POLITECNICO DI MILANO
DEPARTMENT OF ENERGY ENGINEERING
MASTER PROGRAMME IN
ENERGY ENGINEERING

Advanced nonlinear solver for a discrete fracture modelling

Master thesis of
Kiarash Mansour pour 851598

Supervisor
Prof. Alberto Guadagnini

Co supervisor
Prof. Denis Voskov and Prof. Luca Formaggia

Academic year 2017-2018

Abstract

Simulation of multiphase flow in reservoirs with complex heterogeneous structure requires adopting stable numerical methods that rely on an implicit treatment of the flux term in the conservation equation. Consequently, robust and efficient techniques are needed to solve the governing non-linear system of equations. The solution of the transport problem often requires the propagation of the displacement front to multiple control volumes at each time step. Coping with this issue is particularly challenging in the presence of highly heterogeneous systems such as fractured reservoirs. In this study, we present a nonlinear solver based on a trust region technique and aimed at serving as a general-purpose tool to solve multiphase flows in highly heterogeneous reservoirs. The approach is designed to embed a newly introduced Operator-Based Linearization technique and is grounded on the analysis of multi-dimensional tables related to parameterized convection operators associated with the governing equations. We segment the parameter-space of the nonlinear problem into a set of regions where the convection operators maintain their second order behavior (i.e., they remain either convex or concave). The proposed nonlinear solver locally constraints the updating of the overall compositions across the boundaries of these regions. We enhance the performance of the nonlinear solver by exploring diverse preconditioning strategies. We demonstrate that the initial guess in the nonlinear solution process plays an important role in the heterogeneous settings explored. The proposed strategies of nonlinear solution were tested for various multiphase problems including black-oil and compositional models and considering a variety of combinations of model parameters. In all cases, our approach yields an improved behavior of the nonlinear solution in comparison to state-of-the-art nonlinear solvers.

Extended Abstract

0.1 Introduction

Simulation of multiphase flow in reservoirs with complex heterogeneous structures requires robust nonlinear solvers. The main source of nonlinearity is related to an implicit approximation of flux term in conservation equations which is required for the robustness (unconditional stability) of reservoir simulation process. In this work we discuss the nature of nonlinearities in simulation and solution method that understand them rather than simple naive newton strategy that standard newton method will not converge and it is highly dependent to the timestep selection. We understand how the nonlinearity evolves with enlarging the timestep and thus newton updates overshoot and collapse for a large timestep. Recent Simulator time-step selection is heuristic based on Try-Adapt-Try strategy, which means an attempt to solve for a time-step is made. If that fails within a specified finite amount of time, the time-step is adapted heuristically, and the previous effort is wasted. After analysis of the state of the art for overcoming this issue we develop our implicit transport solver based on the trust region techniques for incompressible two- phase flow in viscous dominate forces in 1D. We obtained unconditionally convergent nonlinear solver with two different newton modification method so called Appleyard chop and inflection point correction. Next, we make comparison between our solvers by changing relative permeability parameters (eg. Mobility ratio and Corey exponent) and see how the flux curve varies and thus we understand inflection point correction works better rather than appleyard chop and appleyard chop going to crash for higher corey exponents. Moreover, we represent the discontinuous relative permeability curves since in real application, we usually measure relative permeability and provide them as tables (standard input in any reservoir simulator). Discontinuous representation of relative permeability simplify by far the Jacobian assembly and finding the inflection point. Moreover, The nonlinear nature of complex flow and transport in porous media requires a linearization of governing equations for the numeri-

cal solution. We present a recently developed linearization strategy, so called (Operator Based Linearization) capable to deal with complex nonlinear problems. We extend our trust region newton method in Operator Based linearization prototype for binary and ternary system. In this case, the inflection point is both the function of Pressure and compositions We analysis our operators for binary and ternary system to find the inflection point(s) based on the linear interpolation of the second derivative and our newton update is a cell-wise chopping strategy guided by trust region of the operators. In our studies we also address the issue related to the slow convergence rate of our newton solver which is one the challenges in current reservoir simulator. We understand that the solution of the transport problem often requires the propagation of displacement front to multiple control volumes per single timestep. This problem became especially serious in the limiting case of heterogeneous property distributions related to fractured reservoirs. The proposed strategies of nonlinear solution were tested for various multiphase problems including Dead oil , Supercritical CO_2 injection and compositional models and considering a variety of combinations of model parameters. In all cases, our approach yields an improved behavior of the nonlinear solution in comparison to state-of-the-art nonlinear solvers.

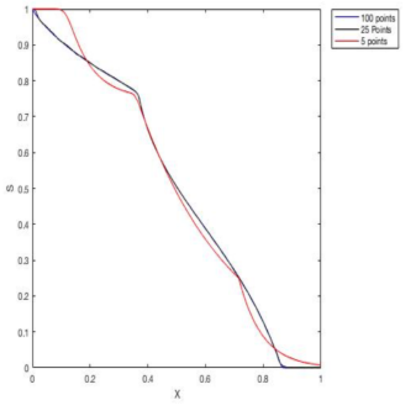
0.2 Methodologies

In this section, we describe how we implement our 1D simulator for two phase immisible viscous dominated fluid.

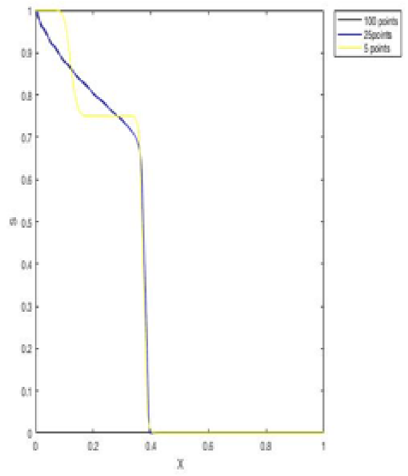
0.2.1 discontinious representation of the relative permeability curves

Before modifying the standard newton method, we represent the discontinuous relative permeability curves since in real application, we usually measure relative permeability and provide them as tables (standard input in any reservoir simulator). In the preliminary analysis, we use uniform mesh with specified number of entries (Nxx = 100). In this case, we evaluate my flux term in each newton iteration as an interpolation from my table and we introduce the derivative as an interpolation coefficient. Next, we analyze how coarsening the table effect the performance of our Newton solver. Here for the relative permeability curve with corey exponent 2 and fixing the mobility ratio (M=1) we experiment the impact of of the resolution in our result.

Table 1 demonstrates by decreasing the number of interpolation



(a) For the timestep equals to 0.2 (dt=0.2)



(b) For the timestep equals to 0.01 (dt=0.01)

Figure 1: coarsening

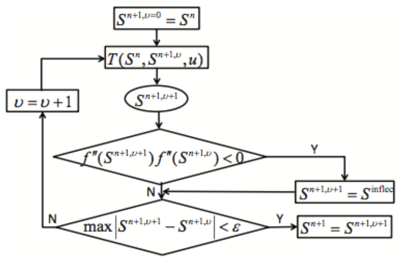
Table 1: Result Of Simulation for timestep=0.2

Number of Grids	Iteration Number	S inflection
500	56	0.5
100	31	0.5
50	23	0.5
25	16	0.4792
5	6	0.625

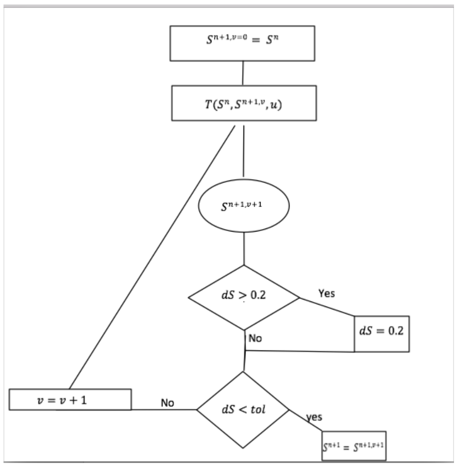
points, the number of iteration decrease although the error is getting higher. Moreover, the preconditioning performance also improves slightly. From the graph, we can see that by dropping the number of points until 25 the saturation curve is not changing a lot until we drop the number of points to 5. In conclusion, by coarsening we expect more linear effect however, in the case of binary system the effect is not that tangible.

0.2.2 Trust Region Newton Solver

Here we introduce two different methods for overcoming the problem related to evolution of nonlinearity with time and get global convergence for the newton method. First method is called Appleyard chop. In Appleyard chop, we added the constraint on the local update not to be bigger than 0.2. Second method is the inflection point correction which we did not let our newton update cross the inflection point. If the newton update cross the inflection point, we land it back to the vicinity of the inflection point. In other words, two successive iteration always have the same concavity.


$S^{n+1,\upsilon=0} = S^n$
$T(S^n, S^{n+1,\upsilon}, u)$
$S^{n+1,\upsilon+1}$
$\upsilon = \upsilon + 1$
$f''(S^{n+1,\upsilon+1})f''(S^{n+1,\upsilon}) < 0$
Y
N
$S^{n+1,\upsilon+1} = S^{\text{inflec}}$
N
$\max |S^{n+1,\upsilon+1} - S^{n+1,\upsilon}| < \varepsilon$
Y
$S^{n+1} = S^{n+1,\upsilon+1}$

(a) Inflection point correction [11]



(b) Appleyard chop

Figure 2: Trust Region Algorithms

0.2.3 Comparison between Appleyad chop and Inflection point correction

We make comparison between our trust region solvers by changing relative permeability parameters (eg. Mobility ratio and corey exponent) and see how the flux curve varies and thus we understand inflection point correction works better rather than appleyard chop and appleyard chop going to crash for higher corey exponents. The flux function varies significantly with the mobility ratio when the exponent is low and quite mildly when exponent is higher. The higher the mobilty ratio the less favour is propagation and thus higher number of iteration is expected.The higher the exponent the less disperse is our flux. We understand that the appleyard chop going to diverge for high corey exponents. Basically, the reason Appleyard chop works perfectly in the previous case is that we artificially do not let our update to be outside of the trust region. However, by increasing the parameter in a way the inflection point moves and by updating we are overshooting it even outside then the Appleyard chop does not work anymore.

0.2.4 Design of the better nonlinear solver and Preconditioning

We understand that the solution of the transport problem often requires the propagation of displacement front to multiple control volumes per single timestep. This problem became especially serious in the limiting case of heterogeneous property distributions related to fractured reservoirs. In fracture reservoir, the saturation front is moving slowly in the matrix and suddenly it reaches the fracture and it propagates. To effectively, imitate this process in our 1D simulation code instead of increasing the permeability we enlarge the timestep. We tackle the problem due to the slow wave propagation by introducing preconditioning strategy. We analyse linear and nonlinear relative permeability curves and understand that if the derivative of the flux is positive, The newton iteration yields a saturation distribution that is monotonic and positive. Therefore, to allow for maximum propagation of the saturation waves downstream, we make the saturation in two successive control volumes as close as possible. By analysis wave speed for relative permeability curves we understand that using the inflection point as our initial guess since the absolute value of the derivative is maximum we maximum the propagation of the saturation downstream. Table 2 illustrated one of the example of precond in the fracture reservoir. We first ran simulation until particular time (t = 0.3 *d*) with a small ltimestep 0.001 *d* and save this solution. Next we ran for onetime step only enlarging from 0.001 *d* and finishing with 0.2 *d* and count Newton iterations which required for this one timestep. Here we just report the performance for the timestep equals to 0.2*d* with and without preconditioning.

M	Different Preconditioning		
	Without Precond.	Initial guess= Not initial cond	With Precond.
0.5	198	27	10
1	204	27	12
10	229	30	11

Table 2: Performance comparison (number of iteration) between different nonlinear solvers, with and without preconditioning, for 1D transport under viscous forces. the exponents of the Rel.Perm is 10

0.2.5 Trust Region newton method based on Operator Based Linearization

This approach approximates exact physics of simulation model which is conceptually similar to an approximate representation of space and time discretization performed in conventional simulation. In this approach, The governing equations are introduced as a combination of operators, dependent on spatially altered properties and operators, fully controlled by nonlinear properties of fluid and rock. Next, a parametrization in the physics space of the problem is introduced. The property- based operators are approximated using direct interpolation in the space of nonlinear unknowns. The discrete version of the governing equations is constructed as a combination of operators that approximate both nonlinear physics and discretization in time and space. This approach is applied to the reservoir simulation of miscible and immiscible displacement processes. Furthermore, We extend our trust region newton method in Operator Based linearization prototype for binary and ternary system. In this case, the inflection point is both the function of Pressure and compositions We analysis our operators for binary and ternary system to detect the inflection point(s) based on the linear interpolation of the second derivative. After finding the inflection, our newton update is a cell-wise chopping strategy guided by trust region of the operators.

0.3 Verification for realistic examples in the 1D simulation prototype in OBL

0.3.1 Binary System

In this case, we have two independent variables [P, Z]. To find inflection point(s) of the convection operators, for a fix pressure we find the inflection point. Next, updated the pressure and find the other inflection point in the case it exists. The analysis of finding the inflection point is based on the linear interpolation of the second derivative.

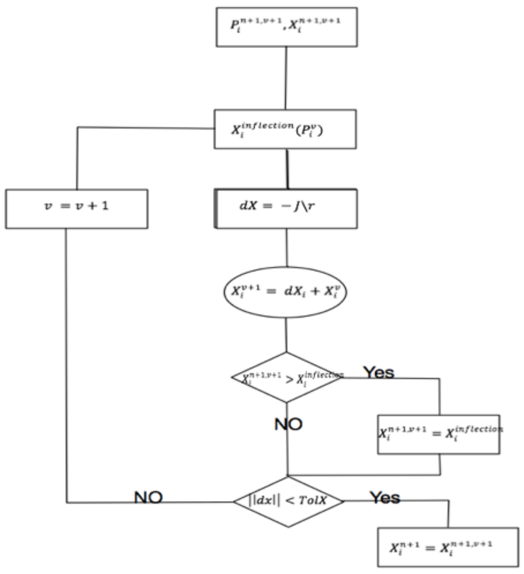

$P_i^{n+1,v+1}, X_i^{n+1,v+1}$
$X_i^{inflection}(P_i^v)$
$v = v + 1$
$dX = -J\backslash r$
$X_i^{v+1} = dX_i + X_i^v$
$X_i^{n+1,v+1} > X_i^{inflection}$
Yes
NO
$X_i^{n+1,v+1} = X_i^{inflection}$
NO
$||dx|| < TolX$
Yes
$X_i^{n+1} = X_i^{n+1,v+1}$

Figure 3: Flowchart of the modified newton method for one time step considering also pressure update and correction of inflection point for pressure

0.3.2 Dead Oil

Fig. 5 summarize the performance of different solvers for the fracture model and it is clear that the trust region solver with preconditioning works significantly better than the other solvers. Note that since the preconditioning strategy only provides initial guesses for the saìturation solution, when it is used in the fully-implicit method, only the initial guesses for saturation variables are modified to be the inflection point, the initial guesses for the pressure variables are unaffected.

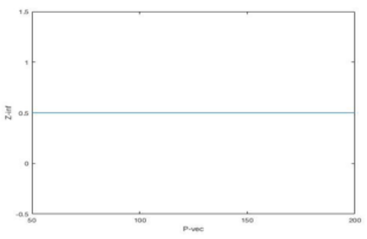


Figure 4: Inflection point of our beta operator for different pressure

0.3.3 Viscosity variation with Pressure

Here for the dead oil example, we make the case where the inflection point varies also with the pressure for the beta operator. To do so we

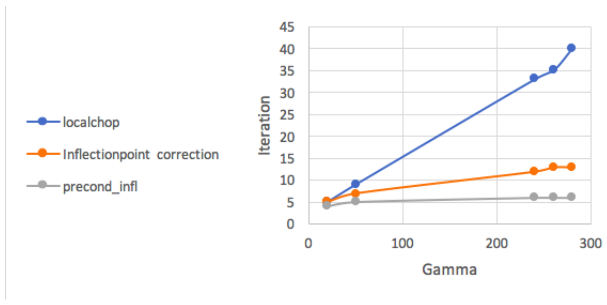


Figure 5: Comparison of different solvers with Corey exponents 2 2 and the resolution with 32 OBL resolution and the oil viscosity 1.5cp, water viscosity 1cp.

make the viscosity a function of pressure and thus the inflection point varies for different pressure.

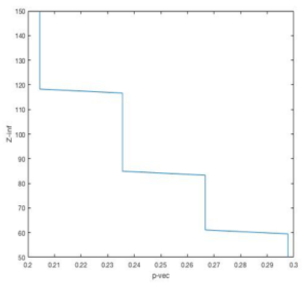


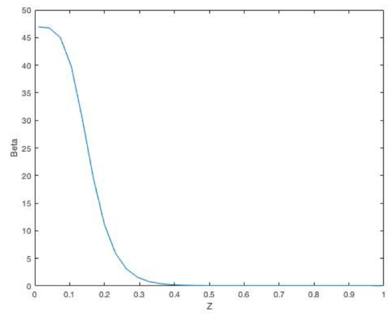
Figure 6: Inflection point variation with Pressure for dead oil with variation of viscosity with pressure example

In this case, Alpha operator linearly changing with composition for all the pressure and thus never introduce problem for our trust region solver. We compare the performance of different solvers in the fracture reservoir in this case. To imitate it, i ran the simulation until a particular time ($T = 5000 days$) Next, only by enlarging one timestep we compare the behavior of our different solvers. In the next step by introducing the infection point for a preconditioning we enhance the performance even further. Table 3 illustrates the different behavior of our solver.

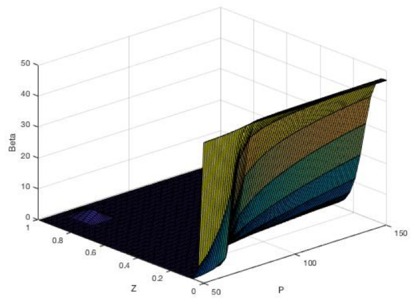
Table 3: Performance behavior of the different solvers in the case of dead oil for fractured reservoir

Gamma	Global chop	Trust Region solver	Trust region with precond
20	3	4	4
50	6	5	5
100	10	6	5
280	15	8	5
300	16	8	5

0.3.4 Supercritical CO2 injection

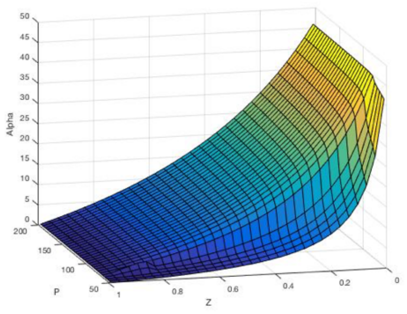


(a)

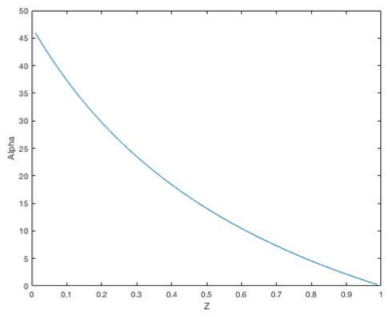


(b)

Figure 7: (a) Beta operator variation with composition given the fix Pressure (b) Beta Operator



(a)



(b)

Figure 8: (a) Alpha Operator (b) Alpha operator variation with composition given the fix Pressure

From the operators graph, it is clear that the alpha operator is not anymore linearly change with composition since the density is changing but still it keeps its second order behavior. On the other hand, Beta operator seems to be problematic due to the presence of inflection point. Therefore, we make our trust region solver in a way it does not cross its inflection point.

