coarse grid, so we maintain a reinitialization frequency of once every 10 iterations while we use a reduced tolerance value $c = 7.5 \times 10^{-4}$ as discussed previously.

Figure 19 shows three-phase fluid configurations and patches with finer grid for three different gas-water capillary pressures during the AMR simulation. A video is provided as supplementary information. In the initial stages, gas invasion occurs by double displacements in which gas displaces several oil slugs that displace water. After oil reaches the top boundary the dominating mechanisms are direct gas-oil

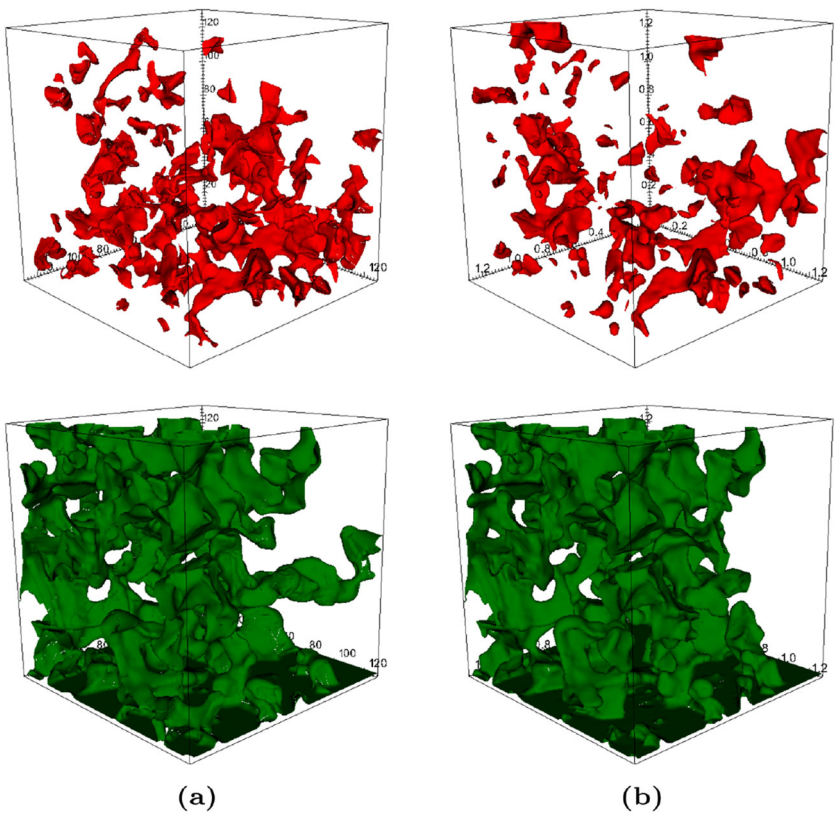

Fig. 21 Oil (*top*) and gas (*bottom*) configurations at the final state ($P_{gw} = 3.44$ kPa) of gas invasion in Castlegate sandstone from three-phase simulations with (a) AMR-1 (this work), and (b) uniform, coarse grid [28]

displacements (that reduces oil saturation) and double displacements. During this process, oil ganglia detach from both continuous oil and larger oil slugs of which some of the ganglia continue their displacement while others get trapped. The trapped oil phase occupies mostly intermediate-sized pores and pore corners as intermediate-wet layers. The last stage of the simulation is dominated by direct gas-water displacements.

Figure 20 compares the three-phase saturation paths and gas-water capillary pressure curves from the AMR-1 and coarse-grid simulations. Typically, capillary pressure curves for drainage on small pore-scale samples display a staircase trend (for pressure-controlled invasion) that consists of intervals with irreversible saturation jumps between equilibrium states (e.g., Haines jumps) connected by intervals with smooth reversible saturation changes [7, 17, 19]. Although the level of the two capillary pressure curves are similar, the AMR simulation shows a more pronounced staircase trend than the coarse simulation. The saturation paths from the two simulations differ more. The deviation at intermediate gas saturations occurs because the AMR simulation creates a different route for the gas and oil movement to the top face, leading to gas breakthrough at a higher gas saturation. Figure 21 shows that this leads to significantly different residual oil configurations in which the AMR simulation traps more oil in the lower half of the geometry than the coarse simulation. The residual oil saturations from the simulations are $S_{or} = 0.156$ (coarse grid) and $S_{or} = 0.182$ (AMR). Further, the residual oil configuration from the AMR simulation is less fragmented as it contains fewer but more elongated oil ganglia with shapes that are more reminiscent of intermediate-wet oil layers from three-phase pore-scale experiments in porous rocks [e.g., 53]. Figure 21 shows that there is also some differences in the final gas configurations from the two simulations even though the final gas saturations are about the same.

