Petroleum and Coal

Article **Open Access**

Grid Adaptation of Multiphase Fluid Flow Solver in Porous Medium by OpenFOAM

*L. Sugumar, Abhishek Kumar, and Suresh Kumar Govindarajan**

Reservoir Simulation Laboratory, Petroleum Engineering Programme, Department of Ocean Engineering, Indian Institute of Technology – Madras, Chennai, India

Received June 10, 2019; Accepted September 29, 2020

Abstract

The focus of the present work is to develop the two-phase immiscible and incompressible fluid flow solver for porous medium using the object-oriented (C++) Open Source Computational Fluid Dynamics (CFD) toolbox known as OpenFOAM. The reason for choosing the OpenFOAM framework is first elaborated by looking into the history of the programming concepts used in the development of numerical simulators. The scopes and limitations of the existing solvers in the OpenFOAM for the porous medium are looked in detail for identifying the elements of adding any new features to it. The pressure-saturation formulation is the mathematical approach adopted in this present study. In the immiscible two-phase flow, both the wetting phase saturation and the non-wetting phase pressure are solved implicitly using the IMPSAT (IMplicit in Pressure and SATuration) numerical formulation. Additionally, the coupled effect of IMPSAT with Adaptive Mesh Refinement (AMR) technique and the specialized two-dimensional Adaptive Mesh Refinement (2DAMR) technique for 2D problems are investigated. The superiority of using 2DAMR over the existing default hex-mesh cutter algorithm is quantified. The numerical solution is validated with standard Buckley-Leverett semi-analytical solution for one dimension and two-dimension problems. Also, the developed solver is tested for the three-dimensional case.

Keywords: *Two-phase; CFD; Pressure-Saturation; IMPSAT; AMR; Buckley-Leverett.*

1. Introduction

The usage of the proprietary closed source software for porous medium numerical simulations usually attracts a hefty annual licensing cost to the company. At the same time, the end-user is not provided access to the source code of the software for customization or new implementation. Moreover, every new module or additional features are made available by request from the vendor at some additional cost. As a matter of fact, the substantial intellectual contents in the commercial closed source software are primarily inspired or developed from the basic research work conducted by educational and research institutions and made available in the open literature. Scientific Free and Open-Source Software (FOSS) for numerical simulation also do exist along with proprietary commercial software. An inquisitive mind engaged in scientific research always wants to explore and so closed source software is definitely a detriment and FOSS is the only solution. The effects to embrace scientific FOSS for Porous Medium simulation at least among the educational and scientific research community spread across the globe opens the door wider for innovation and reproducible research. With time and collective community-driven quality research, FOSS in the porous medium numerical simulation could be a preferred option to consider over the proprietary commercial software even by the industries. Some of the existing FOSS for porous medium applications are OpenGeoSys (OGS) [1], PFLOTRAN [2], OPM-FLOW, DUMUX [3].

Looking at the history of the development of numerical simulators for the porous medium there is an obvious shift from the functional programming language (FORTRAN, C) to the object-oriented programming (OOP) language (C++) in creating numerical simulators. Tradi-

tionally FORTRAN and C are the preferred programming language, especially among the scientific community. The major drawback of it is first the modeled PDEs are to be numerically discretized and then segregated in terms of components before starting the process of coding. The fate of any functional programming language is to rely heavily on the low-level of programming by performing the four basic mathematical operations (+, -, *, /) on individual floating-point values and arrays. Implementing any new models or validating any new numerical technique was challenging and painstakingly slow even to the well-experienced programmer. The continuous advancement in computer hardware technology bolstered by the research in numerical techniques, linear algebra, data structures, and algorithms equipped us with the necessary infrastructure to solve more and more complex problems. So, the numerical simulators have grown increasingly sophisticated and become even tougher now to handle it in terms of functional programming languages like FORTRAN and C. Initially, there was a strong hesitation among the scientific community to adapt to the object-oriented programming language like C++, fearing the code execution overhead incurred by OOP will impair the performance of a numerical simulator. It was all true that in the mid-1990s, a typical benchmark test proved that the performance of FORTRAN was 1.2 to 10 times faster than C++. But things got changed after 2000, better optimizing C++ compiler emerged and it had improved functionality in the area of vector code generation. The new programming techniques introduced later in C++ like expression template and template meta-program performed optimizations like loop fusion and algorithm specialization. The OOP language like C++ is very expressive and with its powerful functionality of abstraction, it enables the flexible design of algorithms in high-level syntax for the mathematical constructs. At the same time, the benchmark results conducted in 2005 are convincing to believe that C++ runtime performance at the worst can be matched with the functional programming language like FORTRAN while substantially reducing the design and building time of that application [4].

The numerical solution for PDE can be obtained from one of the three numerical methods, (1) finite-difference method (FDM), (2) finite-volume method (FVM), and (3) finite-element method (FEM). The finite-difference is old and simple of all three. It is based on the truncated Taylors series expansion of the derivatives for the differential form of governing PDE. So, it has the restriction of application only to the Cartesian grid. The approach gets complicated when applying to the curvilinear grid and it can't be applied to the unstructured grid which is essential to capture the irregular reservoir topology. While the finite element and finite volume method relies on the integral form of PDE, they can be naturally applied on the unstructured grid. The finite element method is complex and highly mathematical. The governing PDE is not solved directly in the finite element method and rather it undergoes the rigorous mathematical treatment of the weighted residual method in order to get the weak form of the PDE which is then solved. It is not very intuitive to use the finite element method (FEM) to fluid flow problems. The finite element method is convenient and intuitive for finding the stress distribution of the physical system since it deals with local and global stiffness matrices in its method for finding the numerical solution. In the Finite volume method, the divergence term of the volume integrated PDE is converted into a surface integral by applying the Gauss divergence theorem. Then it is computed as the summation of fluxes through the control surface enclosing the finite-sized control volume. Since two adjacent finite-sized control volumes share a common face (control surface) between them, the flux entering one control volume through the common face is the same as the flux leaving the adjacent control volume. So, the finite volume method has an inherent conservativeness property in it. Also, the formulation of finite volume on the computational domain discretized as polyhedral cells is very simple and easy, unlike finite element formulation. It is just a systematic and repeated application of conversation laws on each and every cell of the discretized CFD domain.

Initially, there was a need for many FOSS groups with specific and individual objectives since it is very laborious to build a numerical simulator in a functional programming language like FORTRAN. Even after adapting to object-oriented programming like C++, they failed to appreciate the scope to build a quality software framework to regroup the fragmented FOSS groups and co-exist within it. Most of the existing FOSS for the porous medium numerical

simulator is not build on a quality software framework. The existing DUNE framework is not successful since there is no sense of connecting between the modeled mathematical PDEs and the syntax of the programming language used to code and solve it. Moreover, it is a finite element based which is less charming compared to finite volume for CFD simulations. Another important aspect is it failed to attract more users and developers even after a decade ever since its first release in 2007. Individual groups of FOSS developers not into the quality software framework are more susceptible to lose their focus of research, by often doing the redundant work of reinventing the wheel to meet their short-term goal. Therefore, the need of the hour is the quality software framework for developing FOSS for porous medium numerical simulation preferably using the finite volume formulation.

In OpenFOAM, the current IMPES solver for multiphase flow in the porous medium doesn't have the Adaptive Mesh Refinement (AMR) technique implemented and tested in it. Additionally, there is no specialized mesh cutter algorithm for solving the two-dimensional problem. The default hex mesh cutter algorithm for the three-dimensional problem is a computational overhead when applied for a two-dimensional problem. The present study focuses on using a 2D hex-mesh cutter algorithm with the AMR technique in FVM based OpenFOAM for multiphase fluid flow in porous media. The developed and validated IMPSAT solver is first tested for the static grid. Necessary modification is made to the native IMPSAT solver first to equip it with the default AMR capability of the hex-mesh cutter algorithm (IMPSAT-AMR). Then, the 2D hex-mesh cutter algorithm is developed and tested with an IMPSAT solver for 2D problems. The IMPSAT-AMR solver is tested for a two-phase air-water system.