Table 4 shows the comparison between different nonlinear solvers, trust region solver, global chopping and local chopping with $dx = 0.1$ for a fracture reservoir while I ran the simulation for the maximum time 6000 days with the timestep = 10 days imitating the fluid flow moving in the matrix.

Table 4: Performance Of Different Solvers (running the simulation until 6000 days with the small timestepeuqals to 10days)

Solver	Total number of Newton iteration
Inflection point correction	838
Global chopping	1152
Local chopping (dx=0.1)	Diverge

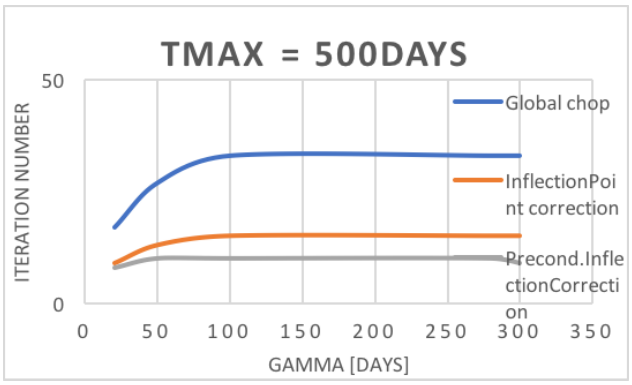


Figure 9: Inflection point variation of the operator beta with pressure in the case of sCO2 injection

Gamma	Global chop	Trust Region Solver	Trust Region with precond
20	17	9	8
50	27	13	10
100	33	15	10
280	33	15	10
300	33	15	9

Table 5: sCO2 injection

As we can see the trust region works significantly better than the Global chop.

Graph 9 and table 5 illuatrate the comparison of fractured reservoir in the case of sCO2 injection. Moreover, the performance of the trust region solver enhances further by preconditioning.

0.3.5 Three component case

Table 6 shows the comparison between different nonlinear solvers, trust region solver, global chopping and local chopping with $dx = 0.1$ for ternary system while i ran the simulation for the maximum time 10000 days with the the aggressive timestep = 500 days.

Table 6: Performance of different solvers in three components case. (Running the simulation until 10000 days with the very aggressive time step 500 days)

Advanced nonlinear Solver	Total number of Newton iteration
Inflection point correction	154
Global chopping	201
Local chopping (dx= 0.1)	205

0.4 Conclusion

We developed the advanced nonlinear solver for fractured reservoirs in 1D. This solver is developed based on the trust region technique to get the global convergence and then we design our solver better by applying preconditioning strategy to get the better performance of the trust region solver. Moreover, introducing our relative permeability in the table and interpolate it rather than analytical formula we make our solver more real applicable since in the industry the relative permeability is estimated. Next, we analyze the coarsening and resolution on the performance of the trust-region solver. Finally, we extend our frame work in the simulation prototype based on OBL (Operator based linearization). In this case, we consider more general situation by coupling the transport and flow and extend it to the compositional flow problems.

The future work can be in a way that we find the inflection point adaptively rather than preliminary analysis. because of hyperbolic nature of overall composition, the vast majority of parameter space remains unused. In other words in the newton update we find the inflection point and check trajectory whether pass it or not. Adaptive Parameterization with buoyancy has been shown by Voskov and Khait

[28] rather than preprocessing and storing OBL tables. The total size of the interpolation tables is defined by the number of dimensions N and the number of interpolation points n as n^N. Although the dimensionality depends on the number of components and thermal assumptions in a problem of interest, the number of interpolation points corresponds to the desired accuracy of the physical representation. Therefore, parameterization at the preprocessing stage would require a substantial amount of memory for the multicomponent systems modeled at a high interpolation precision. Furthermore, the necessity of searching supporting points (i.e., operator values) for every interpolation in a huge array of data affects the performance of the simulation. adaptive parameterization in space with adaptively finding the inflection point is one of our interest due to the reduce of the cost of the data storage.

Another future work will be related to extension of our preconditioning strategy to compositional problems unconditional to the number of components.

Acknowledgement

I would first like to thank my thesis advisor Dr. Denis Voskov at Delft University of Technology for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of my work. I am very fortunate to have him as my advisor to introduce me to this scientific research.

I would also like to thank my advisor in Politecnico di Milano Professors Alberto Guadagnini for giving me this opportunity to work with him and for his continuous support of my thesis.

Special thanks to my advisor Professor Luca Formaggia for importing his knowledge in this study. The door of his office was always open whenever i had a question about my research.

Finally, I must express my very profound gratitude to my parents and my sister with unfailing support and continuous encouragement throughout my years of study. Special thanks to all my dear friends that i made in Politecnico di Milano that are not afraid of dreaming big and unconditionally support me.

Contents

List of Figures

List of Tables

List of Symbols

J	Jacobian
K	Effective permeability tensor
R	Residual
S	Saturation
S_{nwr}	Non-wetting phase residual saturation
S_{wr}	Wetting (water) phase residual saturation
λ_j	Phase mobility
ω	Physical properties
ϕ	Effective rock porosity
ρ_p	Phase molar density
ξ	Spatial properties
c	Dimensionless timestep size
f	Wetting phase flux
n_c	Number of compoents
n_p	Number of phases
p_c	Capillary pressure
x_{cj}	component mole fraction in a phase

List of Abbreviations

ADGPRS	automatic differentiation based general purpose research simulator
AIM	Adaptive implicit method
CFL	Courant-Friedrichs-Lewy number
CPR	Constraint pressure residual
DFM	Discrete Fracture Model
FIM	Fully implicit method
IMPES	Implicit pressure Explicit saturation
IMPSAT	Implicit pressure and saturation and explicit mole fraction
NR	Newton Raphson
TPFA	Two point flux approximation

Chapter 1

Introduction

1.1 Discrete fracture model (DFM)

The presence of fractures and faults within geological formations can have a major impact on fluid flow and mechanical behavior of the rock. The detailed understanding of such systems is of interest in a variety of engineering fields. In the environmental sector, for example, typical applications are aquifer management, underground waste disposal, and CO2 sequestration. In the energy sector, oil and gas recovery, and the exploitation of geothermal reservoirs, often involve flow in fractured systems.

In DFM approach each fracture is modelled explicitly using, in most of the cases, highly resolved unstructured grids[12]. This allowed the simulation of fine scale geological models with complex and various fracture geometries. For these reasons, DFM is considered as the most accurate representation of fracture networks but with the disadvantage of high computational cost as a tremendous amount of grid cells are involved. In this work, we address the cause of these high computational cost and tackle them.

1.2 Reservoir simulation

A reservoir is a porous medium that contains hydrocarbons. The primary goal of reservoir simulation is to predict future performance of a reservoir and find ways and means of optimizing the recovery of some of the hydrocarbons [1].

The two important characteristic of a petroleum reservoir are the natures of the rock and the fluids filling it. A reservoir is usually heterogenous; its properties heavily depend on the space location. A fracture reservoir is considered as highly heterogenous, for example it consist of a set of blocks of porous media (the matrix) and a net of fractures. The rock properties in such a reservoir dramatically change; its permeability may vary from one millidacry(md) in the matrix to thousands md in the fractures. While the governing equations for the fractured reservoir are similar to those for an ordinary reservoir are similar to those for an ordinary reservoir. In general, the equations governing a mathematical model of a reservoir cannot be solved by analytical methods. Instead, a numerical model can be produced in a form that is amenable to solution by digital computers. Since the 1950s, when digital computers became widely available, numerical models have been used to predict, understand, and optimize complex physical fluid flow processes in petroleum reservoirs. The most frequently used approaches for reservoir simulation are natural [4] and molar formulations. Usually, the natural formulation in combination with different by passing strategies for a stability test [23] performs better in immiscible gas displacement, whereas the molar formulation does better for miscible gas injection [26]. It was shown recently that some specific treatment on phase appearance or disappearance may help to imrove the nonlinear behavior of the natural formulation in the miscible regimes. [3]

1.3 Nonlinearities

Multiphase, Multicomponent, flows through subsurface porous media couple several physical phenomena with vastly differing characteristic scales. Moreover, scale separation is not always apparent. The fastest processes such as component phase equilibria occur instantaneously and subsequently they are modeled using nonlinear algebraic constraints. Mass conservation laws govern the transport of chemical species propagating through an underlying flow field. These transport phenomena are near-hyperbolic. In the limit of low capillary numbers, the transport equations are purely advective and they give rise to an evolution with a finite domain of dependence. In the other limit, the transport problem is diffusive. Moreover, the underlying flow field itself is also transient, and it evolves with parabolic character. In the limit of no total compressibility, the flow field reaches instantaneous equilibrium, and is governed by an elliptic equation. Constitutive relations such as that for the velocity of a phase couple the variables across governing equations in a strongly nonlinear manner. Consequently, one challenge in modelling large scale flows through general porous media is in resolving this coupling without sacrificing stability. In the fully-coupled, fully-implicit approach, while there are no stability restrictions in the sense of the discrete approxiations, the resulting algebraic nonlinear stystems are difficult to solve.

1.3.1 Implicit approach and nonlinearity in timestepping

One attractive aspect of this approach is its unconditional stability which is obtained at the cost of tight nonlinear coupling between fundamentally differing phenomena; e.g. parabolic and hyperbolic components. Two practical shortcomings of this approach continue to receive attention from the research community (see for example [14] [9].)

The first is that available solution methods for the discrete nonlinear systems may themselves not be unconditionally convergent. The second aspect is that regardless of the technical details of the particular solution method, the computational effort required to solve large couple systems can be significantly larger than that for de-couped, localized computations, such as in a convergent step of a semi-implicit method. The practical implication of these two shortcomings is the use of time-step chops. With a try-Adapt-Try strategy, an attempt to solve for a time-step is made. If that fails within a specified finite amount of time, the time-step is adapted heuristically, and the previous effort is wasted.

Current simulators rely on a fixed-point iteration, such as a variant of Newton's method in order to solve these problems. (See for example [1] [7] [20]). For general problems, Newton's method is not guaranteed to converge, and it is known to be sensitive to the initial guess, which must be supplied somehow. In most reservoir simulators, the initial guess to the iteration is the old state. For small timestep sizes, this is a good approximation to the new state, and is therefore likely to be a good starting point for the Newton iteration. For larger time steps, however, this is less likely to be the case, and the iteration may converge too slowly, or even diverge. As we can see from the figure 1.1, since the concavity of the flux function is changing our newton iteration literally oscillating around this inflection point and thus the divergence happens.

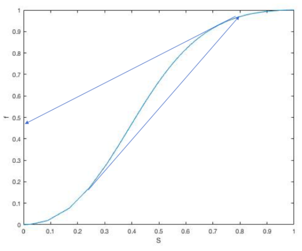


Figure 1.1: Oscilating around inflection point

1.4 Mathematical Modelling

We consider nonlinear immiscible, incompressible, two-phase flow in porous media. The conservation law for the two phases - referred to as non wetting and wetting - can be written as:

$$\phi\frac{\partial S_w}{\partial t} + \nabla.u_w = q_w \tag{1.1}$$

$$\phi\frac{\partial S_n}{\partial t} + \nabla.u_n = q_n \tag{1.2}$$

where ϕ is the porosity of the medium. We use subscript α to denote the phases, i.e., w or n. S_α is the saturation, u_α is the velocity, and q_α is the source term. The phase velocity is given by Darcys law:

$$u_\alpha = -K\frac{K_{r\alpha}}{\mu_\alpha}(\nabla p_\alpha + \rho_\alpha g\nabla h), \alpha = w, n \tag{1.3}$$

where p_α the pressure, ρ_α is the mass density, h is the height, $K_{r\alpha} = K_{r\alpha}(S_\alpha)$ is the relative permeability, andμ_α is the viscosity.

We define the phase (relative) mobilities as $\lambda_\alpha = \frac{K\alpha}{\mu\alpha}$ The end-point mobility ratio is defined as

$$M^0 = \frac{\frac{K^0_{rw}}{\mu_w}}{\frac{K^0_{rn}}{\mu_{nw}}} \tag{1.4}$$

where $K^0_{rw} = K_{rw}(1 - S_{nr})$ and $K^0_{rn} = K_{rn}(S_{wr})$ are the maximum (end-point) relative permeabilities.

Substituting the phase velocities into 1.1 and 1.2, we obtain two mass-balance equations with four unknowns: $S_w, S_n, p_w,$ and pn To close the system, we add the following relations:

$$S_w + S_n = 1, \text{ (saturation constraint)} \tag{1.5}$$

$$p_n - p_w = p_c, \text{ (capillary-pressure relation)} \tag{1.6}$$

Where the capillary pressure $p_c = p_c(S_w)$ is a nonlinear function of the wetting saturation.

Substitution of 1.3 into 1.1 and 1.2 yield a coupled system of nonlinear parabolic equations. The system often exhibits a mixed near elliptic-hyperbolic charecter, which becomes apparent when we sum up 1.1 and 1.2 to obtain the pressure equation. We can use the saturation constraint in the summation of equations 1.1 and 1.2 to get:

$$\bigtriangledown .u_t = q_t \tag{1.7}$$

where

$$u_t = -K\lambda_t \nabla p_w - kg(\lambda_w \rho w + \lambda_n \rho_n)\nabla h - k\lambda_n \nabla p_c \tag{1.8}$$

and

$$q_t = q_w + q_n \tag{1.9}$$

Here, $\lambda_t = \lambda_w + \lambda_n$ is the total mobility. Substitution of 1.8 into 1.7 gives the pressure equation, which is an elliptic PDE. In the absence of source/sink terms, 1.7 indicated that the total-velocity U_t is divergenefree, i.e.,

$$\nabla .u_t = 0. \tag{1.10}$$

We can rewrite the wetting-phase velocity in terms of the total-velocity as

$$u_w = \frac{\lambda_w}{\lambda_t} u_t - kg\frac{\lambda_w \lambda n}{\lambda t}(\rho_w - \rho_n)\nabla h + k\frac{\lambda_w \lambda n}{\lambda t}\nabla p_c \tag{1.11}$$

In 1.11 if the total-velocity is constant, $u_w = u_w(S_w)$ is then a function of saturation only. Substituting1.11 into 1.1, the transport (saturation) equation is obtained as:

$$\phi\frac{\partial S_w}{\partial t} + \nabla .(\frac{\lambda_w}{\lambda t} u_t - kg\frac{\lambda_w \lambda_n}{\lambda_t}(\rho_w - \rho_n)\nabla h + k\frac{\lambda_w \lambda_n}{\lambda_t}\nabla p_c) = 0; \tag{1.12}$$

Subject to proper initial and boundary conditions, the saturation distribution can be obtained by solving this PDE. Combining the pressure equation 1.7 with the transport equation 1.12, we obtain the flow-transport system for immiscible, incompressible, two-phase flow.

For flow in one dimention (1D) and assuming that the total velocity, u_t, is constant, the transport equation can be written as

$$\phi\frac{\partial S_w}{\partial t} + u_t\frac{\partial F_w}{\partial x} = 0... \tag{1.13}$$

Here, $F_w = \frac{u_w}{u_t}$ is the flux (fractional flow) of the wetting phase. It is defined as

$$F_w = \frac{\lambda w}{\lambda t} - kg\frac{\lambda w\lambda n}{\lambda t}(\rho_w - \rho_n)\frac{\nabla h}{u_t} + k\frac{\lambda w\lambda n}{\lambda t}\frac{\nabla p_c}{u_t} \tag{1.14}$$

We introduce two dimensionless quantities:

$$N_g = \frac{k(\rho_w - \rho_n)g\nabla h}{\mu_n u_t}, \tag{1.15}$$

and

$$Pe = \frac{u_t\mu_n L}{kp_c} \tag{1.16}$$

Where N_g is the gravity number, which is the ratio of buoyancy to viscous forces. Here we assume that the z coordinate is pointing upward, and use h to denote the height. Therefore we have $\Delta h > 0$. We also assume that the wetting phase is heavier, i.e., $\rho_w > \rho_n$. P_e is the Peclect number, which is the ratio of viscous to capillary forces. L is a charecteristic length scale, and p_c is a charecteristic capillary pressure. With these definitions, F_w can be written as:

$$F_w = \frac{\lambda_w}{\lambda_t} - \frac{k_{rn}\lambda_w}{\lambda_t}N_g + \frac{\lambda_w k_{rn}}{\lambda_t}\frac{\Delta p_c}{p_c/L}\frac{1}{Pe} \tag{1.17}$$

The three terms on the tight-hand side account for the viscous, buoyancy, and capillary fluxes, In the absence of gravity and capillarity, $F = \frac{\lambda_w}{\lambda_t}$ and is a monotonic function of saturation. Since the total-velocity is assumed constant, we can make the transport equation dimensionless by defining

$$t_D = \frac{tu_t}{\phi L} \tag{1.18}$$

$$x_D = \frac{x}{L} \tag{1.19}$$

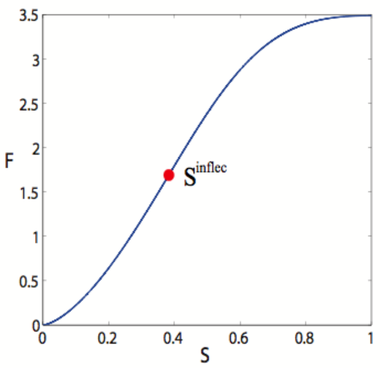
Then, the 1D dimensionless transport equation is

$$\frac{\partial S_w}{\partial t_D} + \frac{\partial F_w}{\partial x_D} = 0 \tag{1.20}$$

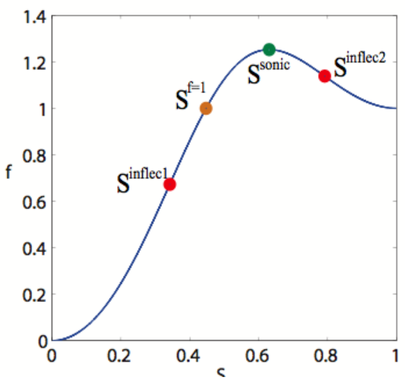
From here on, we drop the subscripts, so we can write the equation in a more concise form:

$$\frac{\partial S}{\partial t} + \frac{\partial F}{\partial x} = 0 \tag{1.21}$$

For multiphase flow in porous media, the flux function, $F = F(S)$ in the absence of gravity and capilarity is usually S-shaped [1]. The presence of gravity, or capillarity, change the shape of the flux function. When buoyancy is dominant, the flux function becomes non monotonic indicating the occurance of counter-current flow for part of the saturation range. Note that the convexity of the flux function changes as N_g and Pe change. Note that the convexity of the flux function changes as N_g and Pe change.