Figure 22 illustrates the performance of the MLS-LVC model during the AMR simulation. The numbers of patches and fine grid cells in the geometry increases as gas displaces and traps oil. After gas breakthrough, these numbers reach a maximum and experience a small decline when direct gas-water displacements start dominating. The slope of the total CPU usage curve responds to the change in the numbers of patches and grid cells during the simulation.

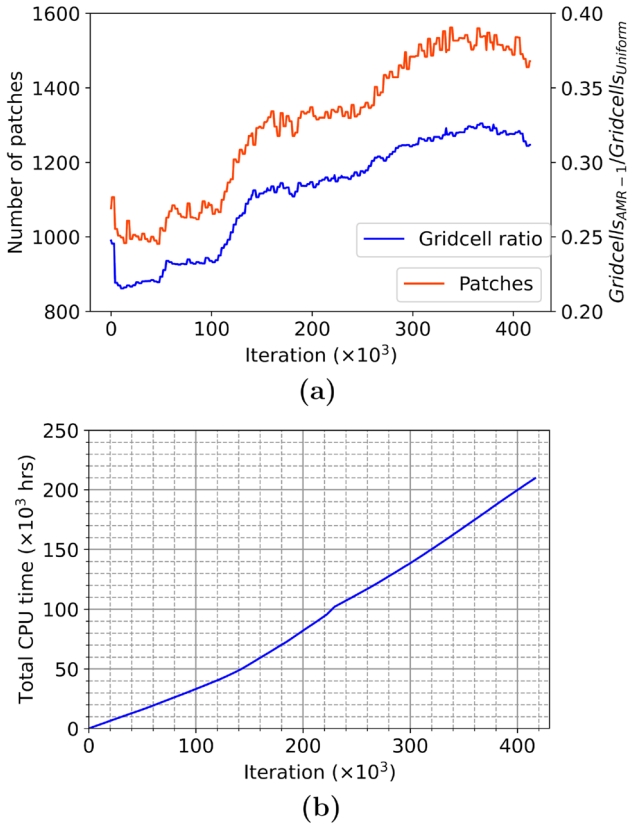

Fig. 22 Model performance during three-phase AMR-1 simulation on Castlegate sandstone. (a) Evolution of the number of patches and the number of grid cells (normalised by the number of grid cells in a corresponding uniform fine grid). (b) Total CPU usage curve

5 Conclusion

This work has added structured, adaptive mesh refinement capabilities to locally, conservative level set methods for simulating two-phase and three-phase capillary-controlled displacement with disconnected ganglia at the pore scale. The AMR implementation of these LS/MLS-LVC methods was carried out within the patch-based software framework SAMRAI. We created new integer operators that are essential in preserving the local volume of isolated domains. The implementation also improved memory requirements and computations by preventing multiple identifications of any particular set of neighbouring domains. We validated the MLS-LVC method with AMR against analytical three-phase configurations in idealized pores, where we treated the oil layers as distinct domains with their volumes preserved. The simulation results provide pressure and volume convergence while tracking more than one isolated domain. Specifically, we obtain second-order volume convergence with respect to grid spacing. The computational performance of the model during simulation underscores the strength of AMR compared to a corresponding uniform fine grid. Further, the results from simulations with AMR and a corresponding fine uniform grid coincide, which shows that AMR simulations provide similar accuracy with faster computations. The AMR implementation shows good performance in both strong and weak scaling tests. The scaling tests also point out the impor-

tance of the number of patches per processor as well as the total number of patches in gaining computational efficiency.