2. Methodology

2.1. OpenFOAM framework for porous numerical simulator

Unlike, the other open-source simulator like OpenGeoSys, PFLOTRAN, OPM-Flow, DuMux which are solely developed for some specific porous medium application, OpenFOAM [5] is not intended for any specific application and it is designed and developed with an insight of generic toolkit for solving PDE by finite volume method. The conservation laws of continuum mechanics are precisely expressed in the mathematical sense with the help of Partial Differential Equations (PDE). The best part of OpenFOAM is all its effort to retain the same PDE representation while computationally solving the continuum mechanics problems by using the high-level syntax of object-oriented programming. For example, if one intends to solve the transient heat conduction equation which is represented as parabolic PDE $\frac{\partial T}{\partial t} - K\left(\frac{\partial^2 T}{\partial x^2}\right)$. It can be achieved simply by calling the below high-level syntax in OpenFOAM.

*solve(fvm::ddt(T)-kappa*fvm::laplacian(T))*

So obviously there is a strong sense of connection between the mathematical representation of the physical model and the syntax of the programming language used to represent it in the numerical simulator. It additionally checks the dimensional homogeneity of the PDE intended to solve. There is a dissociation between the implementation and interface. Now, the research into mathematical modeling & simulation is completely diversified with advancement into numerical schemes (QUICK scheme, Upwind scheme, GMRES, etc). Thus, it permits the scope for any user to participate at different levels according to their own level of interest and knowledge in physics and programming. The choice of using OpenFOAM as a black-box CFD application or as a platform for research code is left to the individual. It has been successfully used for Large-Eddy Simulation (LES) of external aerodynamic flows [6], realistic wave generation, and interaction on the coastal structure [7], turbo machinery application [8], mass transfer in oxide cells [9].

2.2. Treatment of porous medium in OpenFOAM

As of now, there are no devoted solvers and dedicated boundary conditions for modeling the multiphase porous medium application at the macro scale even in the latest official release of OpenFOAM. But in situations like airflow through the straight or angled duct with porous

plug in its path acting like filters or flow conditioners, the airflow effects in the porous plug are modeled effectively by marking the cell zones of porous plug and adding the additional viscous and inertial resistance by Darcy-Forchheimer relation in the N-S momentum equation [10]. Penalizing the momentum equation only in the region of porous plug and solving the normal momentum equation at the other remaining free space. There exist two types: (1) implicit porous treatment and (2) explicit porous treatment. The implicit porosity treatment is more robust and is opted if (1) the pressure drop is large, (2) porous substance is more anisotropic, and (3) its axis is not aligned with global coordinates. The explicit porosity treatment is preferred for a simple flow condition. The modeling of the multiphase flow in the porous medium essentially involves concepts like phase saturation, relative permeability, and capillarity effect. Horgue *et al.* [10] started a new chapter in OpenFOAM for multiphase fluid flow in a porous medium by developing an open-source toolbox considering all the essential elements mentioned above and published his work. The developed toolbox includes (1) Dedicated IMPES (IMplicit Pressure Explicit Saturation) solver for incompressible and immiscible two-phase porous medium flow, (2) The special boundary condition for treating the phase velocity in the porous medium, (3) The porous medium two-phase flow relative permeability models and capillarity models. In OpenFOAM, the current IMPES solver for multiphase flow in the porous medium doesn't have the Adaptive Mesh Refinement (AMR) technique implemented and tested in it. Additionally, there is no specialized mesh cutter algorithm for solving the two-dimensional problem. The default hex mesh cutter algorithm for the three-dimensional problem is a computational overhead when applied for a two-dimensional problem. Therefore, an AMR technique with a 2D hex-mesh cutter algorithm has been used herein OpenFOAM for multiphase fluid flow in porous media. This part should contain sufficient detail so that all procedures can be repeated. It can be divided into subsections if several methods are described.

2.3. Pressure-Saturation formulation

Choosing the two primary unknowns as wetting phase saturation S_b and non-wetting phase pressure p_a results in the system of improved characteristics with reduced coupling and non-linearity behavior. The following is the system of governing equation in S_b and p_a

$$\frac{\partial}{\partial t}\left((1-S_b)\phi\right)+\nabla.\left[-\frac{Kk_{ra}}{\mu_a}(\nabla p_a-\rho_a g)\right]=Q_a \quad (1)$$

$$\frac{\partial}{\partial t}(S_b\phi)+\nabla.\left[-\frac{Kk_{rb}}{\mu_b}(\nabla p_a-\nabla p_c(S_b)-\rho_b g)\right]=Q_b \quad (2)$$

where K is the absolute permeability of the porous medium; p_c is the capillary pressure which depends on saturation S_b and ϕ is the porosity of the medium.

Adding the equation (1) and (2) results as

$$\frac{\partial}{\partial t}(S_b\phi+\phi-S_b\phi)+\nabla.\left[-\frac{Kk_{ra}}{\mu_a}(\nabla p_a-\rho_a g)-\frac{Kk_{rb}}{\mu_b}(\nabla p_a-\nabla p_c(S_b)-\rho_b g)\right]=Q_a+Q_b \quad (3)$$

$$\nabla.[(M_a+M_b)\nabla p_a]=-\nabla.\left(M_a\rho_a g+M_b\rho_b g-M_b\frac{\partial p_c}{\partial S_b}\nabla S_b\right)+Q_a+Q_b \quad (4)$$

Rewriting the equation (2) by substituting the $\nabla p_c=\frac{\partial p_c}{\partial S_b}$

$$\phi\frac{\partial}{\partial t}(S_b)+\nabla.\left[-\frac{Kk_b}{\mu_b}\left(\nabla p_a-\frac{\partial p_c}{\partial S_b}\nabla S_b-\rho_b g\right)\right]=Q_b \quad (5)$$

$$\phi\frac{\partial}{\partial t}(S_b)+\nabla.\left[-M_b\nabla p_a+M_b\frac{\partial p_c}{\partial S_b}\nabla S_b+M_b\rho_b g\right]=Q_b \quad (6)$$

The equations (4) & (6) form the set of governing equations for Pressure-Saturation formulation [11-12].

2.4. IMPSAT (IMplicit in Pressure and SATuration) method

This method is also called as Sequential Fully Implicit (SFI) method. Unlike the IMPES method where the divergence term $\left(\nabla.(-M_b(S_b^n)\nabla p_a^n)\right)$ is at the n-th time step, in this method, it is solved by the inner newton's iterative correction for each time step. Though it results in extra computational cost but gives more numerical stability as compared to the IMPES method. The more details of the IMPSAT method can be found in [13].

$$T\left(S_b^{v+1}, S_b^v, u_b^{v+1}\right) := \phi \frac{S_b^{v+1} - S_b^v}{\Delta t} + \nabla . \left(u_b^{v+1}\right) = f^v(Q_b, M_b, p_c, \rho_b) \quad (7)$$

where $u_b^{v+1} = -M_b(S_b^{v+1}) \nabla p_a^{v+1}$

$$P(S_b^v) := -\nabla . \left(M_a(S_b^v) + M_b(S_b^v)\right) \nabla p_a^{v+1} = f_p^v(Q_b, M_b, p_c, \rho_b) \quad (8)$$

3. Problem definition

3.1. One-Dimensional case for validation

In order to first test the IMPSAT solver and then it's AMR capability, the standard one-dimensional case of Buckley and Leverett [14] for the air-water system is considered for the 1D reservoir domain. The respective equation for the radial Buckley-Leverett flow is as follows [15]

$$\frac{\partial S_b}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}\left(r f_b(S_b, r)\right) = 0 \quad (9)$$

Here '*r*' is defined as the radial distance from the point of injection in the circular domain and S_b is the wetting phase saturation. Since the flow is incompressible $f_b(S_b,r)=f_b(S_b)/r$. So, the above equation (9) is written as

$$\frac{\partial S_b}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}\left(f_b(S_b)\right) = 0 \quad (10)$$

$$\frac{\partial S_b}{\partial t} + \frac{1}{r}\frac{\partial f_b(S_b)}{\partial S_b}\frac{\partial S_b}{\partial r} = 0 \quad (11)$$

The information about reservoir properties, fluid properties, and Relative permeability model used is listed in Table 1. The obtained numerical result is validated against the standard Buckley-Leverett semi-analytical solution as in Fig. 1.