(a)



(b)

Figure 1.2: (a) Viscous and Capillary flux ($M^0 = 1, N_g = -5, Pe = 0.2$). (b) Viscous and buoyancy flux From Wang [30]

In fig. 1.2 (b). There are two inflection points (red dots), a sonic point (green dot), and a unit-flux point (orange dot). The sonic point iswhere the flux funtion is maximum; the unit-flux point is where the flux is unity at a point $S < 1$. Fig 1.2 (a) shows that the presence of strong capillary forces (e.g., $P_e = 0.2$) changes the shape, including the inflection point, of the flux function. This is due to the nonlinear diffusion term, i.e., $\Delta p_c(S_w)$ in the third term of F_w introduced into the phase flux by capillarity. This diffusion term tends to mix the two fluid phases together if there is a saturation gradient.

In this work we just focus on the viscous dominated forces flow.

1.5 Different model formulation considering the implicit level range

In this case we demonstrate the explicit time integration of the Eq. 1.21. This approach allows us to simply "march" from time t_n to t_{n+1} by explicitly solving for the only unknown at the new time level.

$$\frac{S_m^{n+1} - S_m^n}{\Delta t} + \frac{f(S_m)^n - f(S_{m-1})^n}{\Delta x} \tag{1.22}$$

Explicit time integration offers accuracy a computational efficiency as long as the limit on the stable time step size is not a major concern. In some problems, however, the time step restriction associated with an explicit scheme is quite severe, and the use of implicit schemes is necessary. In reservoir simulation problems, where the evolution of the saturation field in the geologic porous formation, as a function of space in time is sought, it is often the case that for a given global time step size the CFL numbers in the computational domain can vary by orders of magnitude- In such cases the use of explicit time integration schemes is simply not feasible, and implicit time integration is required. In other words, the use of explicit time integration schemes often leads to severe restrictions on the timestep size [1, 5]. When a large-scale heterogeneous reservoir model is simulated, it is often the case that for a given global timestep size, the CFL numbers in the computational domain can vary by orders of magnitude.

For compositional models, the implicit level (number of implicit variables per gridblock) ranges from one in IMPES (Implicit pressure explicit saturation) to n_c in fully implicity model (FIM) [2]. The IMPES formulation treats the interblock flowrates implicitly in pressure, but explicitly in saturations and compositions. Stability increase by increasing the implicit level of the model. However, the computational cost per newton iteration also increase dramatically. To make a simulation run fast, we need to reduce the number of unknowns per grid block from n_c. This number is minimum for IMPES and maximum for FIM . Cao [2] proposed IMPSAT model, where only pressure and saturation are treated implicitly, and all of the component mole fractions are treated explicitly. Since one of the saturations can be removed using the saturation constraint explicitly, we only have np unknowns per gridblock. In reservoir simulation typically we have two or three phases, so the number of unknowns per gridblock for the IMPSAT model is three or less. Therefore IMPSAT model has the potential to yield big savings in computational cost per linear solver, especially for problems with large number of components.

AIM: The adaptive implicit method [25] uses different levels of implicitness in different blocks. In each gridblock, each of the n variables may be chosen explicit or implicit independent of the choices in other grid blocks. The choices may change from one timestep to the next. AIM offers a balance between FIM and IMPES by employing implicit treatment only when and where necessary.

Furthermore, Cao [2] proposed IMPSAT Based AIM Model for large and difficult problems. Especially for problems with a large number of components. This model basically add the IMPSAT model into the traditional AIM model [8]. In conclusion, FIM models are the most stable, and they can use large timestep sizes, but the cost of each newton iteration is also high, especially for problems with large number of components. IMPES models are least stable, because only pressure is treated implicitly, but their cost is the lowest for each Newton iteration. The IMPSAT model is much more stable than the IMPES model, due to the implicit treatment of saturation. AIM models are more efficient than the FIM model, since high flow rates are generally restricted to a small portion of the reservoir and the FIM model is only needed in the regions of high flow rates.

1.6 Thesis Outline

This thesis is organized as follows. In chapter 2, we review the literature on nonlinear solvers of multiphase flow in porous media. In chapter three we develop our advanced nonlinear solvers for fractured reservoir based on trust region technique and discontinuos representation of relative permeability and make comparison between the performance of different solvers. In chapter 4, we propose a preconditioning strategy to overcome convergence difficulties in the nonlinear solver that are associated with the propagation of saturation fronts into the fracture reservoir and make comparison between the performance of the solver with different preconditioning strategy. In chapter 5, we adapt our trust region nonlinear solver in the simulation prototype based on Operator based linearization (OBL) [27].

Chapter 2

State of the art

We are dealing with the nonlinearty stems from dicretization of the governing equation in a fully implicit method (FIM). Due to the intrinsic nonlinearity nature of the equation, Newton's method is not guaranteed to converge, and it is known to be sensitive to the intial guess [1, 19, 7]

In reservoir simulators, the intial guess is the old state(i.e., the pressure and saturation distributions from the previous timestep). For small timestep sizes it is a good approximation and the newton solver is likely to converge. However, by increasing the timestep and thus nonlineairty we see that our standard newton method blows out.

What is usually done to overcome this convergence difficulties, emprical techniques for timestep control are utilized in reservoir simulator [1, 2] and do the timestep chops. With a try-adapt-try strategy, an attempt to solve for a time-step is made. If that fails within a specified finite amount of nonlinear iteration, the time-step is adapted heuristically, and the previous effort is wasted. Current simulators rely on a fixed-point iteration, such as a variant of Newton's method in order to solve these problems.

In this chapter, we review the state-of-the-art in reservoir simulation practice.

2.1 Modified newton method

To get the global convergent which is unconditional to our intial guess and the timestep, Jenny et al [11] proposed a modified Newton method for hyperbolic conservation laws in the absence of gravity and capillarity. They proposed a simple chopping scheme within the Newton loop results in a nonlinear solver that is convergent for arbitrarily large timestep sizes, and hence allowing one to choose the timestep size based on accuracy considerations only. A brief description of the Jenny et al [11] method follows that we also use for developing our own solver priliminary:

The degree of nonlinearity of the residual for the transport problem, Eqn. 1.21, is strongly related to the shape of the flux function, expecially when the timestep is large. For viscous-dominated multiphase flow, the nonlinear flux function is usually S-shaped Fig. 1.1. The inflection point is the saturation where the flux function has the largest slope. As expected, if the solution (saturation) resides on one side of the inflection point, whereas the initial guess is on the other side, it can be hard for the Newton iterative process to converge. What Jenny proposed [11] is the following: If the Newton update would cross the inflection point, it is scaled back to the inflection point, and the Newton process is continued based on the scaled back (chopped) update. Note that the Newton update is not scaled back exactly to the inflection point. Instead, it is scaled back to one side of the inflection point, i.e., $S_{\text{inflec}} \pm \epsilon$, to make sure that the two successive updates reside on the same side of the inflection point. The flow chart of this modified Newton method is presented in the following diagram:

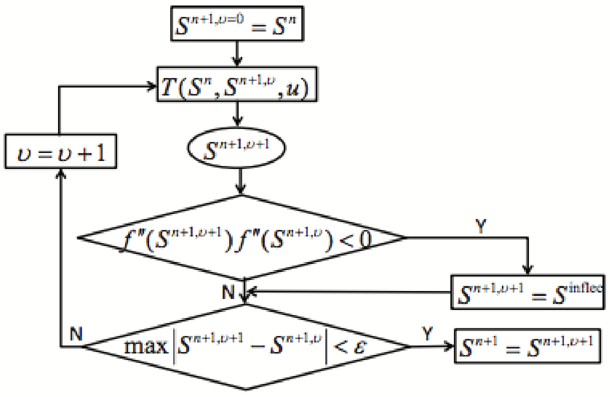

$S^{n+1,\upsilon=0} = S^n$
$T(S^n, S^{n+1,\upsilon}, u)$
$\upsilon = \upsilon + 1$
$S^{n+1,\upsilon+1}$
$f''(S^{n+1,\upsilon+1}) f''(S^{n+1,\upsilon}) < 0$
Y
N
$S^{n+1,\upsilon+1} = S^{\text{inflec}}$
N
$\max \left| S^{n+1,\upsilon+1} - S^{n+1,\upsilon} \right| < \varepsilon$
Y
$S^{n+1} = S^{n+1,\upsilon+1}$

Figure 2.1: Flow chart of the modified Newton method of Jenny et al [11] for one timestep. Solution of the linearized transport equation is represented by the operator T

This flux-based Newton method has proved to be quite powerful for the simulation of immiscible viscous-dominated displacements in large-scale heterogeneous models. Its applicability and efficiency for general-purpose compositional simulation was demonstrated by Voskov and Tchelepi in [29] ,who extended the approach to solve compositional problems that employ the molar (or mass) variables. They showed that the flux functions associated with the key tie-lines play a dominant role in the evolution of the solution. For a four-component gas injection problem in the top eight layers of the SPE 10 model without gravity, the modified flux-based Newton scheme is shown to be always more stable and converges faster than the safeguarded Newton method, which employs heuristics on maximum changes in the variables. The gas saturation at the end of the simulation for immiscible gas displacement is shown in fig 2.2

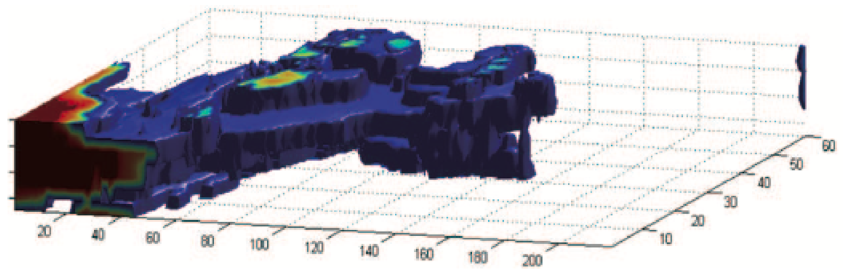


Figure 2.2: FGas saturation for immiscible gas displacement (from Voskov and Tchelepi

2.2 Modified newton method taking into account capillary and gravity

Wang [30] described a nonlinear solution algorithm for coupled flow and transport in heterogeneous porous media where both the viscous and buoyancy forces are significant. The solver employs trust regions of the flux function to guide the Newton iteration. The delineation of the trust-region boundaries was detected by the unit-flux and inflection points of the flux function. the unit-flux and inflection points are available before solving the transport problem, since for any given total velocity, the flux function depends on the mobility ratio and the gravity number.

The trust-region nonlinear solver is essentially a multi-point chopping strategy. That is, at the end of each Newton iteration, we do not allow two successive updates to cross any of these critical points.

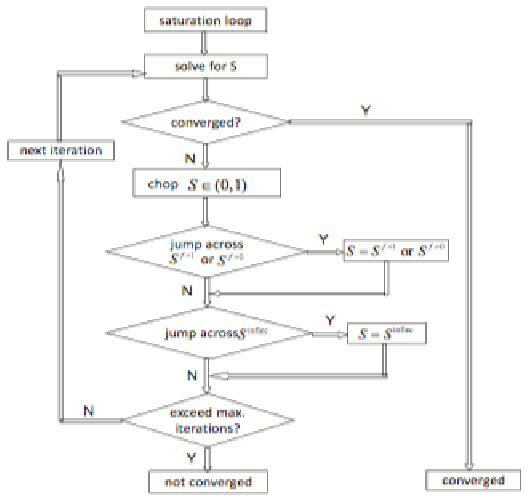


Figure 2.3: The Flow chart of the trust-region Newton scheme in the presence of viscous, gravitational and capillary forces, for one timestep.

2.3 Continuation-Newton method

Continuation (homotopy, or embedding) methods [13] are nonlinear solvers that associate a timestep with each iteration. These approaches converge when the residual drops below a certain threshold and the associated timestep reaches the target timestep.

Younis et al. [32] developed a Continuation-Newton (CN) method that solves the implicit residual system using a combination of the Newton method and continuation on the timestep size. n [32], a continuation-based solution process that associates a timestep size with each iteration is formulated, i.e., the time step size is a parameter which is continuously changing. The CN method of Younis et al. follows the solution path loosely. A more detailed description of CN follows: In the nonlinear problem

$$R(S^{n+1}, \Delta t; S^n) = 0 \tag{2.1}$$

R is the vector of discrete residual equations and S is the saturation. The solution path can be written as:

$$S^{n+1} = S^{n+1}(\Delta t), \tag{2.2}$$

which is continuous and emanating from the condition at the previous timestep S^n.

For the solution path, we have

$$\frac{dS^v}{d\Delta t} = -J(S^v, \Delta t; S^n)^{-1}\frac{\partial R(S^v, \Delta t; S^n)}{\partial \Delta t} \tag{2.3}$$

i.e., the tangent of the solution path is known.

An illustration of solution path with only one unknown is shown in Fig. 2.4. In Fig. 2.4, the solution path emanates from the initial condition ($S = Sn, \Delta t = 0$), and continues to the target time step, Δt, augmented with its solution, S^{n+1}.

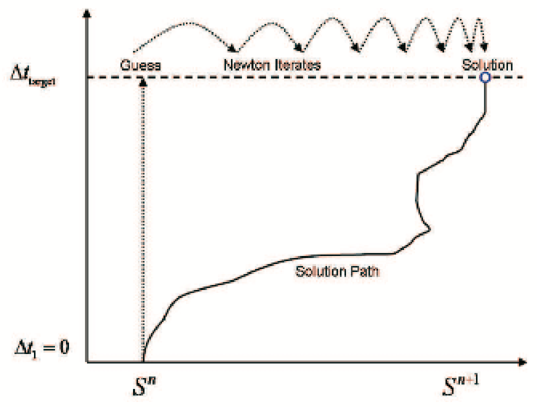


Figure 2.4: Illustration of solution path and iterative solutions for Newton method (from Younis [32]). Note the iterative solutions are evaluated at the target timestep.

The proposed algorithm in[26] and [31] defines a convergence neighborhood around the solution path (illustrated in Fig. 2.5). Any point inside the neighbor- hood is considered to be a good estimate of the solution.

In Fig. 2.5, it is shown that for each iteration in CN, the solution is obtained either by a tangent prediction (e.g., from $p0$ to $p1$, or from $p1$ to $p2$) or by a (Newton) correction (e.g., from $p2$ to $p3$).

For each tangent step, the step length is chosen, such that the next solution remains within the convergence neighborhood. A correction step is triggered in order to bring the solution closer to the solution path, if the next tangent step-length would be too small, or zero.

The algorithm guarantees convergence for any timestep size. If the iteration process is stopped before the target timestep is reached, the last iterate is a solution to a smaller and known timestep.

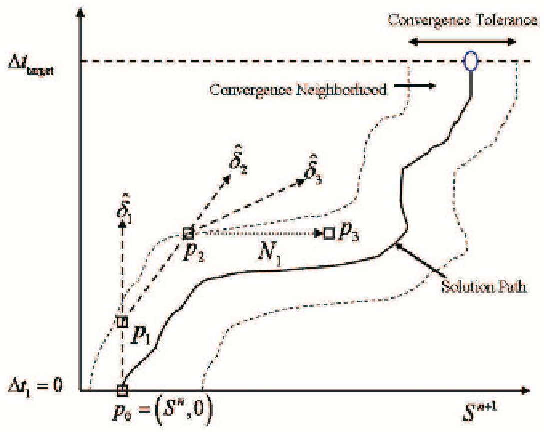


Figure 2.5: Illustration of iterative solutions using the continuation-Newton approach (from [31]). In the illustration, two tangent steps are followed by a correction step.