Finally,t we investigated the impact of AMR on LS/MLS-LVC simulations of two-phase imbibition and three-phase gas invasion with formation of disconnected oil ganglia on 3D pore structures of water-wet Castlegate sandstone. Compared with coarse-grid simulations, the most significant impact of AMR was obtained for three-phase displacements with respect to the behaviour of the conserved, intermediate-wet phase. Differences in the shapes and spatial distribution of oil ganglia lead to different residual oil saturations and saturation paths. These results suggest that AMR is an important technology for obtaining desired accuracy and efficiency in pore-scale simulations of three-phase displacements. This will in turn, improve calculations of constitutive relationships, such as three-phase capillary pressure-saturation curves, which are required for large-scale multiphase flow simulations in porous media.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10596-023-10219-0.

Acknowledgements Financial support was provided by the Research Council of Norway through Petromaks2 project "Foam dynamics in the presence of oil during multiphase flow in porous rock (grant no. 294886)". The computations on 3D geometry were performed on resources provided by UNINETT Sigma2, the National Infrastructure for High Performance Computing and Data Storage in Norway.

Author Contributions Deepak Singh: Conceptualization, Investigation, Methodology, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. Helmer Andrè Friis: Conceptualization, Methodology, Software, Writing - review & editing. Espen Jettestuen: Conceptualization, Methodology, Software, Writing - review & editing. Johan Olav Helland: Conceptualization, Funding acquisition, Methodology, Software, Supervision, Validation, Visualization, Writing - review & editing.

Funding Open access funding provided by University of Stavanger & Stavanger University Hospital. The financial support was provided by the Research Council of Norway through Petromaks2 project "Foam dynamics in the presence of oil during multiphase flow in porous rock (grant no. 294886)".

Availability of data and materials The datasets generated during and/or analysed during the current study are available in the Figshare repository at: https://doi.org/10.6084/m9.figshare.21202469.v1.

Code Availability Available on request.

Declarations

Competing interests The authors declare no conflict of interest/competing interests.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Appendix A: Analysis of isolated domain identification methodology

In our work, isolated domains are identified in two steps: (1) Identify isolated regions on a patch, and (2) generate unique global tags for isolated domains. The first step provides isolated regions on a patch with patch-level unique tags on each processor. The second step uses the processor rank and patch-level unique tags to generate unique global tags for isolated domains. When an isolated domain lies on more than two patches on a processor, we identify it with a different tag on each patch. Identifying the same domain by multiple tags on a processor leads to more data communication with the root processor. A higher communication load negatively impacts computational performance. An intuitive way to reduce the data communication is to split the second step into two parts: (i) processor level unique identification and (ii) unique global identification from a reduced list aggregated from the processors. We refer to the two-step identification method as serial identification and the three-step identification method as parallel identification.

We carry out three sets of strong scaling tests with both serial and parallel identification of isolated domains following the procedure described in Section 4.3. The tests are carried out on a cubic geometry with two phases (using LS-LVC model) such that one phase is completely continuous and the other is completely disjointed (see Fig. 23). We refine grid cells around all interfaces ($|\phi| < \Delta x$). This kind of geometry provides the highest level of overlaps between patches with different levels of refinement that identify the same isolated region. Therefore, it is highly suitable for an algorithm that gives processor-level unique tags to isolated domains before providing them with globally unique tags in the root processor. Table 8 shows the data from these tests. We can see that the tagging criteria provides a large number of refined grid cells because the number of grid cells between different processors remains the same while the number of