Table 1. Information of (a) Reservoir properties, (b) Fluid properties, and (c) Relative permeability model [10]

(a) Reservoir properties		
Length of reservoir	1 m	
Porosity	0.5	
Permeability	1×10^{-11} m^2	
Production rate	1×10^{-7} m^3/sec	
(b) Fluid properties		
Fluid	ρ (kg/m^3)	μ (Pa.s)
Water	1000	1×10^{-3}
Air	1	1.76×10^{-5}
(c) Relative permeability mode		
	power coefficient *m*	
Brooks and Corey [16]	3	
van Genuchten [17]	0.5	

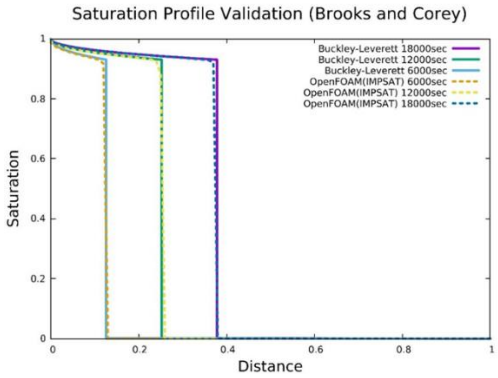


Fig. 1 Validation of Water Saturation with Buckley-Leverett (Air-Water) [14]

3.2. Two-dimensional case for validation

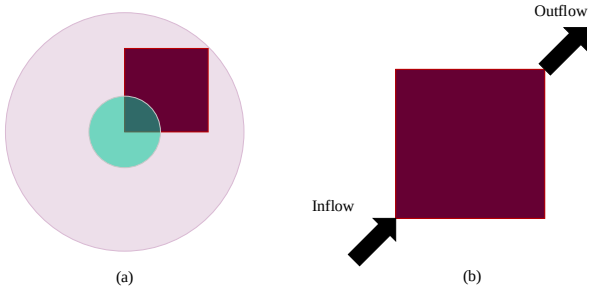


Fig. 2 Water Flooding for diagonal Injection and Extraction Well

Consider the case as shown in Fig. 2 where the water phase is pumped in from the left lower corner into the domain saturated with oil. Doing so will displace the oil to the right upper corner production well because the sides of the domain are impermeable. So, the principal flow direction will be the diagonal of the domain.

The water saturation line profile is examined along the diagonal line *(x, y), x = y*, for the considered case computed in OpenFOAM with the fine-scale semi-analytical numerical result of the radial Buckley-Leverett flow [18]. It is described as a scenario where the water phase is being pumped into the center of a circular domain.

Huber and Helmig [15] adopted the above approach of validating the special case of diagonal well two-dimensional problem by radial Buckley-Leverett flow semi-analytical solution as shown in Fig. 2(a) and 2(b). The initial condition used by Huber and Helmig is uniform saturation of $S_b=0$ inside the domain and the results are compared at 0.075 PVI (pore volumes injected). The pore volume injected is given by $PVI = \left(\frac{(injectionrate \times time)}{(volume \times porosity)}\right)$.

In OpenFOAM, the computational domain of dimension 1m×1m×2.83cm is considered and the porosity of it is 0.5. The injection well is located at the lower-left corner. All the walls of the domain are impermeable except a patch on the top right corner of 10cm. The injection well strength is 1×10^{-6} m^3/sec.

To validate with Huber and Helmig, the above-considered case in OpenFOAM has to be simulated until 1061 seconds to compare the saturation profile at 0.075 PVI. Fig. 3(a) shows the saturation profile at 0.075 PVI for a two-dimensional case. The quantitative validation on the diagonal line from the injection well to the producer well is shown in Fig. 3(b). The numerically simulated results of the IMPSAT solver developed in OpenFOAM are in good comparison with Huber and Helmig.

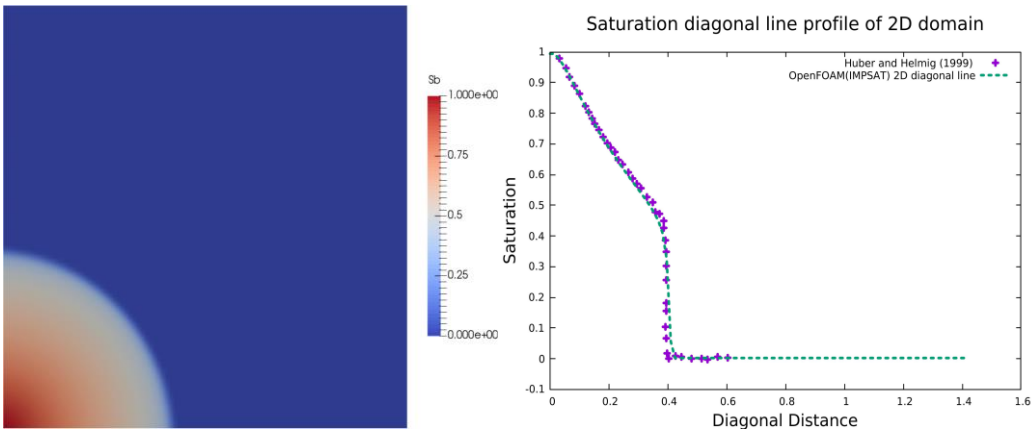


Fig. 3. (a) Water Flooding at the instance of 0.075 PVI, (b) 2D Diagonal line validation at the instance of 0.075 PVI

4. Results and discussion

4.1. Grid independent study

The domain of the water injection configuration along with its boundary conditions (BCs) used in this present work is shown in Fig. 4(a). A non-uniform grid has been generated with fine cells towards the injection well location in X-direction. It results in slightly clustered grids in the X direction whereas equal and uniform grids are generated in the Y direction as shown in Fig. 4(b). The computational grid is generated by using *blockMesh* utility.

Grid independence study is to be conducted by taking into consideration four different mesh configurations as listed in Table 2. The water is injected at the rate of 1 liter/sec from the injection well. The simulated numerical results are post-processed in *Paraview*.

Table 2. Three mesh configuration for grid independence study

Grid	X-Direction	Y-Direction	Total Cells
Very coarse	40	20	800
Coarse	80	40	3200
Medium	160	80	12800
Fine	240	120	28800

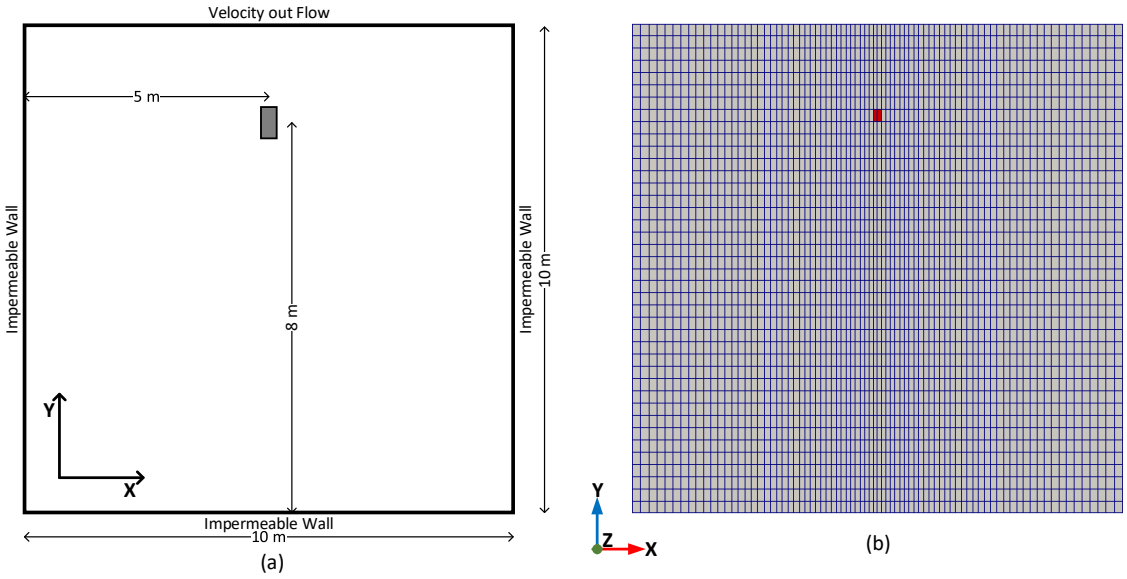


Fig. 4. (a) Computational domain and (b) Computational grid with injection well location