2.4 Ordinary trust-region method: NLEQ-RES algorithm

The standard Newton method can be algebraically derived by linearization of the nonlinear equation around the solution point S^{n+1} . This kind of derivation supports the interpretation that the Newton correction is useful only in a close neighborhood of S^{n+1}. Far away from S^{n+1} , such a linearizationmight still be trustedin some 'trust region' around the current iterate S^v. there are several models defining such a region. one of them is Levenberg-Marquardt [17] method have been worked out, e.g., by M.D. Hebden [10]. An affinecontravariantreformulationof the Levenberg-Marquardtmodel leads us to study the damped Newton iteration:

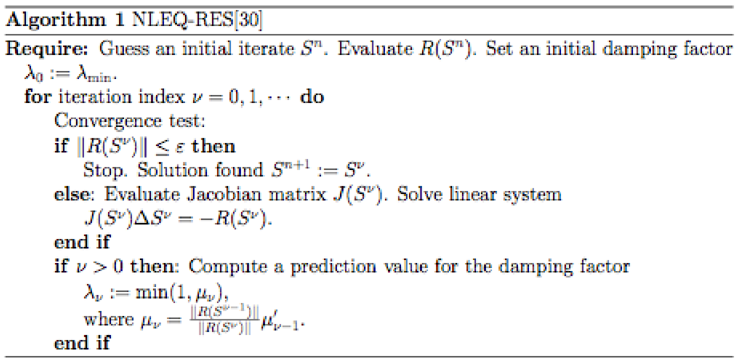
$$J(S^v)\Delta S^v = -R(S^v), \tag{2.4}$$

$$S^{v+1} = S^v + \lambda_v \Delta S^v, \lambda_v \in [0, 1] \tag{2.5}$$

Under the requirement of residual contraction

$$||R(S^{v+1})|| < ||R(S^v)||, \tag{2.6}$$

Which is certainly the most popular and the most widely used global convergence measure. There are theoretical analysis, which are characterized by means of affine contravariantLipschitz conditions, to derive the theoretically optimal iterative damping factors and prove global convergence within some range around these optimal factors.[7] However, in the theoretical analysis, there are parameters needed for the calculation of damping factors, e.g., Lipschitz constant, that are computationally unavailable. Thus, theoretically optimal damping factors generally cannot be obtained in numerical computation. to getan algorithmic estimation of the damping factors, residual based on trust-region strategies has been developed based on the theoretical analyses and computational estimates. Here, we focus onthe global Newton method with residual based convergene criterion and adaptive trust-region strategy,so-called NLEQ-RES algorithm. Note that the trust-region strategies in the NLEQ-REQ are defined based on the local descent of residual, ie., $||R(S^{v+1})|| < ||R(S^{v})||$. without considering the global structure and nonlinearity of the residual function in any specific nonlinear problem. In the NLEQ-RES, the trust regions for current iteration, $v + 1$, is derived based on the residual norm in the previous iteration, $||R(Sv||$, and the current iteration. Hence the definition of trust regions is history-based. Also, since $R(S^v)$ is unavailable a priori, an attempt of try-adapt-try is necessary to obtain the solution for the current iteration. Compared to the NLEQ-RES and other algorithmic trust-region strategies based on Levenberg-Marquardt model in, our trust region Newton method in next chapter is basen on the structure and nonlinearity of residual function (i.e., the flux function), specially, forimmiscible two-phase flow and transport in porous media, in which the trust regions are delineatedbefore solving the nonlinear problem. The algorithm NLEQ-RES is described as follows:



```
**Algorithm 1** NLEQ-RES[30]
**Require:** Guess an initial iterate $S^n$. Evaluate $R(S^n)$. Set an initial damping factor
  $\lambda_0 := \lambda_{\min}$.
  **for** iteration index $\nu = 0, 1, \cdots$ **do**
      Convergence test:
      **if** $\|R(S^\nu)\| \leq \varepsilon$ **then**
          Stop. Solution found $S^{n+1} := S^\nu$.
      **else**: Evaluate Jacobian matrix $J(S^\nu)$. Solve linear system
          $J(S^\nu)\Delta S^\nu = -R(S^\nu)$.
      **end if**
      **if** $\nu > 0$ **then**: Compute a prediction value for the damping factor
          $\lambda_\nu := \min(1, \mu_\nu)$,
          where $\mu_\nu = \frac{\|R(S^{\nu-1})\|}{\|R(S^\nu)\|}\mu'_{\nu-1}$.
      **end if**
```

Figure 2.6

2.5 Reduced-Newton method

Kwok and Tchelepi proposed a potential based ordering of the equations and unknowns that allows one tosolve for the saturations one cell at a time. The proposed ordering is valid for both two-phase and three-phase flow and for viscous, buoyancy, and capillary forces. For a two-phase system where the transport equations are discretized by a standard, implicit, upstream mobility-weighted scheme (standard FIM), the nonlinear system can be arranged in the following form

$$
\begin{aligned}
&f_{w1}(S_1, && p_1, \cdots, p_N) = 0 \\
&f_{w2}(S_1, S_2, && p_1, \cdots, p_N) = 0 \\
& && \vdots \\
&f_{wN}(S_1, S_2, \cdots, S_N, && p_1, \cdots, p_N) = 0 \\
&f_{o1}(S_1, && p_1, \cdots, p_N) = 0 \\
&f_{o2}(S_1, S_2, && p_1, \cdots, p_N) = 0 \\
& && \vdots \\
&f_{oN}(S_1, S_2, \cdots, S_N, && p_1, \cdots, p_N) = 0
\end{aligned}
$$

Figure 2.7

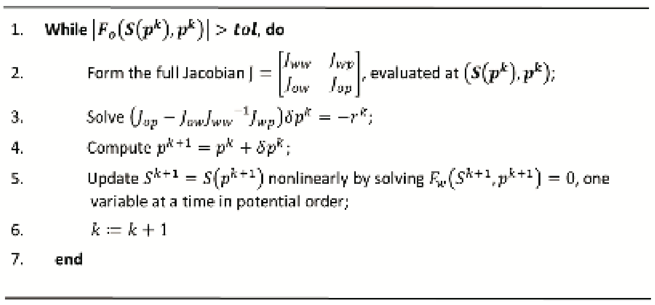
where $p_i \geq p_j$ whenever $i < j$. The monotonicity of the pressure field guarantees that the transport equations for cell j depend only on the saturations S_i with $i \geq j$. the triangular structure carries over the Jacobian, which now has the form

$$J = \begin{pmatrix} J_{ww} & J_{wp} \\ J_{ow} & J_{op} \end{pmatrix} \tag{2.7}$$

where J_{ww} is lower triangular. Based on the above potential-based ordering, a reduced-Newtion method is proposed in . Within each iteration, pressure is first updated by solving the following reduced Jacobian:

$$J_{reduced} = J_{op} - J_{ow} J_{ww}^{-1} J_{wp} \tag{2.8}$$

Since $J_{ww}$is lower triangular, the reduced Jacobian can be constructed effciently. Note that the pressure solution obtained here is identical to the one obtained from the fully-implicit method. Then, for the updated pressure field, the saturations are updated cell by cell. that is, we solve for the saturations of the cells with the highest potential (e.g., thecells perforated by injectors) first, and then proceed to solve for saturations at the downstream of these cells according to the phase potential. This process continues until the saturations at the cells with the lowest potential (e.g., the cells perforated by producers) have been updated. The reduced-Newton algorithm is summarized in 2.8



```
1. While $|F_o(S(p^k), p^k)| > tol$, do
2.     Form the full Jacobian $J = \begin{bmatrix} J_{ww} & J_{wp} \\ J_{ow} & J_{op} \end{bmatrix}$, evaluated at $(S(p^k), p^k)$;
3.     Solve $(J_{op} - J_{ow}J_{ww}{}^{-1}J_{wp})\delta p^k = -r^k$;
4.     Compute $p^{k+1} = p^k + \delta p^k$;
5.     Update $S^{k+1} = S(p^{k+1})$ nonlinearly by solving $F_w(S^{k+1}, p^{k+1}) = 0$, one
       variable at a time in potential order;
6.     $k := k + 1$
7.   end
```

Figure 2.8: Flow chart for thr reduced-Newton method

Numerical evidence in [16] shows that the potential-based reduced-Newton solver is able to converge for time steps that are much larger than what the standard Newton method can handle. In addition, when both methods can converge, the nonlinear solver in [16] converges faster than the standard Newton strategy. The phase-based potential ordering strategy in [16] provides us with the opportunity to resolve the nonlinearity for single-cell problems first, and then extend the methodology derived for single-cell problems to large scale simulation.

Chapter 3

Trust-Region Newton Solver

3.1 Descritized transport problem

We need to solve the following transport equation:

$$\frac{\partial s}{\partial t} + \frac{U_t}{\phi}\frac{\partial f}{\partial x} = 0 \tag{3.1}$$

Eq. 3.1 is a hyperbolic conservation law. In discretizing the flow term in this equation, we cannot use a centered first order difference, as this results in numerical errors. Rather, we must apply an upwinded difference. For accumulation term we use backward Euler approximation:

$$\frac{\partial s}{\partial t} = \frac{s_m^{n+1} - s_m^n}{\Delta t} + o(\Delta t), \tag{3.2}$$

where n and $n+1$ corresponds to the current and next timestep respectively, and m corresponds to the current gridblock. For flux term we use implicit forward Euler approximation:

$$\frac{\partial f}{\partial x} = \frac{f_m^{n+1} - f_{m-1}^{n+1}}{\Delta x} + o(\Delta x) \tag{3.3}$$

The flux function f is an explicit function of s which makes the Eq. 3.4 equal to given that u_t : is positive. Since, in our work we neglect gravity forces then change in the velocity direction and occuring of counter current flow is not consider.

$$\frac{\partial f}{\partial x} = \frac{f(s_m^{n+1}) - f(s_{m-1}^{n+1})}{\Delta x} + o(\Delta x) \tag{3.4}$$

which translates the original transport equation to the following nonlinear equation:

$$r_m(s_m^{n+1}, s_{m-1}^{n+1}) = \frac{s_m^{n+1} - s_m^n}{\Delta t} + \frac{U_t}{\phi}\frac{f(s_m^{n+1}) - f(s_{m-1}^{n+1})}{\Delta x} = 0 \tag{3.5}$$

with corresponding boundary and initial conditions $s_1^n = s_{inj}$ and $s_m^0 = s_{ini}$.

We apply Newton-Raphson method for the solution of this equation, using the following formula:

$$J\Delta s = -r. \tag{3.6}$$

Here s is a vector of saturations, $\Delta s = s^{n+1,k+1} - s^{n+1,k}$, k is the Newton iteration, J is Jacobial:

$$J = [J_{ij}] = \left[\frac{\partial r_i}{\partial s_j}\right]. \tag{3.7}$$

Our Jacobian matrix would be like

$$J(m,m) = \frac{1}{\Delta t} + \frac{Ut}{\Phi \Delta x}\left[\left(\frac{\partial f}{\partial s}\right)_m\right] \tag{3.8}$$

$$J(m,m-1) = \frac{-Ut}{\Phi \Delta x}\left(\frac{\partial f}{\partial s}\right)_{m-1} \tag{3.9}$$

In our iterations for every timestep, we are starting from the initial guess $s^{n+1,0} = s^n$. We find the solution:

$$\Delta s = -J^{-1} r \tag{3.10}$$

We update the solution:

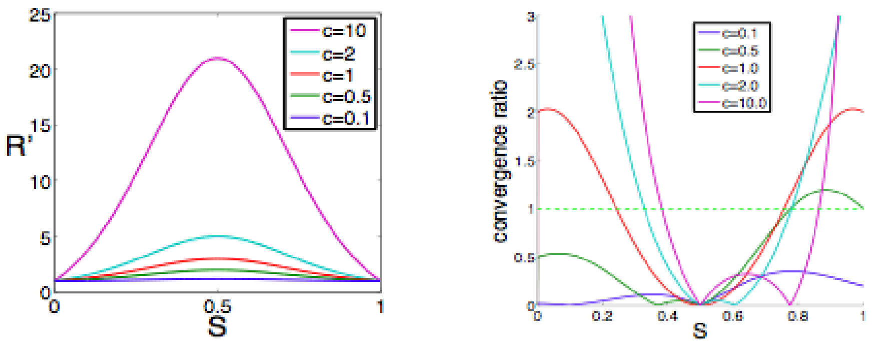
$$s^{n+1,k+1} = s^{n+1,k} + \Delta s \tag{3.11}$$

where k is the number of Newton's iteration. For every timestep, we repeat Newton's iterations until the convergence, which can be identified when $||r|| < \varepsilon_r$ and $||\Delta s|| < \varepsilon_s$.

3.2 The concept of Trust-Region in the context of nonlinear solvers

If the entire range of the unknown can be divided into several subregions, such that the convergence of the nonlinear solver is guaranteed once the iterative solutions are confined in the same subregion as the true solution, these sub regions are called 'Trust regions' [7], Here trust-region Newton methods refer to a specific type of Newton update in order to improve the convergence behavior. For two phase immiscible fluid by neglecting the gravitational and capillary forces, our flux function has the S- Shape. We divided the flux function into saturation trust regions. The delineation of these regions are dictated by the inflection of the flux term. If a crossing is detected, we 'chop back' the saturation value at the appropriate trust-region boundary.

Consider a single-cell problem in the presence of viscous forces only. The boundary condition is set as $S_L = 1$ and $S_R = 0$. The relative permeabilities and viscosities are : $K_{rw} = S^2$, $k_{rn} = (1 - S)^2$, and $\mu_w = \mu_n = 1$. The inflection point of the flux function is $S = 0.5$. The derivative of the residual with repsect to the scalar saturation, R', and the convergence ratio $\frac{|RR''|}{|R'|^2}$, respectively, are plotted versus saturation for different timestep sizes in Fig. Note that $c = \frac{\Delta t}{\Delta x}$, which is the total throughput(express in cell pore volumes), denotes the dimensionless timestep size. In fig 3.1, all the R' are the smooth large timestep sizes (e.g., c=10) correspond to strong nonlinearity. and from 3.1, it is clear that for different timestep sizes, the maxima of R' all occur at $S = 0.5$., the inflection point of the flux function.



(a) (b)

Figure 3.1: Derivative of the residual and the convergence ratio for single (From Wang [30])-cell incompressible two-phase in the absence of gravity and capillarity (a) $R' = \frac{dR}{dS}$ (b) $\frac{|RR''|}{|R'|^2}$
$c = \frac{\Delta t}{\Delta x}$ $M^0 = 1$. The inflection point of the flux function is $S^{inflec} = 0.5$

We can see that when the timstep is very small, e.g., $c = 0.1$, the convergence ratio is beloew unity for all saturation values. According to the Kantorovich theorem [19], convergence is guaranteed when the timestep is small enough. On the other hand, as the timestep size increases, the range of saturation values for which the convergence ratio is below unity gets smaller, and that implies potential convergence difficulties of Newton methods when the timestep size is large. Note that at $S = 0.5$ the convergence ratios are all zero, and there is a saturation region around $S = 0.5$, such that $\frac{|RR''|}{|R'|^2}$ is smaller than unity for all timesteps. Consequently, based on Kantorovich theorem, the nonlinear solution is guaranteed to converge once the iterative solution resides in the neighborhood of the inflection point ($S = 0.5$). That is, the inflection point is crucial to guarantee Newton convergence, which is consistent with the finding of Jenny et al [11].

3.3 Discontinues representation of Relative permeability

In real applications, we usually measure relative permeability and provide them as tables (standard input in any reservoir simulator). In the preliminary analysis, I use uniform mesh with specified number of entries (Nxx = 100). In this case, I evaluate my flux term in each newton iteration as an interpolation from my table and I introduce my derivative as an interpolation coefficient, the second derivative- a combination of two (become directional and require 3 points). One of the biggest advantage of discontinues representation is that we can find inflection point numerically.

Here we show by linear interpolation we find the value of our flux function and its derivate:

$$f(S) = f_j + \frac{f_{j+1} - f_j}{h}(S - S_j) \tag{3.12}$$

with corresponding gradients

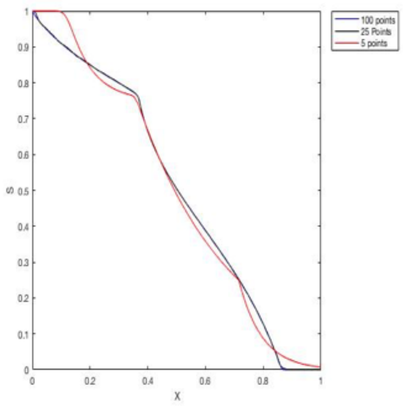
$$\frac{\partial F}{\partial S}(S) = \frac{F_{j+1} - F_j}{h} \tag{3.13}$$

The relation 3.13 provides direct derivatives with respect to nonlinear unknown, which significantly simplifies the evaluation and assembly of jacobians. In the proposed approach the number of points in the interpolation control the accuracy of the approximation nonlinear flux. Next we analyze the resolution of our table and see how the solution varies.

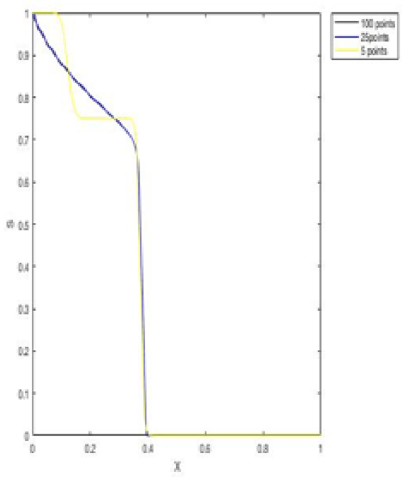
3.3.1 Coarsening

Here for the relative permeability curve with corey exponent 2 and fixing the mobility ratio (M=1) i experiment the impact of of the resolution in our result.

Table 3.1 demonstrates by decreasing the number of interpolation points, the number of iteration decrease although the error is getting higher. Moreover, the preconditioning performance also improves



(a) For the timestep equals to 0.2 (dt=0.2)



(b) For the timestep equals to 0.01 (dt=0.01)

Figure 3.2: coarsening

Table 3.1: Result Of Simulation for timestep=0.2

Number of Grids	Iteration Number	S inflection
500	56	0.5
100	31	0.5
50	23	0.5
25	16	0.4792
5	6	0.625

slightly. From the graph, we can see that by dropping the number of points until 25 the saturation curve is not changing a lot until we drop the number of points to 5. In conclusion, by coarsening we expect more linear effect however, in the case of binary system) the effect is not that tangible.

3.4 Appleyard chop

Here we try to modify my NR to be able to reach the higher time steps. We just added the constraint on local update dS not to be bigger than 0.2, We get very interesting result. By doing so We ran the code until the particular time t = 0.3 and save the result. Then, We restart increase the timestep for one step to imitate the flow in fracture reservoir. As we can see by increasing the timestep size, the number of iteration increase linearly for the first time step. In this case the saturation updates are corrected locally according to the end points of a Relative permeability function.

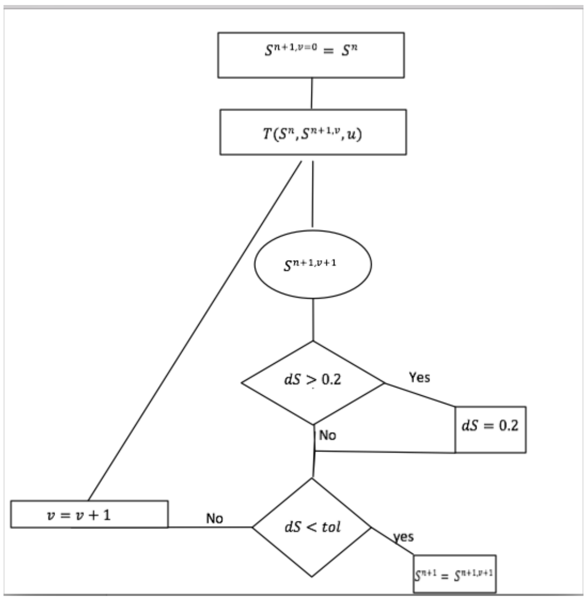


Figure 3.3: Flow diagram of Appleyard chop scheme

Finally, we get the global convergence for our implicit transport for the corey exponent 2 with the mobility ratio equals to 0.5. The number of Iteration increase by the following table by increasing the timestep.

Here we analyze how the number of iteration grows linearly by increasing the timestep. To experiment it we ran the simulation until a particular time (T=0.3) and fromthere we (Re)start and see how the number of iteration increase by enlarging the timestep.

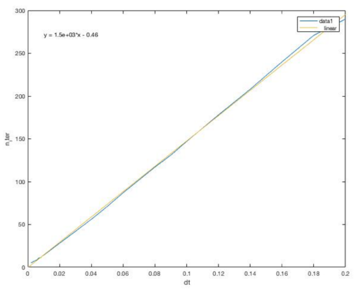


Figure 3.4: Increase in the number of iteration by increasing the timestep with analytical relative permeability curve corey exponent 2

3.5 Inflection point correction

In this case, We developed the solver based on what is already introduced regarding the Trust-region Newton solver by modifed newton update [11]. If the newton update cross the inflection point, we land it back to the ($\pm\epsilon$) vicinity of the inflection point. In other words, the update is directional depending whether we cross from left to right or from right to left the inflection point.