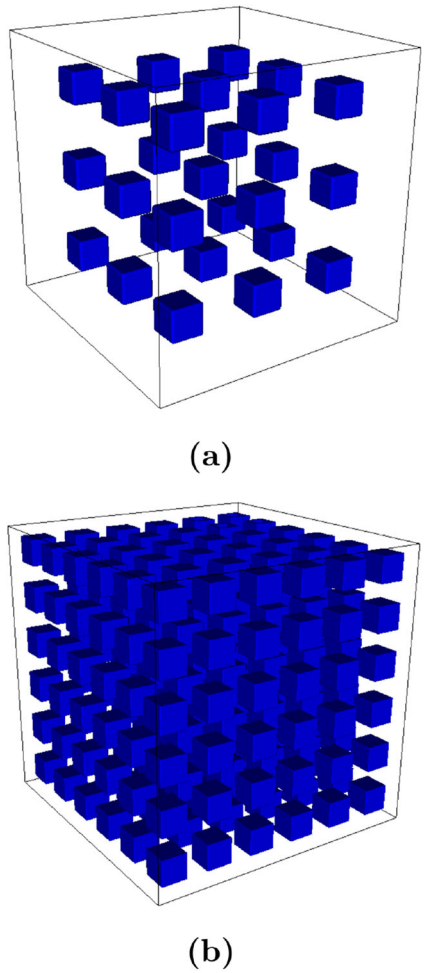

Fig. 23 Geometry used for strong scaling tests with isolated domains of one phase (in *blue*) surrounded by a continuous phase that occupies the remaining part of the computational domain. (a) Geometry used for 16, 32, and 64 processors contains $50 \times 50 \times 50$ coarse grid cells. (b) Geometry used for 128, 256 and 512 processors contains $160 \times 160 \times 160$ coarse grid cells

patches increases. This is unlike the result seen in Section 4.3 with three phases.

Although not intuitive, Table 8 shows that the parallel identification does not provide any gains. Instead, it is generally slightly slower than serial identification. The drop in speed can be attributed to three factors. First parallel identification requires an extra iteration of reading and writing data on a patch along with a processor level run for the unique labelling of domains. These additional executions increase the time spent on each processor before unique global labelling. Second, over the years, there has been a continuous reduction in MPI data communication time between processors, and nowadays, it is of linear order [14]. This reduction diminishes the possible gain from a reduction in computational time from parallel processing. Finally, in our model we utilise the patch generation and load balancing

Table 8 Performance comparison between serial and parallel identification of isolated domains for a given refinement ratio, R, and number of processors, P.

P	N_d	R	E	Pat	$t_{(Serial)}$	$t_{(Parallel)}$	S
16	27	2	0.183	317	13.092	13.415	0.976
32	27	2	0.183	400	8.688	8.924	0.974
64	27	2	0.183	504	5.520	5.636	0.979
16	27	4	0.031	499	23.931	24.453	0.979
32	27	4	0.031	587	13.951	14.239	0.980
64	27	4	0.031	768	9.411	9.587	0.982
128	216	2	0.208	5378	1496.570	1467.503	1.020
256	216	2	0.208	5562	874.735	849.366	1.030
512	216	2	0.208	6405	610.009	654.028	0.933

The table lists the time for computation, t, the speed-up with parallel identification, S, the reduction in the number of elements compared to a uniform grid, E, and the number of patches, Pat, for a particular number of isolated domains, N_d

capabilities of the SAMRAI framework. SAMRAI provides adequate load balancing by striving to put nearby patches into the same processor. However, using the framework reduces our control in forcing adequate overlap onto the patches in a processor to ensure that parallel identification improves computation time.

References

1. Adalsteinsson, D., Sethian, J.: Transport and diffusion of material quantities on propagating interfaces via level set methods. Journal of Computational Physics **185**(1), 271–288 (2003). https://doi.org/10.1016/s0021-9991(02)00057-8
2. Al-Dhahli, A., van Dijke, M.I.J., Geiger, S.: Accurate modelling of pore-scale films and layers for three-phase flow processes in clastic and carbonate rocks with arbitrary wettability. Transport in Porous Media **98**, 259–286 (2013). https://doi.org/10.1007/s11242-013-0144-z
3. Anderson, R.W., Arrighi, W.J., Elliott, N.S., et al. (2013) SAMRAI concepts and software design. Tech. Rep. LLNL-SM-617092-DRAFT, Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory, Livermore, CA (Available at https://computing.llnl.gov/sites/default/files/SAMRAI-Concepts_SoftwareDesign.pdf)
4. Berger, M., Colella, P.: Local adaptive mesh refinement for shock hydrodynamics. Journal of Computational Physics **82**(1), 64–84 (1989). https://doi.org/10.1016/0021-9991(89)90035-1
5. de Chalendar, J.A., Garing, C., Benson, S.M.: Pore-scale modelling of ostwald ripening. Journal of Fluid Mechanics **835**, 363–392 (2017). https://doi.org/10.1017/jfm.2017.720
6. Childs H, Brugger E, Whitlock B, Meredith J, Ahern S, Pugmire D, Biagas K, Miller M, Harrison C, Weber GH, Krishnan H, Fogal T, Sanderson A, Garth C, Bethel EW, Camp D, Durant ORM, Favre JM, and Navrátil P (2012) VisIt: An end-user tool for visualizing and analyzing very large data. In: High Performance Visualization–Enabling Extreme-Scale Scientific Insight, 16. Chapman and Hall/CRC, p 358–396. https://doi.org/10.1201/b12985 (Available for download at: https://visit-dav.github.io/visit-website/releases-as-tables/)