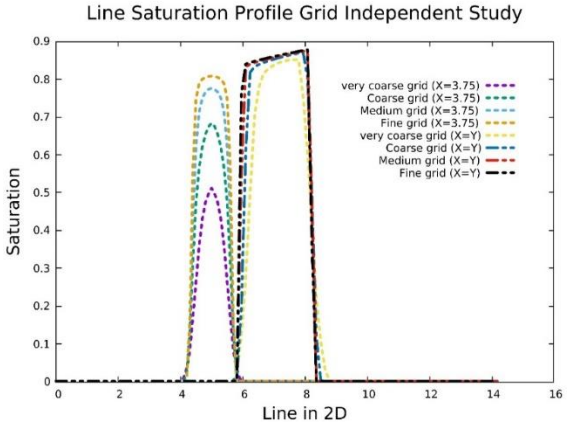


Fig. 5 Grid Independent Study of Saturation on line profiles of 2d domain

It can be inferred from Fig. 5 by the saturation line profile compared for all four-grid configuration (very coarse grid, coarse, medium, fine) over X=Y and X=3.735 of the domain shown in Fig. 4(a). The variation in the saturation is very minimum for the medium and fine grid as can be seen in Fig. 6. The medium grid configuration arrives at the converged saturation profile with lesser computing time as compared to the fine grid. In CFD for the multiphase fluid, the mass balance of the system can be easily verified from the saturation contour profile, and at all times its value is $0 < S_b < 1$. The same is inferred in the present case.

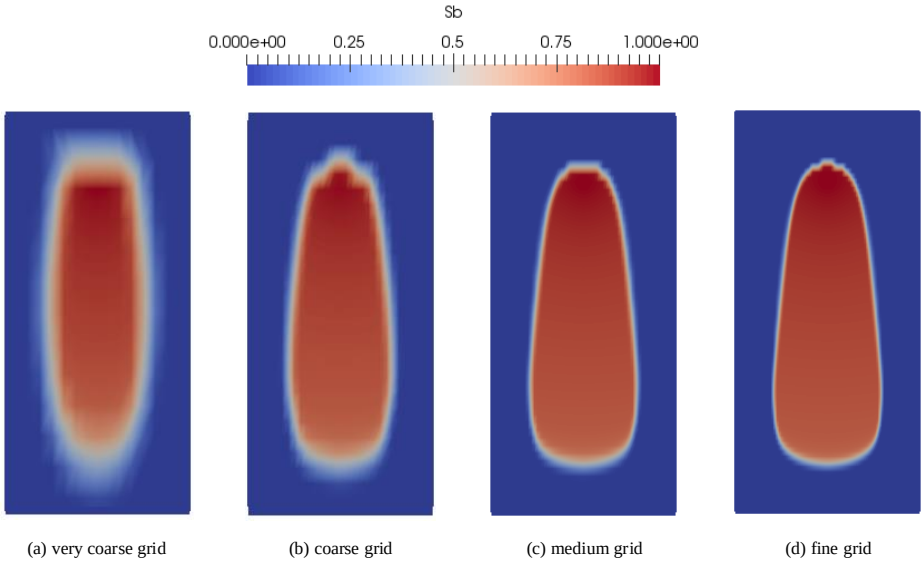


Fig. 6. S_b contour for a very coarse, coarse, medium and fine grid

4.2. Need for 2D hex-mesh cutter

The default hex-mesh cutter engine shipped with OpenFOAM is efficient in handling only three-dimensional mesh. The reason for it could be that OpenFOAM is geared to solve all the problems in 3D by default. In order to solve 2D problems, first, the 3D mesh is created with only one cell in the third dimension and then it is instructed that the boundary patches normal to the third dimension to be specified as 'empty' in the boundary file of the simulation control settings. So, those faces in that boundary patches marked as empty won't participate in the total control volume flux calculation process with its associated cell and therefore it requires no solution on those 'empty' boundary patches. When the default hex-mesh cutter engine is applied to 2D mesh it starts to create the cells unnecessarily in the third dimension also and because of it the solution is in no way going to improve at all. It is only a mere computational overhead to refine the cells in the third dimension for the 2D mesh. It is wise to make the necessary modification in the existing hex-mesh cutter engine and create a specialized 2D hex-mesh cutter engine to enable the handling of 2D mesh more efficiently.

4.3. D Hex-Mesh cutter refinement algorithms

The 2D hex-mesh cutter engine is very similar to that of the default 3D hex-mesh cutter engine with a slight difference. One extra information about the normal axis to the empty boundary patches is to be given for this new engine. The process of 2D hex-mesh cutter refinement is very descriptive in Fig. 7. The new cell center point is not created in the current 2D hex-mesh cutter engine.

(1) As the first step here, the new points are created on the face center of the empty boundary patches of the candidate cell for refinement as in Fig. 7(b).
(2) Then, the next step is to loop for the edges on the faces of the empty boundary patch and create a new point at its mid-point. This is shown in Fig. 7(c).
(3) Four new faces are created on the empty boundary patches of the chosen cell as in Fig. 7(d).
(4) The remaining four faces which don't have the face center are divided into two faces as in Fig. 7(e).
(5) The last step is to create the four new internal faces as in Fig. 7(f).

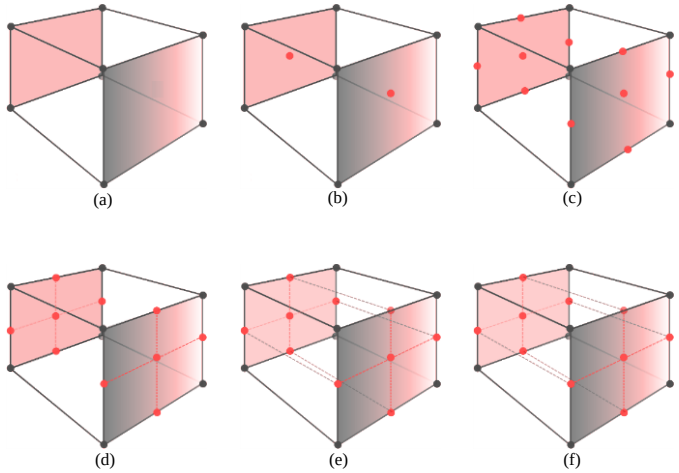


Fig. 7 The Process of Hex-mesh cutter Refinement algorithm

The coarsening process and the solution field mapping are done in the same way as discussed for the default 3D hex-mesh cutter engine.

4.4. Testing the AMR-IMPSAT Solver

The developed and validated IMPSAT solver is first tested for the static grid. Now here, the necessary modification is made to the native IMPSAT solver first to equip it with the default

AMR capability of the hex-mesh cutter algorithm (IMPSAT-AMR). Then, the 2D hex-mesh cutter algorithm inspired by the work of Ahmad Baniabedalruhman [19] is developed and tested with an IMPSAT solver for 2D problems. The IMPSAT-AMR solver is tested for both multiphase air-water and water-oil system in the following sections.

4.4.1. AMR in multiphase air-water porous system

The details of the case like domain size, boundary conditions, and injection well strength areas are given in Fig. 4(a). The numerical simulation is performed on the dynamic grid. The value of the S_b field is used as the criterion to check whether a cell requires refinement or coarsening. Obviously, it could only be a natural choice for the following reasons.

(1) Hex-mesh cutter engine in OpenFOAM necessitates a scalar field as adaptation criterion and S_b comes under the scalar field category.
(2) The range of solution of the water saturation field S_b is well known and it always honors the relation $S_{b,\,min} < S_b < S_{b,\,max}$.
(3) The key functions in the multiphase flow of the porous system computation like relative permeability and capillary functions are related to water saturation field S_b.

The range of the S_b is [0.2; 0.8]. If the S_b value is within this specified range in any cell then it is entitled to the refinement process otherwise it is selected for the coarsening process. The maximum refinement level is chosen to be 2. The refinement interval is specified as 1 which implies that at every time step the adaptation criterion is checked to carry out the mesh modification process. The maximum cell limitation for the refinement process is kept as 400000 cells which may not be achieved during the entire simulation time and so the adaptation process is always on. First, a very coarse mesh of 800 cells is given as initial base mesh to start with for:

(1) The default IMPSAT-AMR of OpenFOAM which is referred to in this work as AMR2D3D
(2) The specialized IMPSAT-2DAMR which is referred to in this work as AMR2D2D.