In this case, we need to know the inflection point. Inflection point in general is the function of both pressure and saturation. The advantage of introducing Relative permeabilty as a table is that we can easily find the inflection point from our table by linear representation of our second derivative.

3.6 Comparison between Appleyard chop and Inflection point

In the table 3.2 we first ran simulation until particular time (say 0.3) with timestep 0.001 and save this solution. Next we ran for one timestep only starting from 0.001 and finishing with 0.2. We count Newton iterations which required for this one timestep by elarging it and see how number of iteration evolves for our two different solvers. As we can see the number of iteration increase linearly with enlarging the timestep for both appleyard chop and Inflection point correction.

For the lower number of exponent when mobility ratio is 1 Appleyard chop works almost as well as inflection point 3.2. However, by increasing the exponent (no =nw= 8) Appleyard chop does not work anymore and it diverges. Basically, the reason Appleyard chop works

Table 3.2: Corey exponent 2 and mobility ratio equals to one.

dt(NXX= 500)	#Iteration for AYC solver	#iteratation for IC solver
0.002	5	5
0.003	6	6
0.004	7	7
0.005	8	8
0.009	11	11
0.01	12	12
0.03	21	21
0.05	28	28
0.1	39	38
0.12	41	42
0.14	44	45
0.16	47	48
0.18	49	49
0.2	49	49

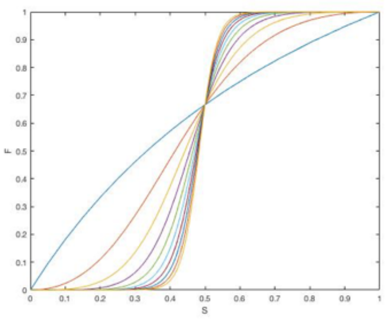
perfectly in the previous case is that we artificially do not let our update to be outside of the trust region. However, by increasing the parameter in a way the inflection point moves and by updating we are overshooting it even outside then the Appleyard chop does not work anymore. To illustrate it better we tried Appleyard chop for different Corey exponents and different mobility ratio and see how Appleyard chop going to crush afterward.

3.6.1 Effect of the Mobility ratio and the corey correlation on the flux function

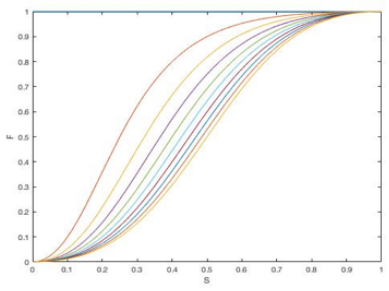
Here we try to understand how does our flux function varies by changing the mobility ratio and the exponents.

dt (NXX=500)	n= 2	n=3	n=10
0.002	5	6	7
0.003	6	7	8
0.004	6	8	8
0.005	7	9	10
0.006	8	9	11
0.007	9	10	12
0.008	10	11	13
0.009	10	12	15
0.01	11	13	16
0.02	16	24	27
0.03	21	34	35
0.04	24	44	46
0.05	28	54	56
0.06	31	61	65
0.07	34	67	75
0.08	35	73	85
0.09	37	80	95
0.1	40	86	106
0.12	45	99	125
0.14	47	113	145
0.16	51	125	165
0.18	54	137	186
0.2	56	149	204

Table 3.3: Increase of number of iteration with increase of time step size experimenting with different corey exponent Nxx is the number of entries in uniformly mesh relative permeability table

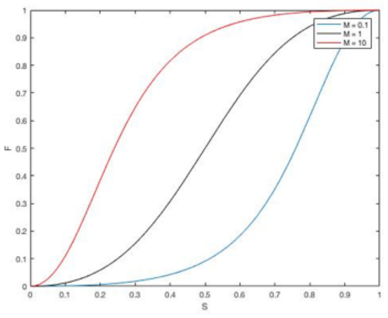


(a) our flux for different exponent from 1 to 10 as we can see by increasing the exponent the inflection point moves toward the right.

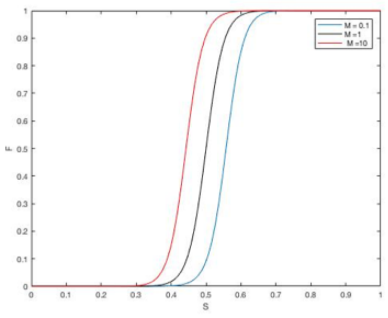


(b) flux function of S for different Mobility ratio. The lower the mobility ratio, the higher is the inflection point number.

Figure 3.5: How inflection point moves for different physics



(a) (a) flux function for different mobility when no=nw=2



(b) (b) Flux function for different mobility when no=nw=10

Figure 3.6: How inflection point moves for different physics

From above figures we understand that the flux functions vary significantly with the mobility ratio when the exponent is low and quite mildly when exponent is higher. Then comparing the flux function for $M = 10$ with $M = 0.1$, it is observe that when $M = 10$, and the slope of the flux function is big for $S \in [0, Sinflec]$. If $S = 0$is the initial condition and $S = 1$ is the injection condition, a shock forms starting from $S = 0$ followed by a spreading wave to $S = 1$.

We expect more nonlinear iterations needed when mobility ratio is higher for the same exponent since the shock speed is relatively high, resulting in early breakthrough, and the spread wave moves slowly. Correspondingly, the solution front will propagate for a longer distance until it reaches the solution front. Therefore, for the same timestep size, we expect more nonlinear iteration needed by the case M is higher(unfavorable). Finally, our Appleyard chop does not work when the mobility ratio is high enough or the exponent increase.

3.7 Comparison between Analytical solution and discontinuos representation of Relative permeability

Here we compare the performance of the Trust-region when considering the analytical solution and the numerical solution for different relative permeability curves by running the simulation until particulat time (t= 0.3). Next restarting from there for one timestep comparing how number of iteration evolves with enlarging the timestep.

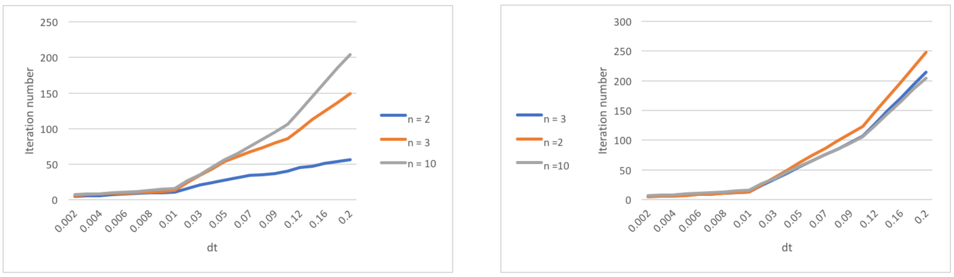


Figure 3.7: How the number of iteration increase with enlarging the timestep for different Corey exponent (a) Discontinuous relative permeability (b) Analytical solution for relative permeability

From graphs 3.7, the number of iteration decrease generally comparing to the analytical solution. Ironically, by increasing the exponent the iteration number increase for the same timestep which is not something that we expect since by increasing the exponent dispersion decrease. In conclusion, in both cases we get the global convergence for all the time steps and the number of iteration grows linearly with increasing the time step but in the table-base case, the performance is slightly better than the analytical solution especially for the lower exponent of relative permeability.

Chapter 4

Design of the better nonlinear solver and Preconditioning strategy for transport

Here we try to tackle the convergence difficulties in the nonlinear solver that are associated with the propagation of saturation front at residual saturation. We assume a simple 1d two-phase transport problem with only viscous forces. The initial condition in the domain is S = 0 (the non-wetting phase (oil) fully saturates the medium). The wetting phase (S=1) is injected from the left boundary and we produce at constant total rate from the right boundary. As is discussed later in this chapter, only the (strictly) single-phase initial and boundary conditions (i.e., propagation of saturation fronts into regions at the residual saturation) will cause the above described convergence difficulties, whereas other initial and boundary conditions will not have such slow convergence. Note that the Newton updates yield saturation distributions that are non-monotonic and then they ultimately converge to a discrete approximation that is monotone and accurate within a specified tolerance. The non-monotonic 'spikes' in the saturation distribution start off being quite large, it is noticed that the 'spikes' have very high saturation values, which are beyond the physical range and hence correspond to non-physical mass accumulation. The 'spikes' are propagating downstream as the Newton iterations proceed. The propagation at the leading edge, where the injected fluid is invading a cell fully saturated with the resident fluid (S=0), is constrained by the small (zero or near-zero) mobility of the invading phase in that cell. In fact, if the derivative of the relative permeability at S=0 is zero, then the propagation proceeds one grid block per iteration. Ultimately, the saturation distribution becomes monotonic, and from that point on, the newton updates converge quickly. Thus, the non-monotonicity of the saturation solution is the reason for the slow convergence. We can see Fig. 5.1 at S=0, the slope of the flux function is zero. Recall that the slope of the flux function is the speed of the saturation wave. With S=0 everywhere as the initial guess for the Newton solver, it is not possible to invade two successive cells in a single Newton update. This is because for each grid block, the influx comes from its upwind neighbor. If this immediate upwind cell has not been invaded by the injected wetting phase, the influx of the current grid block is zero and mass is transported one block per iteration. If the injected wetting phase for the current timestep cannot be transported beyond a single cell, the mass associated with the timestep is placed in that cell. This explains the ‘spikes in the saturation solution during Newton iterations. For a given balance of forces (i.e, viscous, buoyancy, and capillary), the shape of the flux function is determined primarily by the relative permeability curves. Next, we analyze the wave speeds for linear relative permeability curves and then we consider more general nonlinear function.

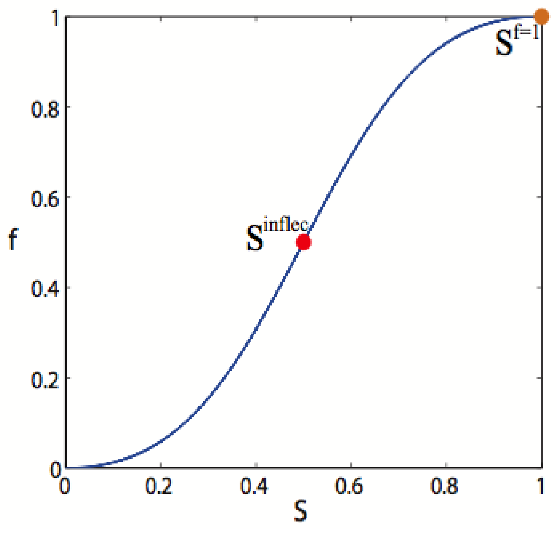


Figure 4.1: S-shape flux term for two-phase flow for viscous forces

Wang [30] demonstrated preconditioning strategy by analyzing the relative permeability curve as following section assuming initial saturation zero. Here we extend this approach to fracture reservoir in 1D and next chapter we apply this approach in OBL for binary system.

4.1 Linear relative permeability curve

Consider a one-dimensional horizontal model and assume that linear relative permeability curves are used, i.e.,

$$k_{rw} = S, \tag{4.1}$$

and

$$K_{ro} = 1 - S \tag{4.2}$$

We denote the viscosity ratio as

$$M = \frac{\mu_o}{\mu_w} \tag{4.3}$$

If only viscous forces are present (e.f., water flooding in a horizontal domain), the wetting-phase flux can be written as

$$f = \frac{\frac{k_{rw}}{\mu_w}}{\frac{k_{rw}}{\mu_w} + \frac{k_{ro}}{\mu_o}} = \frac{MS}{1 + (M-1)S} \tag{4.4}$$

for this simple 1D setting, the residual equation for cell i in discretized form can be written as

$$R = S_i^{n+1} - S_i^n + c(f_{(i+\frac{1}{2})}^{n+1} - f_{i-\frac{1}{2}}^{n+1}) \tag{4.5}$$

where c is a dimensionless timestep

$$c = \frac{u_t \Delta t}{\Delta x} \tag{4.6}$$

Since only viscous forces are present,the flux at the tight and left interfaces are

$$f_{i+1/2}^{n+1} = \frac{MS_i^{n+1}}{1 + (M-1)S_i^{n+1}} \tag{4.7}$$

and

$$f_{i-1/2}^{n+1} = \frac{MS_{i-1}^{n+1}}{1 + (M-1)S_{i-1}^{n+1}} \tag{4.8}$$

Hence, the residual can be expressed as

$$\begin{aligned} R = S_i^{n+1} - S_i^n + c(f_{i+1/2}^{n+1} - f_{i-1/2}^{n+1}) = \\ = (S_i^{n+1} - S_i^n) + cM \frac{S_i^{n+1} - S_{i-1}^{n+1}}{(1 + (M-1)S_i^{n+1})(1 + (M-1)S_{i-1}^{n+1})} \end{aligned} \tag{4.9}$$

We obtain

$$\frac{\partial R}{\partial S_i^{n+1}} = 1 + cM, \tag{4.10}$$

and

$$\frac{\partial R}{\partial S_{i-1}^{n+1}} = \frac{-cM}{(1 + (M-1)S_i^{n+1})(1 + (M-1)S_{i-1}^{n+1})} \tag{4.11}$$

If the initial condition is $S^n = 0.0$ and we take it as intial guess for solution at $n+1$, then

$$\frac{\partial R}{\partial S_i^{n+1}} = 1 + cM \tag{4.12}$$

and

$$\frac{\partial R}{\partial S_{i-1}^{n+1}} = -cM \tag{4.13}$$

For Newton method,the Jacobian for the first iteration is

$$\begin{pmatrix} 1+cM & & & & \\ -cM & 1+cM & & & \\ & -cM & 1+cM & & \\ & & . & . & \\ & & & . & . \end{pmatrix} \tag{4.14}$$

Hence the timestep size c is reflected in th wave speed. See [24]. Assuming that the left-boundary condition is $S = S_0$, the residual (righthand side) for the first iteration is:

$$\begin{pmatrix} -cM\frac{S_0}{1+(M-1)S_0} \\ 0 \\ . \\ . \\ 0 \end{pmatrix} \tag{4.15}$$

Other boundary condition result in different value for the residual term of the left most cell, whereas the residual terms of other cells will remain zero at the first iteration, regardless of the form of the boundary condition. Thus, starting from the initial guess, $S = 0$, the non-zero term in the residual is propagated downstream and the saturation is dispersive throughout the domain, since there is a non-zero element in every row of the lower off-diagonal part of the Jacobian.

Specially, when $M = 1$, the Jacobian has the following form

$$\begin{pmatrix} 1+c & & & & \\ -c & 1+c & & & \\ & -c & 1+c & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \end{pmatrix} \tag{4.16}$$

In fact, for $M = 1$ the residual equation, R, becomes a linear function of the solution,

$$R = S_i^{n+1} - S_i^n + c(S_i^{n+1} - S_{i-1}^{n+1}) \tag{4.17}$$

and this case, the Newton method will converge in one iteration regardless of the timestep size.

4.2 Nonlinear relative permeability curves

Assume the relative permeability curves are given by

$$K_{rw} = S^\alpha \tag{4.18}$$

$$K_{ro} = (1-S)^\beta \tag{4.19}$$

Where $\alpha >= 1$ and $\beta >= 1$. Then the flux function can be writen as:

$$f = \frac{(\frac{S^\alpha}{\mu_w})}{\frac{S^\alpha}{\mu_w} + \frac{(1-S)^\beta}{\mu_o}} \tag{4.20}$$

Hence the derivative is:

$$\begin{aligned} \frac{df}{dS} &= \frac{\alpha\frac{S^\alpha}{\mu_w}(\frac{S^\alpha}{\mu_w}) - \frac{S^\alpha}{\mu_w}(\alpha\frac{S^{\alpha-1}}{mu_w} - \beta\frac{(1-S)^{\beta-1}}{\mu_0})}{(\frac{S^\alpha}{\mu_w} + \frac{(1-S)^\beta}{\mu_0})^2} = \\ &= \frac{\alpha S^{\alpha-1}(1-S)^\beta + \beta S^\alpha(1-S)^{\beta-1}}{M(S^\alpha + \frac{(1-S)^\beta}{M})} \end{aligned} \tag{4.21}$$

for $\alpha > 1$, $\frac{df}{ds} = 0$ for $S = 0.0$.

The residual is expressed as

$$R = S_i^{n+1} - S_i^n + c(f_{(i+\frac{1}{2})} - f_{i-\frac{1}{2}}^{n+1}) \tag{4.22}$$

Hence

$$\frac{\partial R}{\partial S_i^{n+1}} = 1 + c\frac{df_{i+\frac{1}{2}}^{n+1}}{dS_i^{n+1}} \tag{4.23}$$

and

$$\frac{\partial R}{\partial S_{i-1}^{n+1}} = -c\frac{df_{i-\frac{1}{2}}^{n+1}}{dS_{i-1}^{n+1}} \tag{4.24}$$

Assume $\alpha > 1$ It follows that when $S^n = 0$ everywhere and is taken as the initial guess for the next timestep$n+1$, then $\frac{\partial R}{\partial S_{i-1}^{n+1}} = 0$ and $\frac{\partial R}{\partial S_i^{n+1}} = 1$ So, the jacobian matrix for the first Newton iteration is the identity matrix.

$$\begin{bmatrix} 1 & & & \\ 01 & & & \\ & 01 & & \\ & & .. & \\ & & .. & \end{bmatrix} \tag{4.25}$$

The timestep size c does not appear in the first Jacobian Matrix. Assume that $f = 1$ at the left boundary, corresponding residual for the first iteration is

$$\begin{bmatrix} c \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \tag{4.26}$$

Since the Jacobean matrix is the identity, we can see that non-zero terminate residual cannot propagate more than one block after the first iteration. On the other hand, even though $S^n = 0$ is the initial condition, if we take the initial guess of the solution for the current timestep, $S^n + 1, 0$, as $S^*$is not zero, we then have

$$\frac{\partial R}{\partial S_i^{n+1}} = 1 + cf'^* \tag{4.27}$$

and

$$\frac{\partial R}{\partial S_{i-1}^{n+1}} = -cf^{'*} \tag{4.28}$$

Hence the jacobian matrix for the first iteration is:

$$\begin{bmatrix} 1+cf'* & & & & \\ -cf^{'*} & 1+cf'* & & & \\ & -cf^{'*} & 1+cf'* & & \\ & & -cf^{'*} & ... & \\ & & & ... & \\ & & & & \end{bmatrix} \tag{4.29}$$

Where $f^{'*}$ is df/dS evaluated at S^*. the timestep size c appears in the jacobian matrix if $f^{'*}$ is not zero. Assuming that the left boundary condition is$f = 1$, the corresponding residual vector for the first iteration is

$$\begin{bmatrix} S^* + c(f^* - 1) \\ S^* \\ . \\ . \\ S^* \end{bmatrix} \tag{4.30}$$

The solution update, δS, is obtained by solving the linear system $JS = -R$, and then$S^* + S$ serves as the starting point for the second Newton iteration.