7. Cueto-Felgueroso, L., Juanes, R.: A discrete-domain description of multiphase flow in porous media: Rugged energy landscapes and the origin of hysteresis. Geophysical Research Letters **43**, 1615–1622 (2016). https://doi.org/10.1002/2015GL067015
8. van Dijke, M.I.J., Sorbie, K.S.: The relation between interfacial tensions and wettability in three-phase systems: Consequences for pore occupancy and relative permeability. Journal of Petroleum Science and Engineering **33**(1–3), 39–48 (2002). https://doi.org/10.1016/S0920-4105(01)00174-7
9. Enright, D., Fedkiw, R., Ferziger, J., et al.: A hybrid particle level set method for improved interface capturing. Journal of Computational Physics **183**(1), 83–116 (2002). https://doi.org/10.1006/jcph.2002.7166
10. Favino, M., Hunziker, J., Caspari, E., et al.: Fully-automated adaptive mesh refinement for media embedding complex heterogeneities: application to poroelastic fluid pressure diffusion. Computational Geosciences **24**(3), 1101–1120 (2020). https://doi.org/10.1007/s10596-019-09928-2
11. Friis, H.A., Pedersen, J., Jettestuen, E., et al.: Pore-scale level set simulations of capillary-controlled displacement with adaptive mesh refinement. Transport in Porous Media **128**(1), 123–151 (2019). https://doi.org/10.1007/s11242-019-01238-6
12. Garing, C., de Chalendar, J.A., Voltolini, M., et al.: Pore-scale capillary pressure analysis using multi-scale x-ray micromotography. Advances in Water Resources **104**, 223–241 (2017). https://doi.org/10.1016/j.advwatres.2017.04.006
13. Ge, Z., Loiseau, J.C., Tammisola, O., et al.: An efficient mass-preserving interface-correction level set/ghost fluid method for droplet suspensions under depletion forces. Journal of Computational Physics **353**, 435–459 (2018). https://doi.org/10.1016/j.jcp.2017.10.046
14. Gunney, B.T., Anderson, R.W.: Advances in patch-based adaptive mesh refinement scalability. Journal of Parallel and Distributed Computing **89**, 65–84 (2016). https://doi.org/10.1016/j.jpdc.2015.11.005
15. Hazlett, R.D.: Simulation of capillary-dominated displacements in microtomographic images of reservoir rocks. Transport in Porous Media **20**, 21–35 (1995). https://doi.org/10.1007/BF00616924
16. Helland, J.O., Jettestuen, E.: Mechanisms for trapping and mobilization of residual fluids during capillary-dominated three-phase flow in porous rock. Water Resources Research **52**, 5376–5392 (2016). https://doi.org/10.1002/2016WR018912
17. Helland, J.O., Friis, H.A., Jettestuen, E., et al.: Footprints of spontaneous fluid redistribution on capillary pressure in porous rock. Geophysical Research Letters **44**(10), 4933–4943 (2017). https://doi.org/10.1002/2017gl073442
18. Helland, J.O., Pedersen, J., Friis, H.A., et al.: A multiphase level set approach to motion of disconnected fluid ganglia during capillary-dominated three-phase flow in porous media: Numerical validation and applications. Chemical Engineering Science **203**, 138–162 (2019). https://doi.org/10.1016/j.ces.2019.03.060
19. Helland JO, Jettestuen E, Friis HA (2021) A discrete-domain approach to three-phase hysteresis in porous media. Water Resources Research 57:e2021WR029–560. https://doi.org/10.1029/2021WR029560
20. Herring, A., Middleton, J., Walsh, R., et al.: Flow rate impacts on capillary pressure and interface curvature of connected and disconnected fluid phases during multiphase flow in sandstone. Advances in Water Resources **107**, 460–469 (2017). https://doi.org/10.1016/j.advwatres.2017.05.011
21. Hilfer, R., Øren, P.E.: Dimensional analysis of pore scale and field scale immiscible displacement. Transport in Porous Media **22**(1), 53–72 (1996). https://doi.org/10.1007/bf00974311
22. Hilpert, M., Miller, C.T.: Pore-morphology-based simulation of drainage in totally wetting porous media. Advances in Water Resources **24**, 243–255 (2001). https://doi.org/10.1016/S0309-1708(00)00056-7