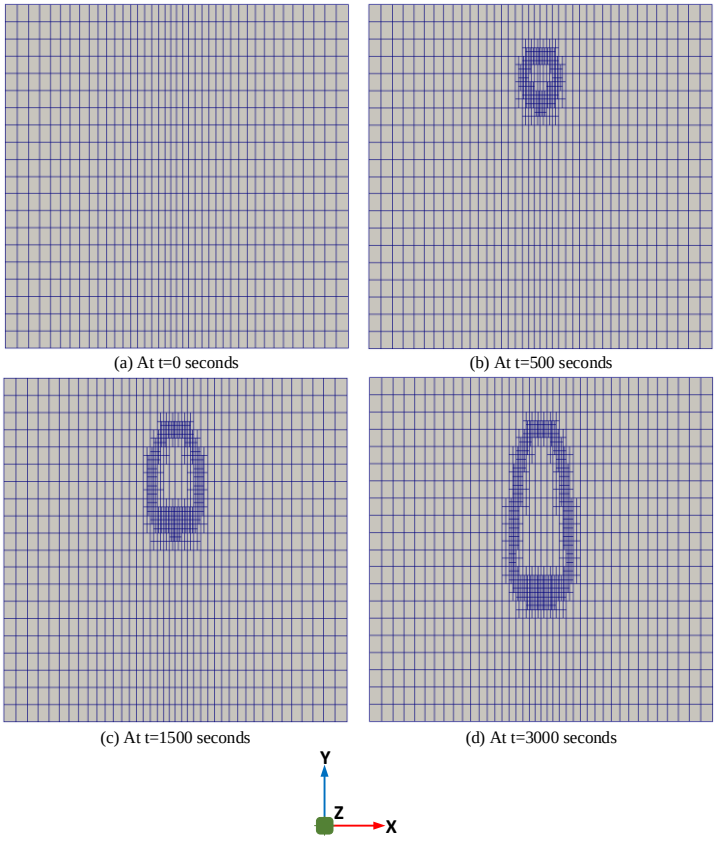


Fig. 8. The grid adaptation process at different time instances for both AMR2D3D and AMR2D2D

The grid adaptation process for both AMR2D3D and AMR2D2D is shown in Fig. 8 for different time instances such as t=0 sec, t=500 sec, t=1500 sec, and t=3000 seconds. There is no difference in both of the processes in the view as shown in Fig. 8. The distinct difference between the AMR2D3D and AMR2D2D solver's result is noticed in the adapted grid cut-view in Fig. 9.

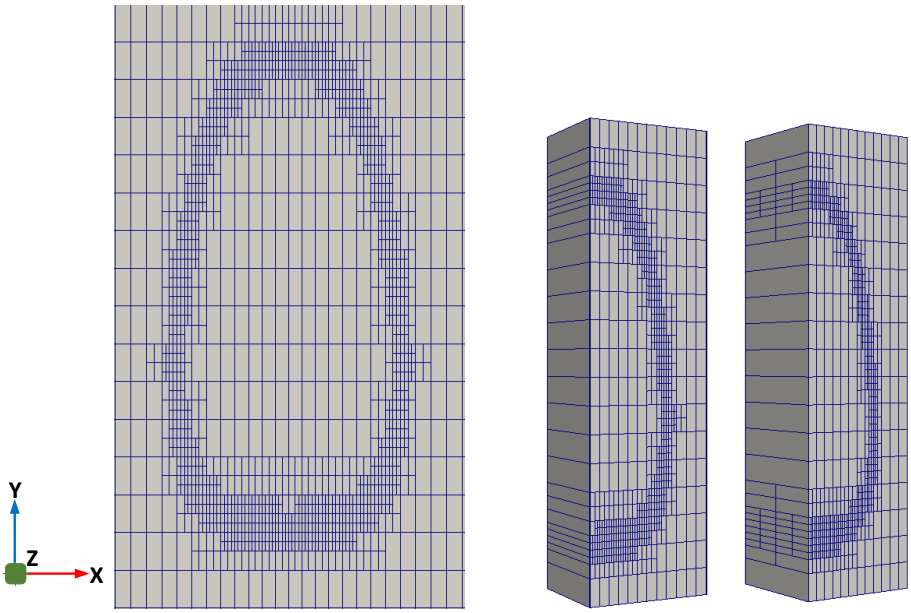
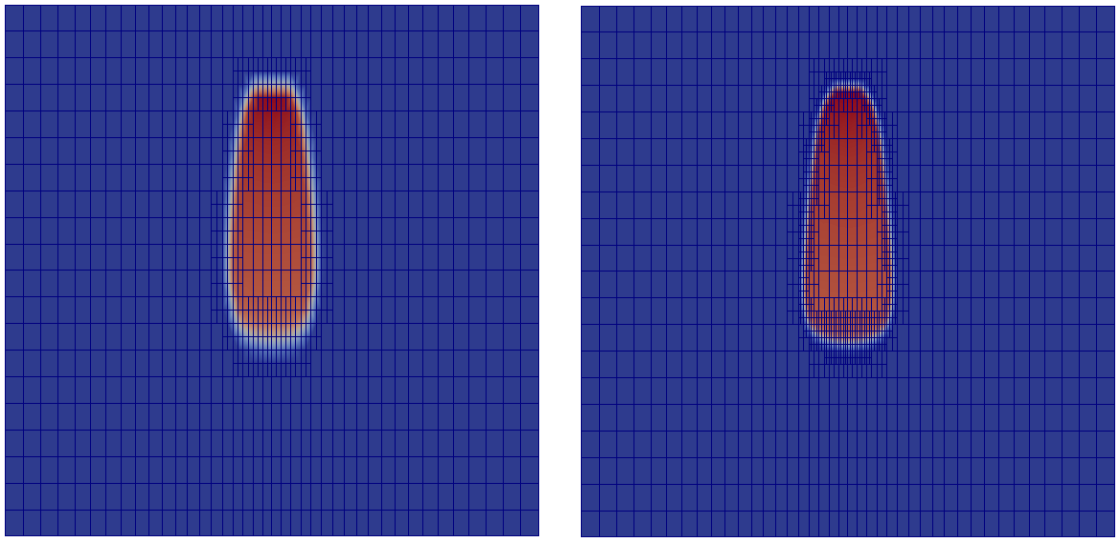


Fig. 9. The zoom-in view of the adaptive grid for AMR2D2D and AMR2D3D



(a) Refinement level 1 (b) Refinement level 2

Fig. 10. Refinement level comparison S_b contour with a grid superimposed

The results of both AMR2D3D & AMR2D2D is compared both quantitatively and qualitatively with fine static grid (28,800 cells) solution. The qualitative comparison is as shown in Fig. 10. It can clearly be seen from Fig. 10 that the interface is captured more precisely for computation set with refinement level 2 over level 1. The quantitative comparison is done at three sampling line profiles such as x=y, y-axis, and x=3.75 m as shown in Fig. 11(a). The slight variation is observed in the quantitative plot at the sampling line profile of x=3.75 m.

The comparison of the time history of the adapted cells for both AMR2D3D and AMR2D2D is plotted in Fig. 11(c) and its inferences are listed below.

(1) Both the processes start with 800 cells as the initial grids at time t=0 seconds.

(2) The general increasing trend is observed in both the processes until the end of simulation of 3000 seconds.
(3) At any given instance, after t=0 seconds, the number of adapted grids for the AMR2D3D process is more than the AMR2D2D process.
(4) At the end of 3000 seconds simulation time, the number of grid cells in AMR2D2D is just 1500 cells while the number of grid cells in AMR2D3D is 3400. The AMR2D2D process takes slightly less than 50% number of grid cells as compared to AMR2D3D.
(5) The initial high slope of the curves for both the processes from t=0 seconds to t=200 seconds is indicative of the fact that the particular phase is only with the refinement process. After that, both the coarsening and refinement processes are taking place.