Now we prove that for the first Newton iteration if $f^{'*} > 0$, the resulting $S * +S$of our 1D problem is decreases monotonically as i increase from 1 to N, where N is the number of blocks. Proof. Since the Jacobian matrix 4.29 is lower triangular, we can solve the elements in$S = [S1, S2, \dots, SN]^T$ one by one.

$$\begin{bmatrix} 1+cf'* & & & & \\ -cf'^* & 1+cf'* & & & \\ & -cf'^* & 1+cf'* & & \\ & & -cf'^* & \dots & \\ & & & \dots & \\ & & & & \end{bmatrix} \begin{bmatrix} \delta S_1 \\ \delta S_2 \\ . \\ . \\ . \\ \delta S_N \end{bmatrix} = - \begin{bmatrix} S^* + c(f^*-1) \\ S^* \\ . \\ . \\ . \\ S^* \end{bmatrix} \tag{4.31}$$

First, we obtain the solution update for the first block. Namely,

$$\delta S_1 = -\frac{S^* + c(F^* - 1)}{1 + cf'^*} \tag{4.32}$$

Hence:

$$S^* + \delta S_1 = S^* - \frac{S^* + c(f^* - 1)}{1 + cf'^*} = \frac{c(S^* f'^* - f^* + 1)}{1 + cf'^*} > 0 \tag{4.33}$$

For $i \geq 2$

$$-cf'^*\delta S_{i-1} + (1 + cf'^*\delta S_i) = -S^* \tag{4.34}$$

Therefore,

$$\delta S_i + S^* = \frac{cf'^*}{1 + cf'^*}(\delta S_{i-1} + S^*) \tag{4.35}$$

and since $f'* > 0$

$$0 < \frac{cf'^*}{1 + cf'^*} < 1 \tag{4.36}$$

It follows that

$$\delta S_i + S^* < \delta S_{i-1} + S^* \tag{4.37}$$

Then, we have proved that $S^* + \delta S$ decreases monotonically as i increase. The solution can be written as

$$\delta S_i + S^* = (\frac{cf'^*}{1 + cf'^*})^{i-1}(\delta S_1 + S^*) \tag{4.38}$$

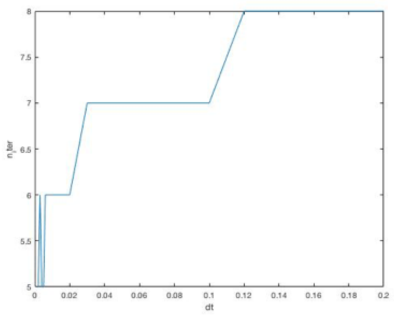
and since $S^* + \delta S_1 > 0$, then

$$\delta S_i + S^* > 0 \qquad \forall i \geq 2, \tag{4.39}$$

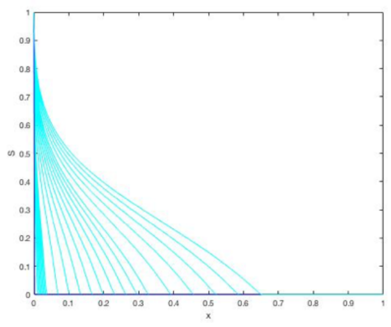
That is, the saturation solution from the first iteration is positive (and less than unity) in every block of the $1D$ domain. Form Eqn 4.35 above, we can see that for any initial guess,S^*, if$f'^* > 0$, the first Newton iteration yields a saturation distribution that is monotonic and positive. And to allow for maximum propagation of the saturation waves downstream, we make$S_i + S^*$ as close to $S_{i-1} + S^*$ as possible. Thus, we maximize cf'^*. Specifically, we use the maximumf'^*, i.e., we use the inflection point. Next, we propose a preconditioning strategy for nonlinear transport solvers based on this analysis.

4.3 Preconditioning Strategy

What we did we put inflection point as the initial guess for the first newton iteration. The number of iteration decrease significantly as we expected. However, the number of iteration does not anymore monotonically increasing with respect to the time step increase. Here we analyze preconditioning in a case that the initial condition is monophase and is fully saturated with oil for the different magnitude of the first timestep for the exponent 2 of my relative permeability and the mobility ratio equals to one:



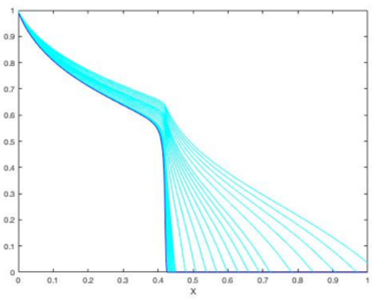
(a) . How number of iteration varies in the case of precondition assuming the initial water saturation is zero



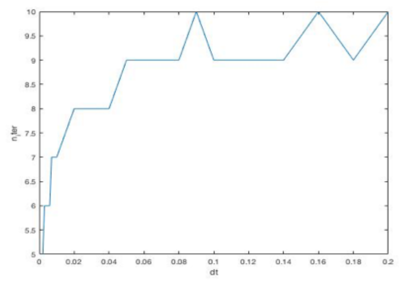
(b) only for one timestep assuming the initial water saturation is zero

Figure 4.2: How inflection point moves for different physics

Next for a DFM, we ran simulation up to the well resolved condition and then we (re)start from this distribution with different size of time step for one step only and we obtained the following result:



(a) Solution of trust-region Newton method for different one timestep after the particular time (t=0.3)



(b) Number of iteration for different one timestep after a particular t (t = 0.3)

Therefore, the preconditiong strategy can work also in the case when my initial water saturation is not zero as well. Note that when the buoyancy is dominant, the flux function becomes non-monotonic and physically indicate the occurrence of counter-current flow for part of the saturation range. In such cases, we use the smaller inflection point, as the initial guess for the grid blocks in the imbibition region (wetting phase displacing non-wetting phase), and we use the larger inflection point as the initial guess for the grid blocks in the drainage region (non-wetting phase displacing wetting phase).

4.4 Performance Comparison between the trust region nonlinear solver with and without preconditioning

We test the transport problems for four types of the relative permeability curves.

$$K_{rw} = S^2; K_{rn} = (1-S)^2 \tag{4.40}$$

$$K_{rw} = S^3; K_{rn} = (1-S)^3 \tag{4.41}$$

$$K_{rw} = S^{10}; K_{rn} = (1-S)^2 \tag{4.42}$$

$$K_{rw} = S^{10}; K_{rn} = (1-S)^{10} \tag{4.43}$$

4.4.1 Pure oil in place

Here for the Viscous dominate flows in 1D (500 gridblocks) where the initial condition is S= 0 and wetting phase (S=1) is injected from the left boundary. We experiment the result with three different initial guess. Firstly, initial condition as the initial guess. Secondly, intial guess equals to last update of S when running our code until t=0.3 with dt = 0.001 and lastly, initial guess equals to the inflection point.

M	Different Preconditioning		
	Without Precond.	Initial guess= Not initial cond	With Precond.
0.5	142	31	9
1	155	21	9
10	196	11	9

Table 4.1: When the no=nw=3. In the column without Precond the initial Guess is the same as initial condition

M	Different Preconditioning		
	Without Precond.	Initial guess= Not initial cond	With Precond.
0.5	188	12	10
1	209	11	10
10	235	12	10

Table 4.2: no=nw=10. In the first column without preconditioning the initial guess equals the initial condition; the second column the initial guess is the last update of S when running our code until t = 0.3 with dt=0.001, Last column initial guess = inflection point

M	Different Preconditioning		
	Without Precond.	Initial guess= Not initial cond	With Precond.
0.5	126	11	10
1	142	11	10
10	148	11	10

Table 4.3: no=2 nw=10,In the first column without preconditioning the initial guess equals the initial condition; the second column the initial guess is the last update of S when running our code until t = 0.3 with dt=0.001, Last column initial guess = inflection point

M	Different Preconditioning		
	Without Precond.	Initial guess= Not initial cond	With Precond.
0.5	62	11	8
1	59	22	8
10	22	32	8

Table 4.4: no=nw=2, In the first column without preconditioning the initial guess equals the initial condition; the second column the initial guess is the last update of S when running our code until t = 0.3 with dt=0.001, Last column initial guess = inflection point

It is noticed that:

(i) With preconditioning strategy, we accelerate the convergence for all the cases. As we expected, choosing the inflection point as an initial guess is the best preconditiong we can consider since we maximize the derivatice and thus maximize the propagation of the saturation wave downstream.

(ii) By increasing the Mobility ratio, the number of iteration increase for the same time step and the same exponent. It can be explained in the sense that by increasing the M (unfavorable displacement), the shock speed is relative high, resulting in early breakthrough, and the spread wave moves slowly. Correspondingly, the solution front will propagate for a longer distance and hence the spread wave also needs to propagate for a longer distance until it reaches the solution front. Therefore, for the same timestep size, we expect more nonlinear iteration needed by the case with M=10 than with M=0.5.

(iii) In general, by increasing the exponent relative permeability curves are less dispersed and the number of iteration needed also decrease. Moreover, the effect of mobility ratio also less tangible by increasing the exponent.

(iv) For the case with different exponent for wetting and nonwetting relative permeability curves we observe that the converged solution fronts is less dispersive (sharper) than those with quadratic, or cubic, relative permeability curves.

4.4.2 Preconditioning for the Fracture Reservoir

In fracture reservoir, the saturation front is moving slowly in the matrix and suddenly it reaches the fracture and it propagates. To effectively imitate this process instead of increasing the permeability we enlarge the timeste. Therefore, i first ran simulation until particular time (t = 0.3 d) with a small ltimestep 0.001 d and save this solution. Next i ran for onetime step only enlarging from 0.001 d and finishing with 0.2 d and count Newton iterations which required for this one timestep. Here we just report the performance for the timestep equals to $0.2d$ with and without preconditioning.

M	Different Preconditioning		
	Without Precond.	Initial guess= Not initial cond	With Precond.
0.5	60	13	9
1	56	14	9
10	11	4	10

Table 4.5: Performance comparison (number of iteration) between different nonlinear solvers, with and without preconditioning, for 1D transport under viscous forces. the exponents of the Rel.Perm is 2

M	Different Preconditioning		
	Without Precond.	Initial guess= Not initial cond	With Precond.
0.5	142	22	11
1	149	25	10
10	111	5	10

Table 4.6: Performance comparison (number of iteration) between different nonlinear solvers, with and without preconditioning, for 1D transport under viscous forces. the exponents of the Rel.Perm is 3

M	Different Preconditioning		
	Without Precond.	Initial guess= Not initial cond	With Precond.
0.5	198	27	10
1	204	27	12
10	229	30	11

Table 4.7: Performance comparison (number of iteration) between different nonlinear solvers, with and without preconditioning, for 1D transport under viscous forces. the exponents of the Rel.Perm is 10

M	Different Preconditioning		
	Without Precond.	Initial guess= Not initial cond	With Precond.
0.5	142	20	11
1	149	21	10
10	111	22	10

Table 4.8: Performance comparison (number of iteration) between different nonlinear solvers, with and without preconditioning, for 1D transport under viscous forces. No=2 and nw=10

Chapter 5

Developed trust region nonlinear solver in the simulation prototype based on Operator Based Linearization

5.1 What is Operator Based Linearization

It is a new approach introduced by Voskov [27], for the linearization of governing equations that describe flow and transport in porous media is proposed in this work. It is based on an approximate representation of the exact physics of the problem, which is similar to an approximate representation of space and time discretization performed in conventional simulation. The governing equations are introduced as a combination of operators, dependent on spatially altered properties and operators, fully controlled by nonlinear properties of fluid and rock. Next, a parametrization in the physics space of the problem is introduced. The property- based operators are approximated using direct interpolation in the space of nonlinear unknowns. The discrete version of the governing equations is constructed as a combination of operators that approximate both nonlinear physics and discretization in time and space. This approach is applied to the reservoir simulation of miscible and immiscible displacement processes. Later, due to the hyperbolic nature of some variables(e.g., overall compositions), the vast majority of parameter space remains unused. The adaptive approach avoids these disadvantages by removing the need for the entire pre-processing stage (Zaydullin et al., 2013). Khait and Voskov [15] demonstrated the adaptive operator based linearization approach. In this approach, base points are computed only when they are required by the current physical state of a control volume. The obtained operator values are then employed in the interpolation process and stored for future use. Consequently, the method adds a new base point and computes appropriate operators, if the base point was not computed before. In the end of the simulation, the resulting sparse multi-dimensional table of stored operators represents an actual subspace of physical parameters used in the process [15]. Once the linear system with the Jacobian and residual is constructed, it needs to be solved. Because the dimensionality of a typical reservoir-simulation problem is rather high, iterative linear solvers are usually used with effective preconditioning (e.g., two-stage preconditioning using constrained pressure residual) (Wallis et al. 1985). Once the linear system with the Jacobian and residual is constructed, it needs to be solved. Because the dimensionality of a typical reservoir-simulation problem is rather high, iterative linear solvers are usually used with effective preconditioning (e.g., two-stage preconditioning using constrained pressure residual) (Wallis et al. 1985). Once the solution to the linear system with predefined tolerance is found, we need to update the nonlinear unknowns and repeat the nonlinear iteration. The nonlinear solution may require several nonlinear iterations to converge, depending on the nonlinearity of a problem. The number of nonlinear iterations can be sufficiently reduced by using advanced nonlinear solvers as we demonstrated in the chapter three. The extension of the natural formulation can help to avoid variable substitution and apply correction to discontinuos changes in the derivative usually related to phase appearance and disappearance. (voskov2012)

5.2 Modelling approach

In this section, we describe the governing equations and nonlinear formulation for a reservoir simulation problem.

The transport equations for an isothermal system containing n_c components and n_p phases can be written as:

$$\frac{\partial}{\partial t}(\phi \sum_{j=1}^{n_p} x_{cj}\rho_j vj) + \text{div} \sum_{j=1}^{n_p} x_{cj}\rho_j q_j + \sum_{j=1}^{np} x_{cj}\rho_j \tilde{q}_j = 0, c = 1, ..., n_c \quad (5.1)$$

Here, we typically define the coefficients of the equations as functions of spatial coordinate ξ and physical state ω :

$\phi(\xi, \omega)$- porosity

$x_{cj}(w)$- the mole fraction of component c in phase j,

$s_j(\omega)$ - phase saturations,
$\rho_j(\omega)$- phase molar density,
$v_j(\xi,\omega)$- phase velocity,
$q_j(\xi,\omega)$- phase rate per unit volume.
To describe the flow of each phase, we use Darcy's law:

$$v_j = -(K\frac{k_{rj}}{\mu_j}(\nabla p_j - \gamma_j \nabla d)), j = 1, ..., n_p \tag{5.2}$$

where
$K(\xi)$ - Permeability tensor, $k_{rj}(\omega)$- Relative permeability,
$\mu_j(\omega)$ - Phase viscosity,
$p_j(\omega)$- vector of pressure in phase j,
$\gamma_j\omega$- gravity term,
$d(\xi)$ - vector of depths (positive downwards).
By applying a finite-volume discretization on a general unstructured mesh and backward Euler approximation in time, we can transform the conservation equations into

$$V((\phi\sum_j x^l_{cj}\rho_j s_j)^{n+1} - (\phi\sum_j x_{cj}\rho_j s_j)^n) - \Delta t\sum_{l\in L}(\sum_j x^l_{cj}\rho^l_j T^l_j \Delta\Psi^l)+ \\ +\Delta t\sum_j \rho_p x_{cj} q_j = 0 \tag{5.3}$$

where V is the volume of a control volume and $q_j = \tilde{j}_j V$ the source of a phase. Here we neglected capillarity, gravity and used a Two-Point Flux Approximation (TPFA) with upstream weighting introducing the summation over all interfaces L connecting the control volume with another grid blocks. Based on these simplifications, $\Delta\psi^l$ becomes a simple difference in pressures between blocks a and b, where T^l_j is a phase transmissibility. These assumptions are not required by the method, but help to simplify the further description.
Nonlinear unknowns
The system of equations 5.3 is the discretized form of flow and transport equations for general multi-component fluid. Here, we used a Fully Implicit Method (FIM) time approximation. It requires the $(x_{cj}l\rho_j lT_j l\delta\psi^l)$ flux term to be defined based on nonlinear unknowns at a new timestep (n+1) and introduces nonlinearity to the system of governing equations. Another source of nonlinearities comes from the additional assumption on instantaneous thermodynamic equilibrium, which is required to close the system.

Several different strategies exists for the nonlinear solution of the resulting system, see [1], and [26] for an extensive description and examples. Here, we used the overall molar formulation suggested by

Collins et al. [6]. In this formulation, thermodynamic equilibrium is assumed at every nonlinear iteration of solution of 5.3 . For the control volume at multiphase conditions with n_p -phases, the following system of equations needs to be solved:

$$F_c = z_c - \Sigma v_j x_{cj} = 0 \tag{5.4}$$

$$F_{c+n_c} = f_{c1}(p, T, x1) - f_{cj}(p, T, x_j) = 0, \tag{5.5}$$

$$F_{j+n_c*n_p} = \Sigma(x_{c1} - x_{cj}) = 0 \tag{5.6}$$

$$F_{n_p+n_c*n_p} = \Sigma v_j - 1 = 0 \tag{5.7}$$

Here $z_c = \Sigma x_{cj}\rho_j s_j/\rho_j s_j$ is overall composition and $f_{cj}(p, T, x_{cj})$ is the fugacity of component c in phase j. This procedure is called a multiphase flash [18]

For a given overall composition z_c , the solution of 5.4 - 5.7 provides molar fractions for each component x_{cj} and phase fractions V_j.

In the overall molar formulation, the unknowns are p andz_c. They fully define the physical state ω for a given control volume:
ω : $[P, z1, ..., z_{n_c-1}]$. Once, multiphase flash is solved, it can provide derivatives of all properties in 5.3 with respect to nonlinear unknowns using the inverse theorem, see [26] for more details.