23. Hirt, C.W., Nichols, B.D.: Volume of fluid (VOF) method for the dynamics of free boundaries. Journal of Computational Physics **39**, 201–225 (1981). https://doi.org/10.1016/0021-9991(81)90145-5
24. Hornung RD, Kohn SR (2002) Managing application complexity in the SAMRAI object-oriented framework. Concurrency and Computation: Practice and Experience 14:347–368. (SAMRAI is available at https://github.com/llnl/samrai)
25. Hornung, R.D., Wissink, A.M., Kohn, S.R.: Managing complex data and geometry in parallel structured AMR applications. Engineering with Computers **22**, 181–195 (2006). https://doi.org/10.1007/s00366-006-0038-6
26. Hui, M.H., Blunt, M.J.: Effects of wettability on three-phase flow in porous media. The Journal of Physical Chemistry B **104**(16), 3833–3845 (2000). https://doi.org/10.1021/jp9933222
27. Jettestuen, E., Helland, J.O., Prodanović, M.: A level set method for simulating capillary-controlled displacements at the pore scale with nonzero contact angles. Water Resources Research **49**, 4645–4661 (2013). https://doi.org/10.1002/wrcr.20334
28. Jettestuen, E., Friis, H.A., Helland, J.O.: A locally conservative multiphase level set method for capillary-controlled displacements in porous media. Journal of Computational Physics **428**, 109–965 (2021). https://doi.org/10.1016/j.jcp.2020.109965
29. Jiang, F., Tsuji, T.: Estimation of three-phase relative permeability by simulating fluid dynamics directly on rock-microstructure images. Water Resources Research **53**(1), 11–32 (2017). https://doi.org/10.1002/2016WR019098
30. van Kats, F.M., Egberts, P.J.P.: Simulation of three-phase displacement mechanisms using a 2D lattice-Boltzmann model. Transport in Porous Media **37**, 55–68 (1999). https://doi.org/10.1023/A:1006502831641
31. Kim, J., Lowengrub, J.: Phase field modelling and simulation of three-phase flows. Interfaces and Free Boundaries **7**, 435–466 (2005). https://doi.org/10.4171/IFB/132
32. Krimi, A., Rezoug, M., Khelladi, S., et al.: Smoothed particle hydrodynamics: A consistent model for interfacial multiphase fluid flow simulations. Journal of Computational Physics **358**, 53–87 (2018). https://doi.org/10.1016/j.jcp.2017.12.006
33. Li, L., Zhu, J., Zhang, Y.: Absolutely convergent fixed-point fast sweeping WENO methods for steady state of hyperbolic conservation laws. Journal of Computational Physics **443**, 110516 (2021). https://doi.org/10.1016/j.jcp.2021.110516
34. Li, T., Schlüter, S., Dragila, M.I., et al.: An improved method for estimating capillary pressure from 3D microtomography images and its application to the study of disconnected nonwetting phase. Advances in Water Resources **114**, 249–260 (2018). https://doi.org/10.1016/j.advwatres.2018.02.012
35. Liang, H., Xu, J., Chen, J., et al.: Lattice boltzmann modeling of wall-bounded ternary fluid flows. Applied Mathematical Modelling **73**, 487–513 (2019). https://doi.org/10.1016/j.apm.2019.03.009
36. Losasso, F., Shinar, T., Selle, A., et al.: Multiple interacting liquids. ACM Transactions on Graphics **25**(3), 812–819 (2006). https://doi.org/10.1145/1141911.1141960
37. Luo, K., Shao, C., Yang, Y., et al.: A mass conserving level set method for detailed numerical simulation of liquid atomization. Journal of Computational Physics **298**(1), 495–519 (2015). https://doi.org/10.1016/j.jcp.2015.06.009
38. Merriman, B., Bence, J.K., Osher, S.J.: Motion of multiple junctions: A level set approach. Journal of Computational Physics **112**, 334–363 (1994). https://doi.org/10.1006/jcph.1994.1105
39. Mohammadmoradi, P., Kantzas, A.: Toward direct pore-scale modeling of three-phase displacements. Advances in Water Resources **110**, 120–135 (2017). https://doi.org/10.1016/j.advwatres.2017.10.010