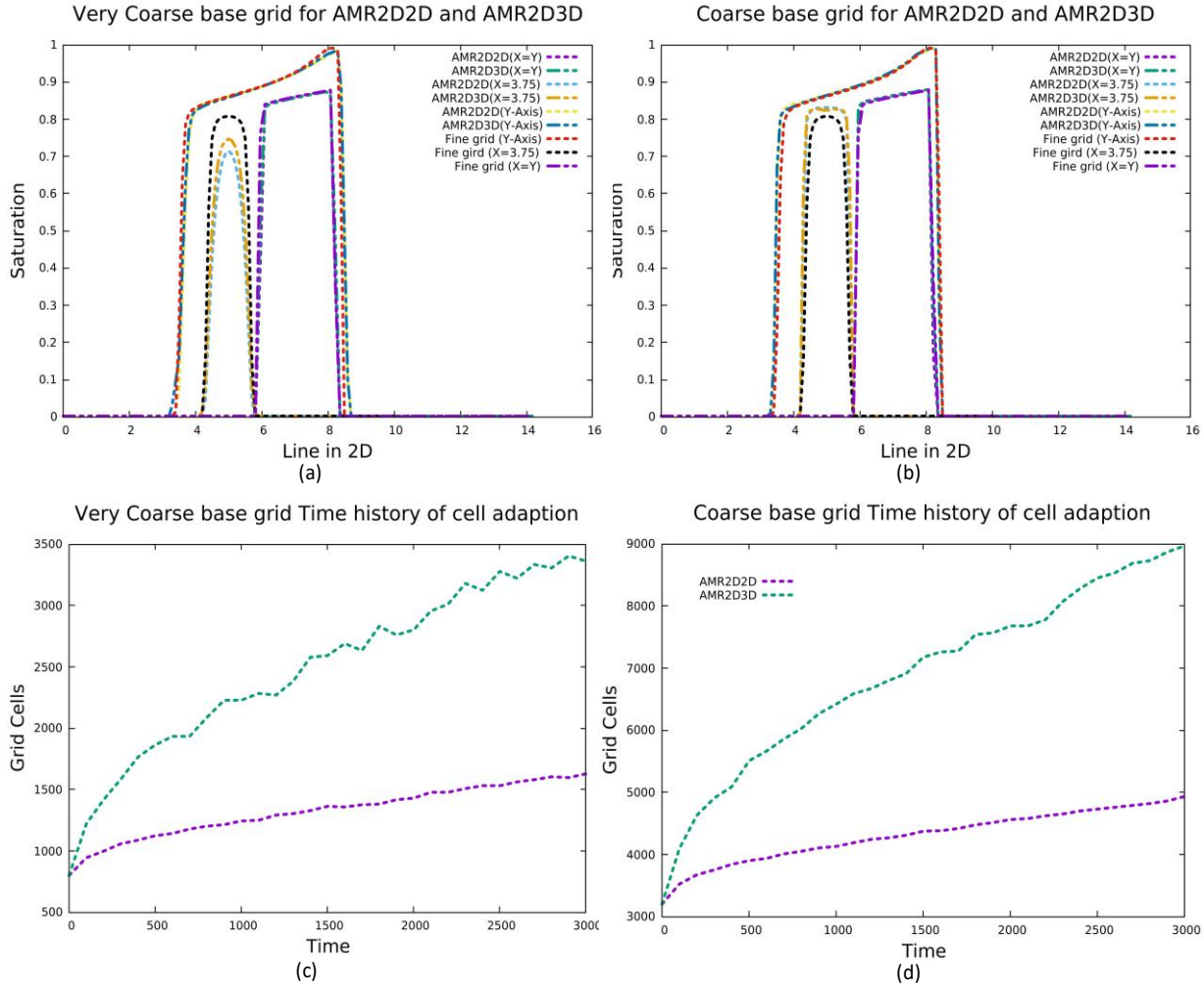
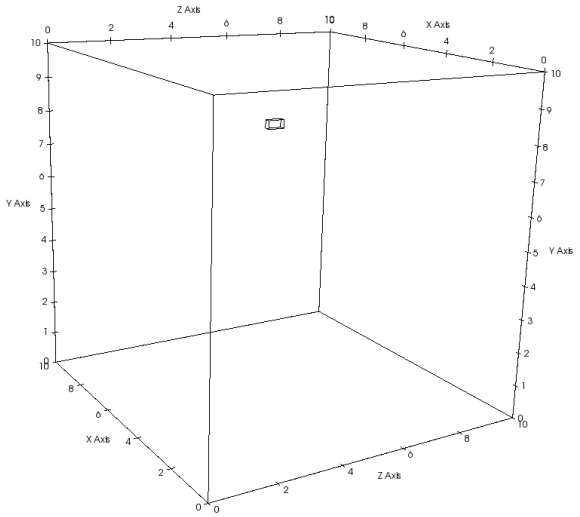


Fig. 11. Comparing S_b on line profiles for AMR2D2D, AMR2D3D with (a) very coarse base grid of 800 cells, (b) coarse base grid of 3200 cells, and; Time history of Adapted cells in AMR2D2D and AMR2D3D with (c) very coarse base grid, and (d) coarse base grid

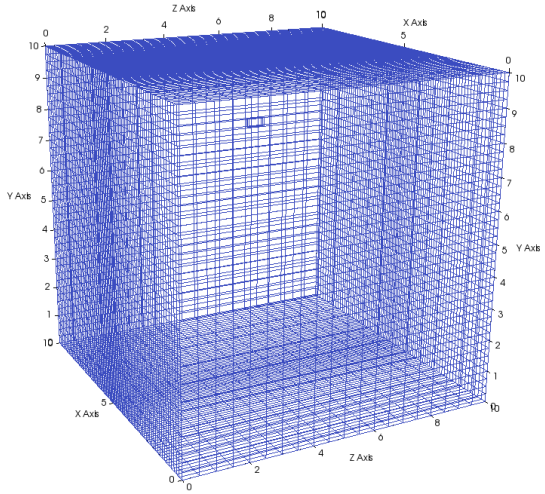
As compared to the static grid process, AMR2D3D is far better since for static grid computation 28,800 cells are used whereas for the AMR2D3D process it took only 3400 cells. The static grid computation took 8.5 times more grid cells than AMR2D3D and 19.2 times more than AMR2D2D. The marked superiority of AMR2D2D is distinctively clear with the above information. As there is a practice of grid-independent study for arriving at the quality computational mesh assertively for the static grid, in a similar way, the whole above set of procedures is repeated for both AMR2D3D and AMR2D2D solver for another coarse base grid of 3200 cells as initial cells. The quantitative comparison is shown in Fig. 11(b) respectively. The time history of refined cell details for both AMR2D3D and AMR2D2D is plotted in Fig. 11(d). The slight

variation observed in the quantitative plot at the sampling line profile of x=3.75 m in the former case is also not recognizable here. The aspect regarding the total computational time for all the above cases is studied separately and elaborately in the later section under performance study.

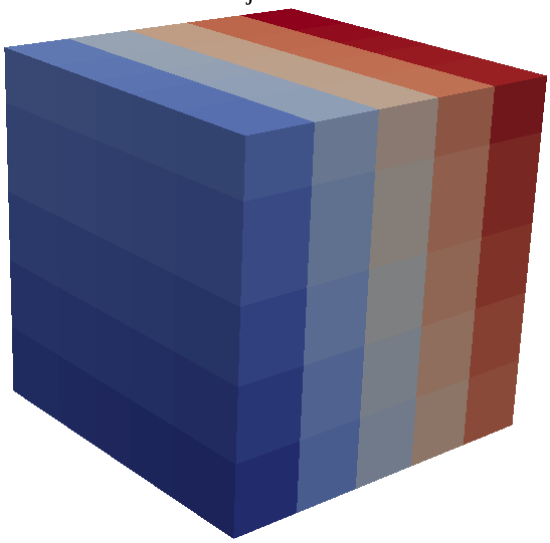
4.4.2. Comparison of static grid versus AMR for 3D case



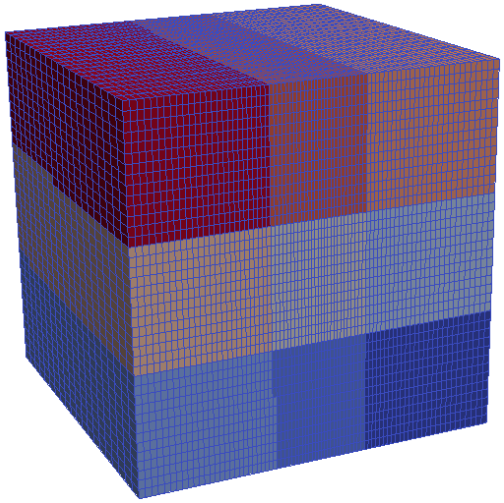
(a) The sketch of physical domain 10m×10m×10m and injection well