OBL Approach

We can rewrite Equation 5.3 as the component of a residual vector in an algebraic form, In this case each term is represented as a product of state-dependent and space-dependent operators. The resulting mass conservation equation, written for a control volume i, is

$$r_c(\xi, \omega, u) = a(\xi)(\alpha_c(\omega) - \alpha_c(\omega^n)) - \sum_l \beta_c^l(\omega) b^l(\xi, \omega) + \theta_c(\xi, \omega, u) = 0, \tag{5.8}$$

Here, we defined

$$\alpha_c(\omega) = (1 + c_r(p - p_{ref})) \sum_{j=1}^{n_p} x_{cj}\rho_j s_j, \tag{5.9}$$

$$a(\xi) = V(\xi)\phi_0(\xi) \tag{5.10}$$

$$\beta_c(\omega) = \sum_j x_{cj}\lambda_{p,j}\rho_j \tag{5.11}$$

$$b(\xi, \omega) = \Delta t T^{ab}(\xi)(p^b - p^a) \tag{5.12}$$

$$\theta_c(\xi, \omega, u) = \Delta t \sum_j \rho_j x_{cj} q_j(\xi, \omega, u) \tag{5.13}$$

In addition, c_r is the rock compressibility, T^{ab} is the geometric part of transmissibility (which involves permeability and the geometry of the control volume), ω and ω^n are nonlinear unknowns at the current and the previous timestep respectively, and u is a vector of well control variables. $\theta_c(\xi, \omega, u)$ is the influx/outflux term. in addition, ϕ_0, V_i, and p, are the initial porosity, volume, and pressure.

The operator α_c is dependent on the properties of rock and fluid, and independent of spatially distributed properties (initial porosity) as in the case of the operator a. Similarly, the divergence operator is present as a fluid-related operator β_c independent of spatial distributed properties (permeability) and the discretization-related operator b. The same approach can be applied for the well source/sink operator θ_c, but for simplicity we did not apply it here.

5.3 Linearization method

In this section, we describe different types of linearization using the general algebraic form of governing equation [33]

Standard linearization approach

To solve nonlinear system 5.8, we need to linearize it. The conventional approach in reservoir simulation is based on the application of the Newton-Raphson method. In each iteration of this method, we need to solve a linear system of equations of the following form

$$J(\omega^k)(\omega^{k+1} - \omega^k) = -r(\omega^k) \tag{5.14}$$

Where J is the Jacobian defined at nonlinear iteration step k. The typical approach requires a sequential assembly of the residual and the Jacobian based on numerical approximation of the analytic relations in 5.9- 5.13 This may demand an interpolation in tables (for standard PVT correlations or relative permeabilities), or a solution of the highly nonlinear equations (for EoS-based properties). Each property evaluation requires a storage space for both values of the property and its derivatives with respect to the nonlinear unknowns. Most reservoir simulation software performs numerical [21] or hand diffferentiation [21] of each property with respect to nonlinear unknowns. In this work, we utilized the ADGPRS framework [26] for the implementation of conventional and newly proposed linearization procedures.

Operator-based linearization

It is a new strategy for the linearization of the reservoir simulation problem described by Eq. 5.8. As can be seen from the structure of each operator in 5.9- 5.13, this system is based on a complex combination of different nonlinear properties and relations. Since we fixed our space and time approximation, the discretization error can be controlled only by the variation of the timestep size ΔT and the characteristic size of the mesh embedded in the T^{ab} term. Both of these errors are controlled by the operators ψ and θ_c.

The operatos α_c and $\beta_c$represent the physics-based terms. The accuracy of the nonlinear physics representation is controlled by these two operators (and a part of θ_c). In conventional simulation, we introduce all the nonlinear properties into the conservation equation as is. Next, the nonlinear solver tries to resolve all the detals of the nonlinear description, struggling sometimes with unimportant features due to the numerical nature of the property representations.

The Operator-Based Linearization (OBL) strategy, proposed in this work is based on the simplified representation of the nonlinear operators α_c and β_c in the parameter-space of the simulation problem. We uniformly discretize the parameter space with a fixed number of points N. The interpolation intervals are defined as $[P_1, P_2, ..., P_N]$ and $[Z_1, Z_2, ..., Z_N]$. Next for $p \in [P_i, P_{i+1}]$ and $z \in [Z_j, Z_{j+1}]$we define

$$p_i = \frac{p - P_i}{P_{i+1} - P_i}, z_j = \frac{z - Z_j}{Z_{j+1} - Z_j} \tag{5.15}$$

and the auxiliary relation $f_{i,j} = f(P_i, Z_j)$- Based on p_i, z_j and $f_{i,j}$, the interpolant of function F_f can be defined as

$$F_f(p, z) = (1 - z_j)[(1 - p_i)f_{i,j} + p_i(f_{i+1,j})] + z_j[(1 - p_i)f_{i,j+1} + p_i f_{i+1,j+1}] \tag{5.16}$$

with correspond gradients

$$\frac{\partial F_f}{\partial p} = \frac{(1 - z_j)(f_{i+1,j} - f_{i,j}) + z_j(f_{i+1,j+1} - f_{i,j+1})}{p_{i+1} - p_i} \tag{5.17}$$

with correspond gradients

$$\frac{\partial F_f}{\partial p} = \frac{(1 - z_j)(f_{i+1,j} - f_{i,j}) + z_j(f_{i+1,j+1} - f_{i,j+1})}{p_{i+1} - p_i} \tag{5.18}$$

The error between an interpolant and function can be evaluated based on the following relation

$$|f - F_f| \leqslant \frac{V_w^2}{4} sup \left| \frac{\partial^2 f}{\partial \omega^2} \right| \tag{5.19}$$

To evaluate $\hat{\alpha_c}(p,z)$ and $\hat{\beta}_c(p,z)$in the course of the simulation, we apply an interpolation

$$\hat{\alpha_c}(p,z) = F_{\alpha_c}(p,z), \hat{\beta}_c(p,z) = F_{\beta_c}(p,z) \tag{5.20}$$

This representation helps to provide a continuous description of the physics based operators in the proposed approach. The number of points in the interpolation controls the accuracy of the approximation of the nonlinear physics, similar to the accuracy of the approximation in space and time being controlled by the grid size. The error described by Eq. 5.19 is similar to the truncation error in the discretization of equation 5.8. The nonlinear solver deals here with a simplified representation directly expressed as a piece-wise linear combination of nonlinear unknowns. Also; the relation (21) - (22) provides direct derivatives with respect to nonlinear unknowns, which significantly simplifies the evaluation and assembly of Jacobians.

Solution method

Once each operator in Eq. 5.8 is linearized, the residual vector r and the Jacobian J can be assembled. The overall- molar formulation does not require a secondary set of equations [26] and we can apply the linear solver directly to system. in this work we emplyed GMRES with the two-stage Constrained Pressure Residual (CPR) preconditioner [22] as a linear solver

Consistency of numerical solution

In this section, we demonstrate the consisteny of the prposed linearization method assuming that the original problem described by [1] has a numerical solution. To simplify analysis,we assumethat the model is limited by a 1D reservoir with Cauchy boundary conditions on left and right side. This simplifies the spacial discretization, which yields to the following equationin vector form (The length of vector corresponds to the number of components n_c) for the block i:

$$\begin{aligned} r_i(\omega_{i-1}, \omega_i, \omega_{i+1}, \omega_i^n) = (\alpha(\omega_i) - \alpha(\omega_i^n)) - \beta(\omega_i) b_{i^+}(\omega_i, \omega_{i+1}) + \\ + \beta(\omega_{i-1}) b_{i^-}(\omega_i, \omega_{i-1}) = 0, \end{aligned} \tag{5.21}$$

where

$$a_i = \phi_{0i} V_i, \tag{5.22}$$

$$b_{i+}(\omega_i, \omega_{i+1}) = \Delta t T_{i,i+1}(P_{i+1} - P_i) \tag{5.23}$$

$$b_{i-}(\omega_i, \omega_{i-1}) = \Delta t T_{i-1,i}(P_i - P_{i-1}) \tag{5.24}$$

In the case of total velocity formulation

$$b_{i+}(\omega_i, \omega_{i+1}) = \Delta t T_{i,i+1}(P_{i+1} - P_i)\Lambda(\omega_i) \tag{5.25}$$

$$b_{i-}(\omega_i, \omega_{i-1}) = \Delta t T_{i-1,i}(P_i - P_{i-1})\Lambda(\omega_{i-1}) \tag{5.26}$$

Next we assume that there is a homogeneous reservoir with V, ϕ_{0i} and T constants. The internal Jacobian row of the equation can be written as:

$$\begin{bmatrix} \gamma B_{i-1} b_{i^-} - \gamma \beta_{i-1} \times b'_{i^-,i-1} \\ A_i a_i - \gamma B_i b_{i^+} + \gamma(\beta_i \times b'_{i^+,i} + \beta_{i-1} \times b'_{i^-,i}) \\ \gamma \beta_i \times b'_{i+,i+1} \end{bmatrix}^T \tag{5.27}$$

where

$$A_i = \begin{bmatrix} \frac{\partial \alpha_i}{\partial \omega_i} \end{bmatrix} = \begin{bmatrix} \frac{\partial \alpha_i}{\partial p_i} & \frac{\partial \alpha_c}{\partial z_{i,1}} & \dots & \frac{\partial \alpha_c}{\partial z_{i,nc-1}} \end{bmatrix}, c = 1, ..., n_c \tag{5.28}$$

$$B_i = \begin{bmatrix} \frac{\partial \beta_i}{\partial \omega_i} \end{bmatrix} = \begin{bmatrix} \frac{\partial \beta_c}{\partial p_i} & \frac{\partial \beta_c}{\partial z_{i,1}} & \dots & \frac{\partial \beta_c}{\partial z_{i,nc-1}} \end{bmatrix}, c = 1, ..., n_c \tag{5.29}$$

$$b'_{i-,i} = \begin{bmatrix} \frac{\partial \beta_{i-}}{\partial \omega_i} \end{bmatrix}^T, b'_{i+,i} = \begin{bmatrix} \frac{\partial \beta_{i+}}{\partial \omega_i} \end{bmatrix}^T \tag{5.30}$$

$$\gamma = \Delta t \frac{T^{ab}}{\phi_0 V} \tag{5.31}$$

5.4 Nonlinear preconditioning in OBL

For binary system, the OBL formulation would be almost the same as our natural formulation introducing by saturation in the previous case. Looking into our Beta Operator Fig 5.1, it has an inflection point very similar to the flux term f in the previous chapter.

Analogously, to the natural formulation in the previous chapter, we use the absolute maximum value of the first derivative to allow for maximum propagation of the waves downstream. Therefore, we use the inflection point as the initial guess for Newton solver.

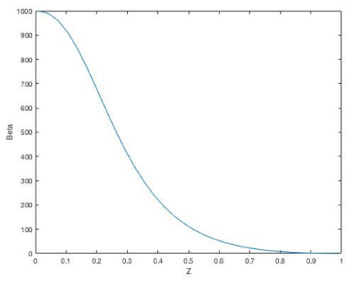


Figure 5.1: S-shape beta operator for two-phase flow for viscous forces

5.5 Trust Region Newton Solver in Simulation Prototype OBL

Although different advanced nonlinear solvers were developed for the natural formulation, there is a lack of advanced strategies for the molar formulation. A version of a trust-region correction was developed for molar formulation [26] but it still lacks robustness compared with the natural formulation. this can be explained by more-complicated nonlinear update procedure, which requires performing an exact flash for every block at a two-phase state in each nonlinear iteration. This problem can be solved by using OBL approach. In the OBL approach, all properties involved in the governing equations are lumped in a few operators, which are parameterized in the parameter space of the simulation problem wither in advance or adaptively during the simulation process. The conrol on the size of the parameterization hypervolume helps to preserve the balance between thr accuracy of approximation and the performance of the nonlinear solver. In the current version of the OBL approach, we do not reduce the number of unknowns and only use the fact that the physical description (i.e., fluid properties) is represented using piecewise linear interpolation. We segment the parameter-space of the nonlinear problem into a set of regions where our hyperbolic unknowns maintain their second order behavior (i.e., they remain either convex or concave). The proposed nonlinear solver locally constraints the updating of the overall compositions across the boundaries of these regions. Essentially, it is a cell-wise chopping strategy guided by trust regions of the operators. Our trust-region Newton method ensures that two successive iteration cannot cross any trust-region boundary. In other words, we chop after crossing each inflection point

5.6 Numerical example

5.6.1 Binary System

In this case, we have two independent variables [P, Z]. To find inflection point(s) of the convection operators, for a fix pressure we find the inflection point. Next, updated the pressure and find the other inflection point in the case it exists. The analysis of finding the inflection point is based on the linear interpolation of the second derivative.

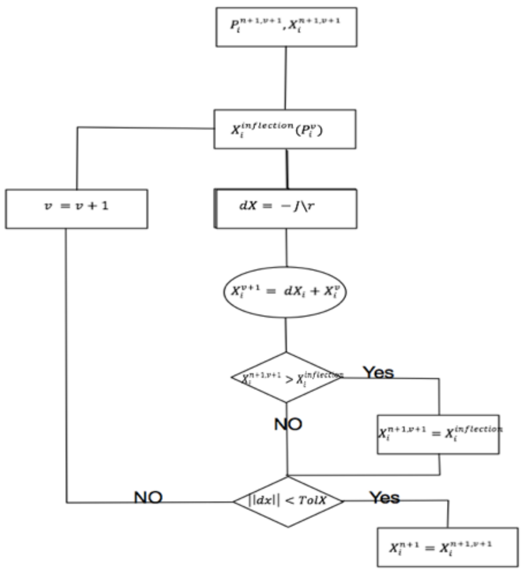

$P_i^{n+1,v+1}, X_i^{n+1,v+1}$
$X_i^{inflection}(P_i^v)$
$v = v + 1$
$dX = -J\backslash r$
$X_i^{v+1} = dX_i + X_i^v$
$X_i^{n+1,v+1} > X_i^{inflection}$
Yes
NO
$X_i^{n+1,v+1} = X_i^{inflection}$
NO
$\|dx\| < TolX$
Yes
$X_i^{n+1} = X_i^{n+1,v+1}$

Figure 5.2: Flowchart of the modified newton method for one time step considering also pressure update and correction of inflection point for pressure

5.6.2 Dead Oil

In this case, we do not expect to see the dependency of the inflection point of our beta operator with pressure; therefore, we have only one inflection point for which we have to be careful not to pass in our newton update. Fig. 5.4 summarize the performance of different solvers for the fracture model and it is clear that the trust region solver with preconditioning works significantly better than the other solvers. Note that since the preconditioning strategy only provides initial guesses for the saturation solution, when it is used in the fully-implicit method, only the initial guesses for saturation variables are modified to be the inflection point, the initial guesses for the pressure variables are unaffected.

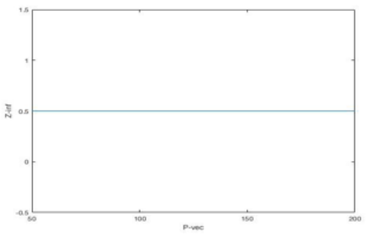


Figure 5.3: Inflection point of our beta operator for different pressure

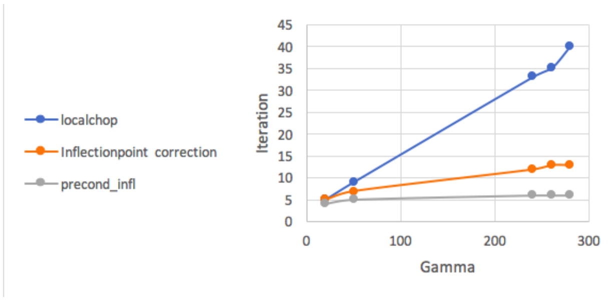


Figure 5.4: Comparison of different solvers with Corey exponents 2 2 and the resolution with 32 OBL resolution and the oil viscosity 1.5cp, water viscosity 1cp.

Water-Wet Sandstone

Krw_max	Nw(water corey exponent)	No(Oil exponent)	Water_viscosity	Oil_Viscosity
0.5	4	2	1	5

Table 5.1: Water-wet sandstone table

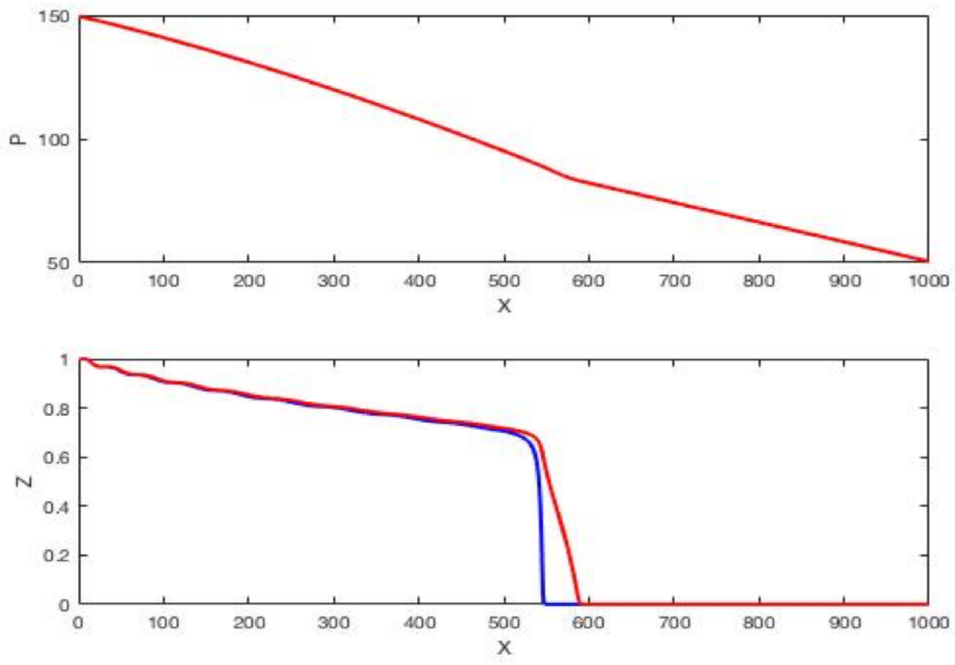


Figure 5.5: Propagation of the solution in fracture for one timestep [240days] in water-wet sandstone

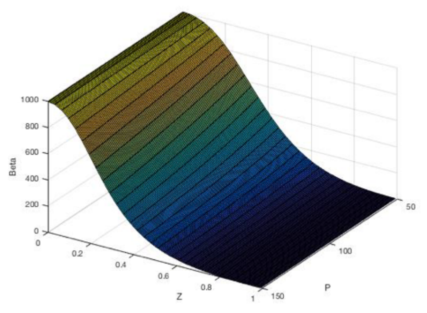
Another example here, in a water wet system comparing the different solvers performance table. 5.2

Table 5.2: Different Solvers behavior in the water-wet sandstone example

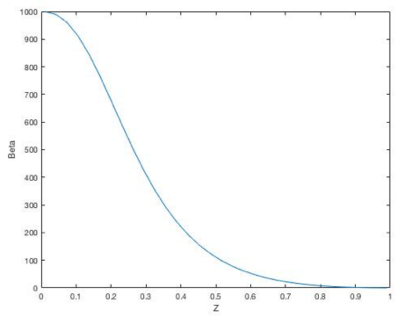
Timestep[days]	without pre localchop=0.1	without pre	with pre	inflection point
240	52	14	7	0.5

5.6.3 Viscosity variation with Pressure

Here for the dead oil example, we make the case where the inflection point varies also with the pressure for the beta operator. To do so we make the viscosity a function of pressure and thus the inflection point varies for different pressure.