40. Nourgaliev, R.R., Theofanous, T.G.: High-fidelity interface tracking in compressible flows: Unlimited anchored adaptive level set. Journal of Computational Physics **224**, 836–866 (2007). https://doi.org/10.1016/j.jcp.2006.10.031
41. Olsson, E., Kreiss, G.: A conservative level set method for two phase flow. Journal of Computational Physics **210**, 225–246 (2005). https://doi.org/10.1016/j.jcp.2005.04.007
42. Øren, P., Pinczewski, W.: Fluid distribution and pore-scale displacement mechanisms in drainage dominated three-phase flow. Transport in Porous Media **20**, 105–133 (1995). https://doi.org/10.1007/BF00616927
43. Øren, P.E., Bakke, S., Arntzen, O.J.: Extending predictive capabilities to network models. SPE Journal **3**, 324–336 (1998). https://doi.org/10.2118/52052-PA
44. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer, New York, (2003). https://doi.org/10.1007/b98879
45. Piri, M., Blunt, M.: Three-dimensional mixed-wet random pore-scale network modeling of two- and three-phase flow in porous media. I. Model description. Physical Review E **71**, 026–301 (2005). https://doi.org/10.1103/PhysRevE.71.026301
46. Popinet, S.: A quadtree-adaptive multigrid solver for the serre–green–naghdi equations. Journal of Computational Physics **302**, 336–358 (2015). https://doi.org/10.1016/j.jcp.2015.09.009
47. Prodanović, M., Bryant, S.: A level set method for determining critical curvatures for drainage and imbibition. Journal of Colloid and Interface Science **304**, 442–458 (2006). https://doi.org/10.1016/j.jcis.2006.08.048
48. Raeini, A.Q., Blunt, M.J., Bijeljic, B.: Direct simulations of two-phase flow on micro-CT images of porous media and upscaling of pore-scale forces. Advances in Water Resources **74**, 116–126 (2014). https://doi.org/10.1016/j.advwatres.2014.08.012
49. Ramstad, T., Idowu, N., Nardi, C., et al.: Relative permeability calculations from two-phase flow simulations directly on digital images of porous rocks. Transport in Porous Media **94**, 487–504 (2012). https://doi.org/10.1007/s11242-011-9877-8
50. Rücker, M., Berg, S., Armstrong, R.T., et al.: From connected pathway flow to ganglion dynamics. Geophysical Research Letters **42**, 3888–3894 (2015). https://doi.org/10.1002/2015GL064007
51. Ruuth, S.J.: A diffusion-generated approach to multiphase motion. Journal of Computational Physics **145**, 166–192 (1998). https://doi.org/10.1006/jcph.1998.6028
52. Saye RI, Sethian JA (2011) The voronoi implicit interface method for computing multiphase physics. Proceedings of the National Academy of Sciences 108(49):19,498–19,503. https://doi.org/10.1073/pnas.1111557108
53. Scanziani, A., Singh, K., Bultreys, T., et al.: In situ characterization of immiscible three-phase flow at the pore scale for a water-wet carbonate rock. Advances in Water Resources **121**, 446–455 (2018). https://doi.org/10.1016/j.advwatres.2018.09.010
54. Scanziani, A., Singh, K., Menke, H., et al.: Dynamics of enhanced gas trapping applied to CO_2 storage in the presence of oil using synchrotron X-ray micro tomography. Applied Energy **259**, 114–136 (2020). https://doi.org/10.1016/j.apenergy.2019.114136
55. Sethian, J.A.: Level set methods and fast marching methods, 2nd edn. Cambridge University Press (1999)
56. Shams, M., Raeini, A.Q., Blunt, M.J., et al.: A numerical model of two-phase flow at the micro-scale using the volume-of-fluid method. Journal of Computational Physics **357**, 159–182 (2018). https://doi.org/10.1016/j.jcp.2017.12.027
57. Shan, X., Chen, H.: Lattice Boltzmann model for simulating flows with multiple phases and components. Physical Review E **47**(3), 1815–1819 (1993). https://doi.org/10.1103/PhysRevE.47.1815
58. Sheppard A, Schroeder-Turk G (2015) Network generation comparison forum. https://doi.org/10.17612/P7059V