(b) The view of the CFD wireframe mesh



(c) Parallel computing 125 CPU domains for water flooding with grid adaptation

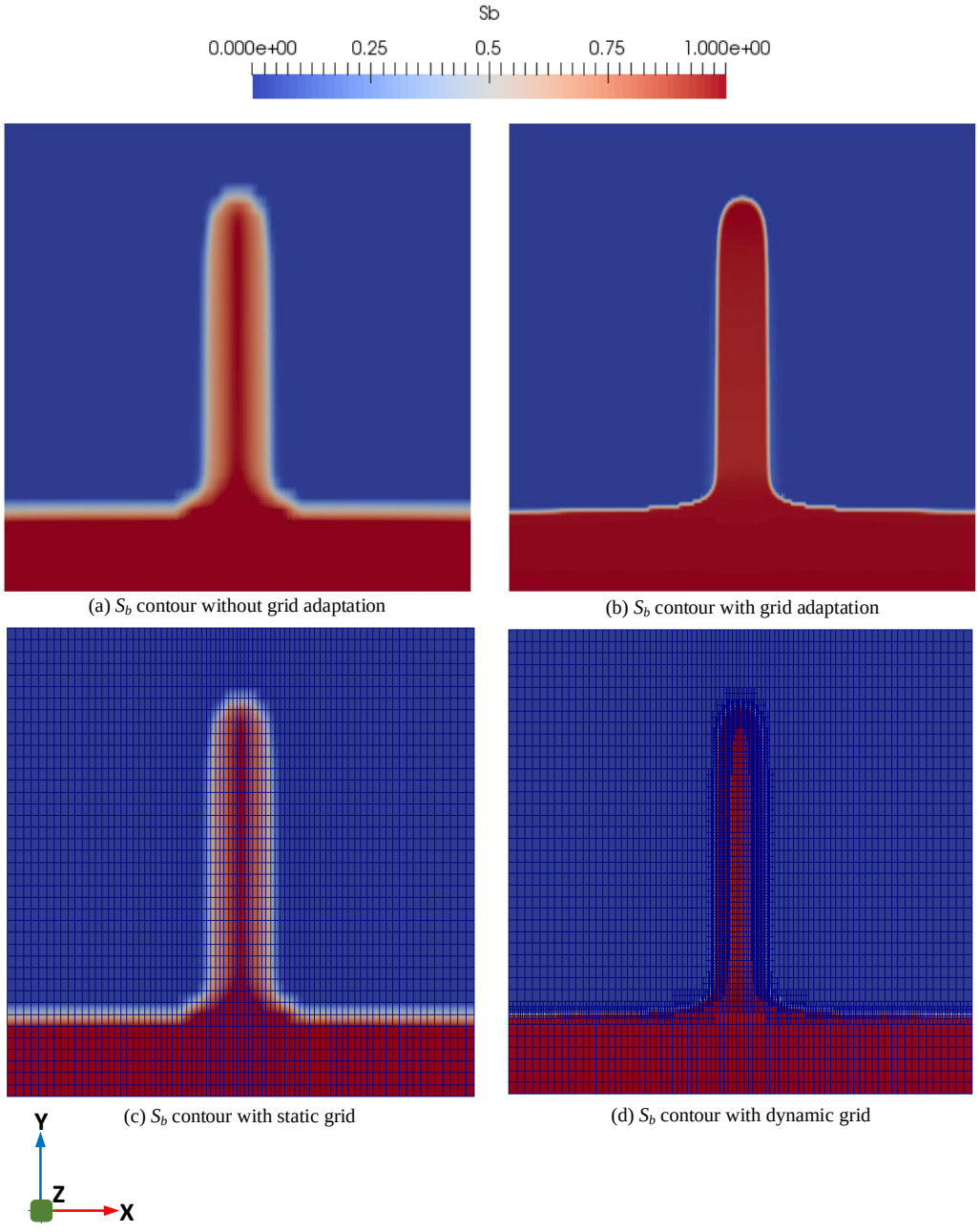


(d) Parallel computing 9 CPU domains for water flooding without grid adaptation

Fig. 12. The physical domain for the 3D case with wireframe mesh and domain decomposition of parallel computing CPUs for AMR and static grid computations

The advantages of the AMR technique and its effects are realized much better when a three-dimensional case is solved. In this pursuit, the previous 2D case of 10m×10m domain is converted into the three-dimensional case by taking the third dimension also to be 10 m. So, the dimension of the 3D domain is 10m×10m×10m as shown in Fig. 12 (a). The injection well is placed at (5m, 8m, 4.75m) and its injection rate is 1 liter/second. The front, back, right, left and bottom wall is chosen to be impermeable wall boundary condition. The top boundary is given as a velocity outflow boundary condition. The number of grids in x, y, and z directions

are 80 cells, 40 cells, and 20 cells respectively. The view of CFD wireframe mesh is shown in Fig. 12 (b). The same CFD mesh is given for both IMPSAT and IMPSAT-AMR solvers. In order to solve the grid adaptation process of IMPSAT-AMR solver by parallel computing, 125 processors are engaged. The domain decomposition for 125 processors is shown in Fig. 12 (c). Since the static grid computation by IMPSAT solver is less computationally intensive only 9 processors are engaged. In Fig. 12 (d) shows the domain decomposition details of the CFD mesh assigned to the nine processors.



(a) S_b contour without grid adaptation

(b) S_b contour with grid adaptation

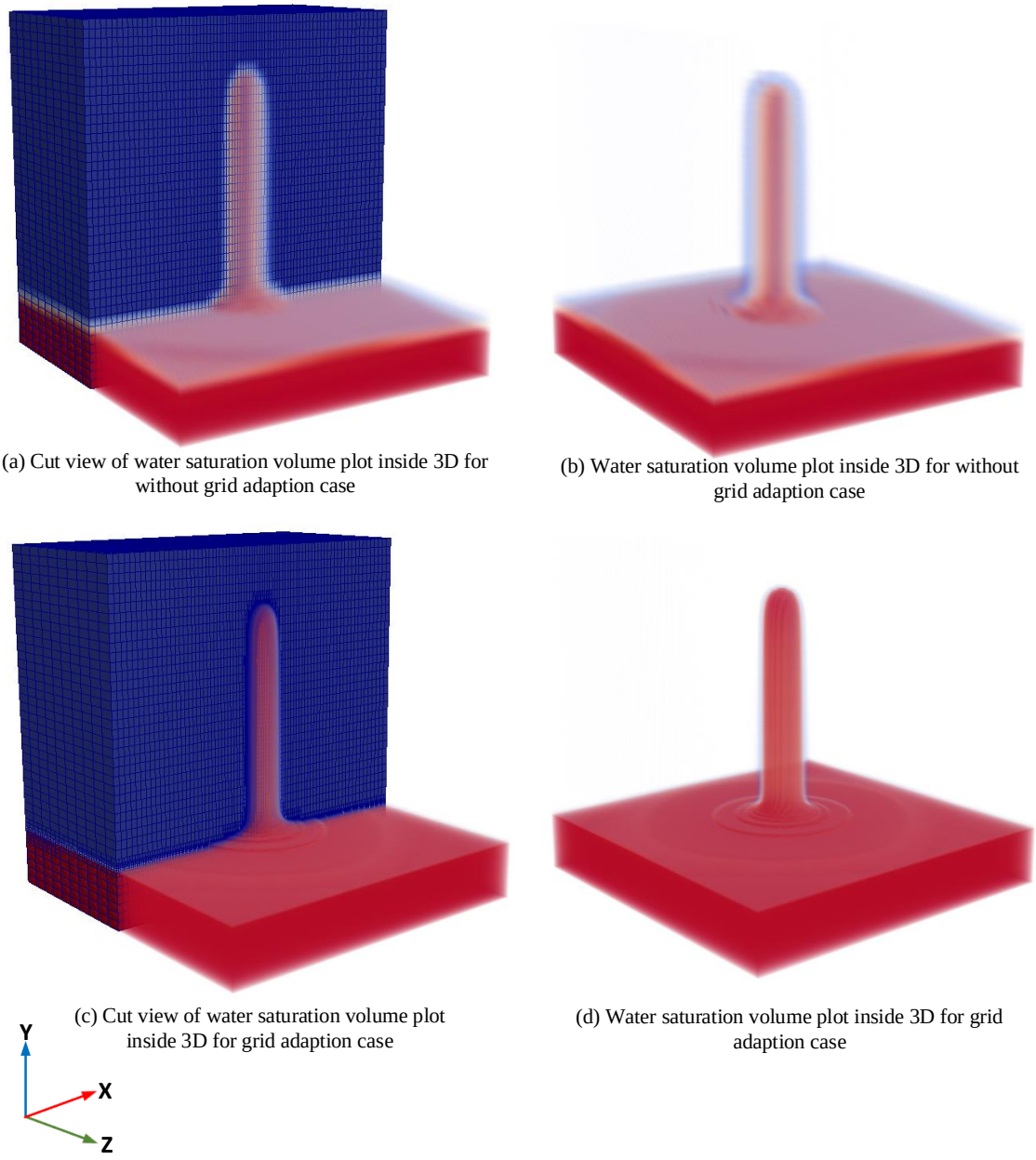
(c) S_b contour with static grid

(d) S_b contour with dynamic grid

Fig. 13 The saturation contour S_b comparison at t=3000 seconds at Z=0 plane

The total simulation time is set as 1,06,000 seconds. The saturation contour is compared for with and without grid adaptation processes at two-time instances such as t=3000 seconds and t=1,06,000 seconds as shown in Fig. 13 and 14 respectively. From the result in Fig. 13 and 14, it can be inferred that the adaptive mesh refinement process captures the air and water interface more precisely. The errors incurred by artificial diffusion due to insufficient

mesh points are addressed by AMR. At any instance in time, the AMR process uses the optimal number of grid points just with the simple concept of adapting and coarsening the mesh at where and when it is required. The volume contour of S_b at 1,06,000 seconds is shown in Fig. 14 and it gives a clear 3D picture of artificial diffusion in the static grid process. The initial mesh size of 64,000 cells at 0 seconds in the AMR process has increased to 1,19,622 cells and 1,76,966 cells at 3000 seconds and 1,06,000 seconds respectively.



(a) Cut view of water saturation volume plot inside 3D for without grid adaption case

(b) Water saturation volume plot inside 3D for without grid adaption case

(c) Cut view of water saturation volume plot inside 3D for grid adaption case

(d) Water saturation volume plot inside 3D for grid adaption case

Fig. 14 S_b volume plot at t=106000 seconds

5. Conclusion

The developed IMPSAT solver in this paper is validated with literature and its fidelity is tested with a variety of different test cases which includes two dimensional and three-dimensional cases. The IMPSAT solver is implemented with default AMR capability and also with a specialized 2D AMR engine. The developed solver is compared with the static grid and the superiority of AMR2D2D is clearly been shown as compared to the AMR2D3D or the static grid process in terms of the number of cells used. It appears that the adaptive mesh refinement

process captures the air and water interface more precisely. The errors incurred by artificial diffusion due to insufficient mesh points are addressed by AMR. And thus, the AMR process uses the optimal number of grid points just with the simple concept of adapting and coarsening the mesh at where and when it is required. It is concluded that the specialized 2D AMR engine outperforms the existing default AMR for 2D cases in terms of both computational time and runtime memory. The MPI (Message Passing Interface) capability of the developed solver is also tested for the 3D case. The future work can focus on using the present solver for improving the assessment for viscous fingering during the enhanced oil recovery.

Symbols

a	*non-wetting phase*
b	*wetting phase*
i	*phase*
K	*absolute permeability of the porous medium*
k_{ri}	*relative permeability of phase i*
M_i	*mobility of phase i*
p_i	*pressure of phase i*
p_c	*capillary pressure*
ρ_i	*density of phase i*
ϕ	*porosity of the medium*
μ_i	*viscosity of phase i*
Q_i	*source term*
r	*radial distance from the point of injection*
S_i	*saturation of phase i*

References

[1] Kolditz O, Bauer S, Bilke L, Böttcher N, Delfs JO, Fischer T, Görke UJ, Kalbacher T, Kosakowski G, McDermott CI, Park CH. OpenGeoSys: an open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical (THM/C) processes in porous media. Environmental Earth Sciences, 2012; 67(2): 589-599.

[2] Lichtner PC, Hammond GE, Lu C, Karra S, Bisht G, Andre B, Mills RT, Kumar J, Frederick JM. PFLOTRAN Web page. Los Alamos National Laboratory, location: Los Alamos, NM [Available at http://www.pflotran.org]. 2019.

[3] Flemisch B, Darcis M, Erbertseder K, Faigle B, Lauser A, Mosthaf K, Müthing S, Nuske P, Tatomir A, Wolff M, Helmig R. DuMux: DUNE for multi-{phase, component, scale, physics,…} flow and transport in porous media. Advances in Water Resources, 2011; 34(9): 1102-1112.

[4] Veldhuizen TL, Jernigan ME. Will C++ be faster than Fortran?. In International Conference on Computing in Object-Oriented Parallel Environments 1997 Dec 8 (pp. 49-56). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-63827-X_43

[5] Weller HG, Tabor G, Jasak H, Fureby C. A tensorial approach to computational continuum mechanics using object-oriented techniques. Computers in physics, 1998; 12(6): 620-631.

[6] Lysenko DA, Ertesvåg IS, Rian KE. Large-eddy simulation of the flow over a circular cylinder at Reynolds number 3900 using the OpenFOAM toolbox. Flow, turbulence and combustion, 2012; 89(4): 491-518.

[7] Higuera P, Lara JL, Losada IJ. Simulating coastal engineering processes with OpenFOAM®. Coastal Engineering, 2013; 71: 119-34.

[8] Beaudoin M, Jasak H. Development of a generalized grid interface for turbomachinery simulations with OpenFOAM. InOpen source CFD International conference 2008 Dec 4 (Vol. 2). Berlin.

[9] Novaresio V, García-Camprubí M, Izquierdo S, Asinari P, Fueyo N. An open-source library for the numerical modeling of mass-transfer in solid oxide fuel cells. Computer Physics Communications, 2012; 183(1): 125-146.

[10] Horgue P, Soulaine C, Franc J, Guibert R, Debenest G. An open-source toolbox for multiphase flow in porous media. Computer Physics Communications, 2015; 187: 217-226.

[11] Adler PM, editor. Multiphase flow in porous media. The Netherlands: Kluwer Academic Publishers; 1995 Nov 30.

[12] Baer J. Dynamics of fluids in porous media. America Elsevier Pubilishing Company. 1972.

[13] Jenny P, Lee SH, Tchelepi HA. Adaptive fully implicit multi-scale finite-volume method for multi-phase flow and transport in heterogeneous porous media. Journal of Computational Physics, 2006; 217(2): 627-641.

[14] Buckley SE, Leverett M. Mechanism of fluid displacement in sands. Transactions of the AIME, 1942; 146(01): 107-116.
[15] Huber R, Helmig R. Multiphase flow in heterogeneous porous media: A classical finite element method versus an implicit pressure–explicit saturation-based mixed finite element–finite volume approach. International Journal for Numerical Methods in Fluids, 1999; 29(8) :899-920.
[16] Brooks RH, Corey AT. Properties of porous media affecting fluid flow. Journal of the irrigation and drainage division, 1966; 92(2): 61-90.
[17] Van Genuchten MT. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils 1. Soil science society of America journal, 1980; 44(5): 892-898.
[18] Blunt M, Rubin B. Implicit flux limiting schemes for petroleum reservoir simulation. Journal of Computational Physics, 1992; 102(1): 194-210.
[19] Baniabedalruhman, A. Dynamic meshing around fluid-fluid interfaces with applications to droplet tracking in contraction geometries. PhD Thesis. Michigan Technological University, 2015.

To whom correspondence should be addressed: professor Suresh Kumar Govindarajan, Reservoir Simulation Laboratory, Petroleum Engineering Programme, Department of Ocean Engineering, Indian Institute of Technology – Madras, Chennai, India, E-mail: *gskumar@iitm.ac.in*