59. Shi, Y., Tang, G.H., Wang, Y.: Simulation of three-component fluid flows using the multiphase lattice Boltzmann flux solver. Journal of Computational Physics **314**, 228–243 (2016). https://doi.org/10.1016/j.jcp.2016.03.011
60. Sun, Y., Beckermann, C.: Sharp interface tracking using the phase-field equation. Journal of Computational Physics **220**, 626–653 (2007). https://doi.org/10.1016/j.jcp.2006.05.025
61. Sussman, M., Puckett, E.: A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. Journal of Computational Physics **162**, 301–337 (2000). https://doi.org/10.1006/jcph.2000.6537
62. Svadlenka, K., Ginder, E., Omata, S.: A variational method for multiphase volume-preserving interface motions. Journal of Computational and Applied Mathematics **257**, 157–179 (2014). https://doi.org/10.1016/j.cam.2013.08.027
63. Tabarraei, A., Sukumar, N.: Adaptive computations on conforming quadtree meshes. Finite Elements in Analysis and Design **41**(7–8), 686–702 (2005). https://doi.org/10.1016/j.finel.2004.08.002
64. Tanino Y, Blunt MJ (2012) Capillary trapping in sandstones and carbonates: Dependence on pore structure. Water Resources Research 48(8). https://doi.org/10.1029/2011wr011712
65. Tartakovsky, A.M., Meakin, P.: Modeling of surface tension and contact angles with smoothed particle hydrodynamics. Physical Review E **72**(2), 026–301 (2005). https://doi.org/10.1103/PhysRevE.72.026301
66. Tomutsa, L., Silin, D., Radmilovic, V.: Analysis of chalk petrophysical properties by means of submicron-scale pore imaging and modeling. SPE Reservoir Evaluation & Engineering **10**(3), 285–293 (2007). https://doi.org/10.2118/99558-PA
67. Wei, B., Huang, H., Hou, J., et al.: Study on the meniscus-induced motion of droplets and bubbles by a three-phase Lattice Boltzmann model. Chemical Engineering Science **176**, 35–49 (2018). https://doi.org/10.1016/j.ces.2017.10.025
68. Wu, L., Zhang, Y., Zhang, S., et al.: High order fixed-point sweeping WENO methods for steady state of hyperbolic conservation laws and Its convergence study. Communications in Computational Physics **20**(4), 835–869 (2016). https://doi.org/10.4208/cicp.130715.010216a
69. Xu, J., Louge, M.Y.: Statistical mechanics of unsaturated porous media. Physical Review E **92**(6), 062–405 (2015). https://doi.org/10.1103/physreve.92.062405
70. Yerry, M., Shephard, M.: A modified quadtree approach to finite element mesh generation. IEEE Computer Graphics and Applications **3**(1), 39–46 (1983). https://doi.org/10.1109/mcg.1983.262997
71. Yu, Y., Liu, H., Liang, D., et al.: A versatile lattice Boltzmann model for immiscible ternary fluid flows. Physics of Fluids **31**, 012108 (2018). https://doi.org/10.1063/1.5056765
72. Zhang, Q., Wang, X.P.: Phase field modeling and simulation of three-phase flow on solid surfaces. Journal of Computational Physics **319**, 79–107 (2016). https://doi.org/10.1016/j.jcp.2016.05.016
73. Zolfaghari, A., Piri, M.: Pore-scale network modeling of three-phase flow based on thermodynamically consistent threshold capillary pressures. I. Cusp formation and collapse. Transport in Porous Media **116**(3), 1093–1137 (2017). https://doi.org/10.1007/s11242-016-0814-8

Authors and Affiliations

Deepak Singh[1] · Helmer André Friis[1] · Espen Jettestuen[2] · Johan Olav Helland[3]

Helmer André Friis
helmer.a.friis@uis.no

Espen Jettestuen
esje@norceresearch.no

Johan Olav Helland
jhel@norceresearch.no

[1] Department of Energy and Petroleum Engineering, University of Stavanger, Kristine Bonnevies vei 22, Stavanger 4021, Norway

[2] NORCE Norwegian Research Centre, Tullins gate 2, Oslo 0166, Norway

[3] NORCE Norwegian Research Centre, Prof. Olav Hanssensvei 15, Stavanger 4021, Norway