(a)



(b)

Figure 5.6: (a) Beta operator variation with composition given the fix Pressure Beta Operator (b) Beta Operator for the dead oil variation of viscosity with pressure

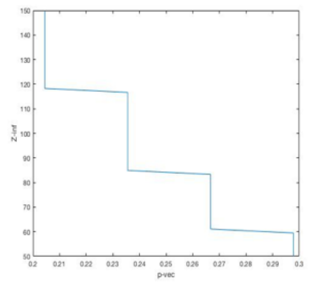
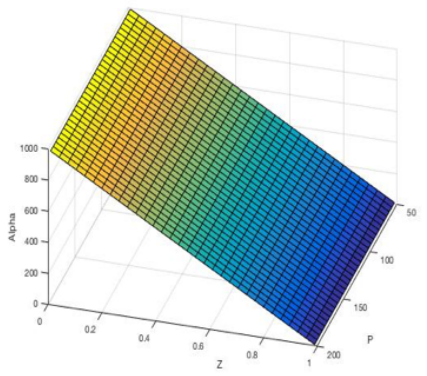
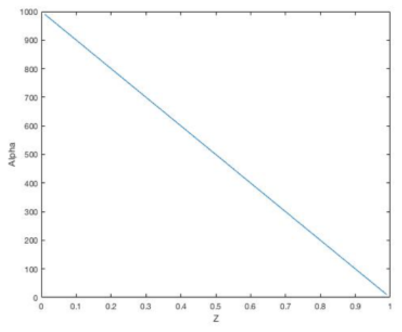


Figure 5.7: Inflection point variation with Pressure for dead oil with variation of viscosity with pressure example



(a)



(b)

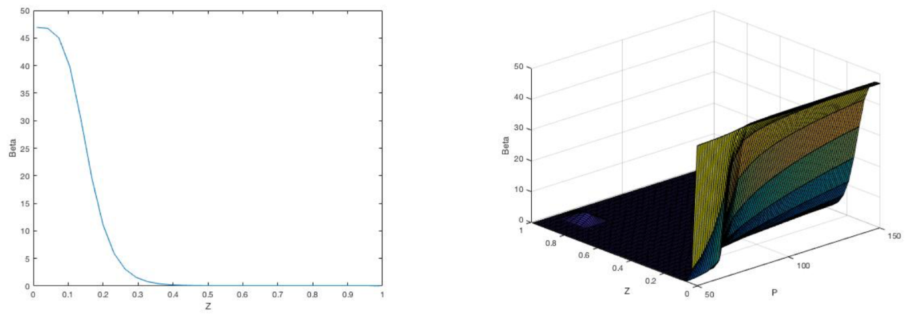
Figure 5.8: (a) Alpha operator variation with composition given the fix Pressure (b) Alpha operator for the dead oil

Table 5.3: Performance behavior of the different solvers in the case of dead oil for fractured reservoir

Gamma	Global chop	Trust Region solver	Trust region with precond
20	3	4	4
50	6	5	5
100	10	6	5
280	15	8	5
300	16	8	5

In this case, Alpha operator linearly changing with composition for all the pressure and thus never introduce problem for our trust region solver. We compare the performance of different solvers in the fracture reservoir in this case. To imitate it, i ran the simulation until a particular time ($T = 5000days$) Next, only by enlarging one timestep we compare the behavior of our different solvers. In the next step by introducing the infection point for a preconditioning we enhance the performance even further. Table 5.3 illustrates the different behavior of our solver.

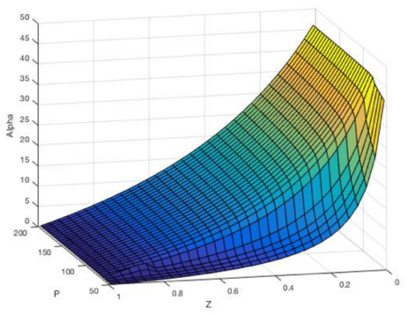
5.6.4 Supercritical CO2 injection



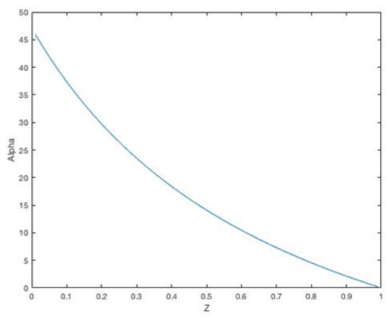
(a) **(b)**

Figure 5.9: (a) Beta operator variation with composition given the fix Pressure (b) Beta Operator

From the operators graph, it is clear that the alpha operator is not anymore linearly change with composition since the density is changing but still it keeps its second order behavior. On the other hand, Beta



(a)



(b)

Figure 5.10: (a) Alpha Operator (b) Alpha operator variation with composition given the fix Pressure

operator seems to be problematic due to the presence of inflection point. Therefore, we make our trust region solver in a way it does not cross its inflection point.

Table 5.4 shows the comparison between different nonlinear solvers, trust region solver, global chopping and local chopping with $dx = 0.1$ for a fracture reservoir while i ran the simulation for the maximum time 6000 days with the timestep = 10 days imitating the fluid flow moving in the matrix.

Table 5.4: Performance Of Different Solvers (running the simulation until 6000 days with the small timestepeuqals to 10days)

Solver	Total number of Newton iteration
Inflection point correction	838
Global chopping	1152
Local chopping (dx=0.1)	Diverge

As we can see the trust region works significantly better than the Global chop.

Graph 5.11 and table 5.5 illuatrate the comparison of fractured reservoir in the case of sCO2 injection. Moreover, the performance of the trust region solver enhances further by preconditioning.

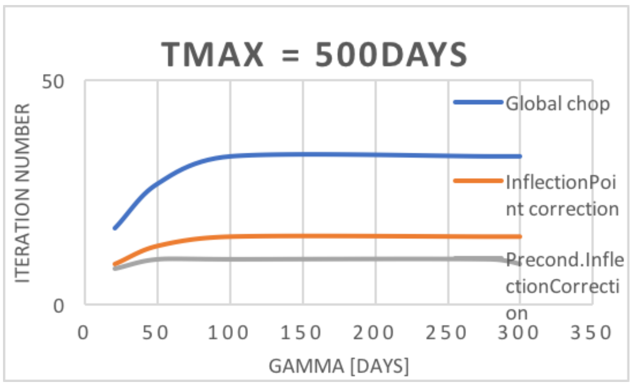


Figure 5.11: Inflection point variation of the operator beta with pressure in the case of sCO2 injection

Gamma	Global chop	Trust Region Solver	Trust Region with precond
20	17	9	8
50	27	13	10
100	33	15	10
280	33	15	10
300	33	15	9

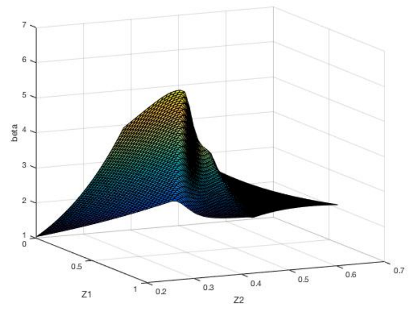
Table 5.5: sCO2 injection

5.7 Three component case

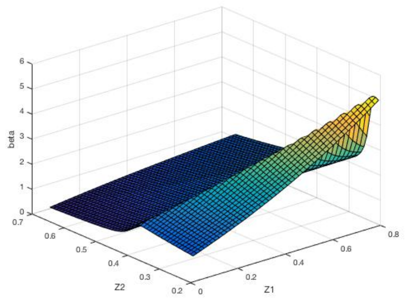
5.7.1 Operators analysis

To analyze the operators in this case and visualization we fix the pressure and plot the 3D graph of our operators for both the first and second composition.

By also fixing the second composition, we can see how beta varies in 1D with first composition.

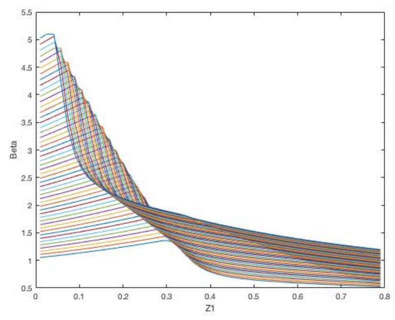


(a)

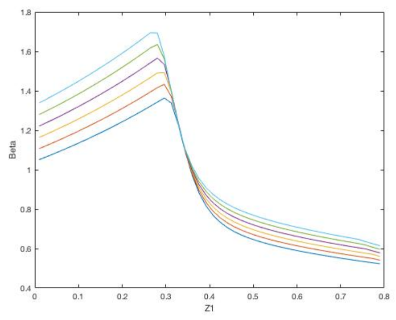


(b)

Figure 5.12: (a) Beta operator for the first composition (b) Beta Operator for the first composition from different angle

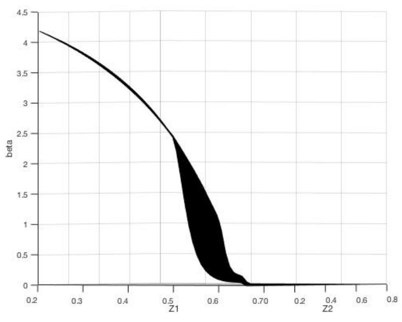


(a)

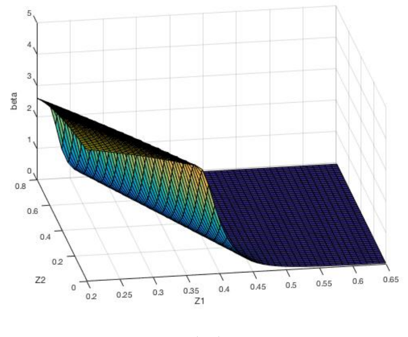


(b)

Figure 5.13: Beta variation as a function of first component given that all other unknowns are fix. Different graph correspond to different value of second composition



(a)



(b)

Figure 5.14: (a) Beta operator for the first composition (b) Beta Operator for the first composition from different angle

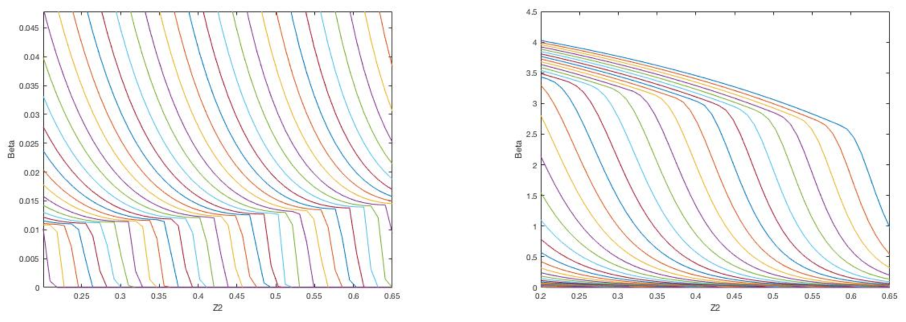


Figure 5.15: (a) Zoom into the graph(b) where the second inflection exist (b) Beta variation as a function of first component given that all other unknowns are fix. Different graphs correspond to different value of second composition

In the graphs 5.15 , for the second composition by fixing other unknowns and visualize how our operator varies with second composition. Zooming into the operator we can notice that another inflection point exists almost near to the zero that correspond to the discontinuity of density at bubble point

5.7.2 Detecting the inflection point

In the preliminary analysis, based on the analysis of our operators behavior , i found the inflection point of our operator by linear interpolation of the second derivative for all the points in my parameter space. In this case, i have the matrix of inflection for the first composition in which each element of my matrix correspond to the inflection point for the fix value of pressure and second composition. Graphs 5.16 illustrate our inflection matrix for the first composition.

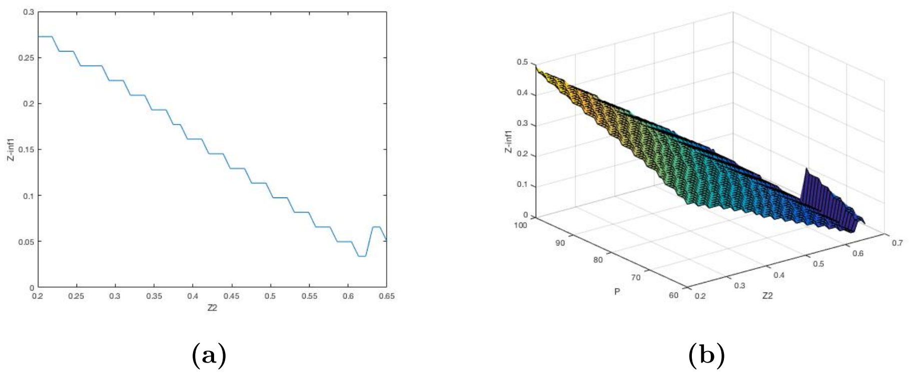


Figure 5.16: (a) Inflection variation of the first composition with the second composition given the fix pressure (b) Inflection curve of the first composition

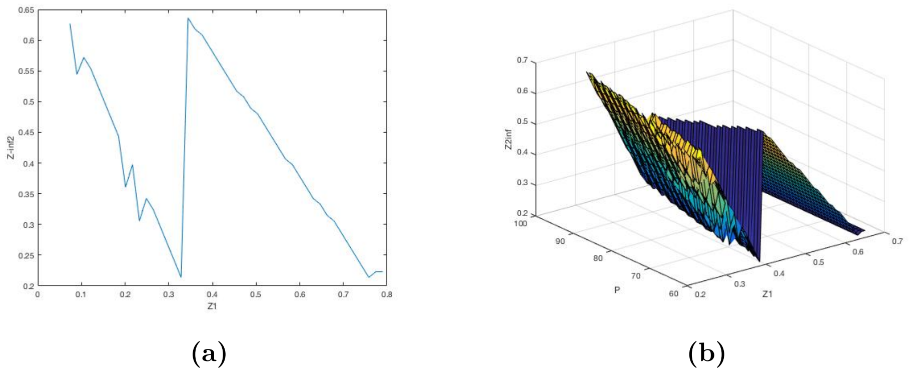


Figure 5.17: (a) Inflection variation of the second composition with the first composition given the fix pressure (b) inflection curve of the second composition

From the inflection graphs of the second compostion 5.17 we can see that it seems we have two families of inflection points which correspond to the inflection point due to the discontinuity of the density at bubble point. Finally, In ternary system it can exist more than one inflection for each composition given the fix other compositions.

Table 5.6 shows the comparison between different nonlinear solvers, trust region solver, global chopping and local chopping with $dx = 0.1$ while i ran the simulation for the maximum time 10000 days with the the aggressive timestep = 500 days.

Table 5.6: Performance of different solvers in three components case. (Running the simulation until 10000 days with the very aggressive time step 500 days)

Advanced nonlinear Solver	Total number of Newton iteration
Inflection point correction	154
Global chopping	201
Local chopping (dx= 0.1)	205

Chapter 6

Conclusion and future work

Simulation of multiphase flow in reservoirs with complex heterogeneous structures requires robust nonlinear solvers. The main source of nonlinearity is related to an implicit approximation of flux term in conservation equations which is required for the robustness (unconditional stability) of reservoir simulation process. In the absence of gravity and capillary, the flow problems dont usually introduce significant di culties to nonlinear solver. However, the solution of the transport problem often requires the propagation of displacement front to multiple control volumes per single timestep. This problem became especially serious in the limiting case of heterogeneous property distributions related to fractured reservoirs.

We developed the Advance nonlinear solver for fractured reservoirs in 1D. This solver is developed based on the trust region technique to get the global convergence and then we design our solver better by applying preconditioning strategy to get the better performance of the trust region solver. Moreover, introducing our relative permeability in the table and interpolate it rather than analytical formula we make our solver more real applicable since in the industry the relative permeability is estimated. Next, analyze the coarsening and resolution on the performance of the trust-region solver. Finally, we extend our frame work in the simulation prototype based on OBL (Operator based linearization). In this case, we consider more general situation by coupling the transport and flow and extend it to the compositional flow problems.

The future work can be in a way that we find the inflection point adaptively rather than preliminary analysis. because of hyperbolic nature of overall composition, the vast majority of parameter space remains unused. In other words in the newton update we find the inflection point and check trajectory whether pass it or not. Adaptive

Parameterization with buoyancy has been shown by Voskov and Khait [28] rather than preprocessing and storing OBL tables. The total size of the interpolation tables is defined by the number of dimensions N and the number of interpolation points n as n^N. Although the dimensionality depends on the number of components and thermal assumptions in a problem of interest, the number of interpolation points corresponds to the desired accuracy of the physical representation. Therefore, parameterization at the preprocessing stage would require a substantial amount of memory for the multicomponent systems modeled at a high interpolation precision. Furthermore, the necessity of searching supporting points (i.e., operator values) for every interpolation in a huge array of data affects the performance of the simulation. adaptive parameterization in space with adaptively finding the inflection point is one of our interest due to the reduce of the cost of the data storage.

Another future work will be related to extension of our preconditioning strategy to compositional problems unconditional to the number of components.

Appendix A

Matlab code of 1D implicit transport trust-region nonlinear solver

Matlab Code

```matlab
close all
clear all
clc

%global Ut Phi dx Nxx

%% Constant definitions:
no =  %input('Set a value for Corey exponent no ? ');
nw =   %input('Set a value f or Corey exponent nw ? ');
M  = %input('What is the viscosity ratio ?        ');
KRW= %input('What is the end point relative Permeability of ...
    water? ');
KRO= %input('What is the end point relative Permeability of ...
    oil ? ');
swi= %input(' What is the initial water Saturation? ');
sor= %input(' What is the residual oil Saturation? ');
Ut = 1;
Phi = 1;
%% time and spatial vectors definition:
dt = [0.001 0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009 ...
    0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 0.12 ...
    0.14 0.16 0.18 0.2];
tf = 0.3;
t = 0:dt:tf;
Nt = length(t);
Nx = 500;
x = linspace(0,1,Nx);
```

```
dx = x(2) - x(1);
%% function definition
krw = @(S) KRW*S^nw;
kro = @(S) KRO*(1-S)^no;
S_ef = @(S) (S-swi)/(1-swi-sor);
%%  Relative permeability Table
Nxx= 500;
krw_table = zeros(Nxx,1);
kro_table = zeros(Nxx,1);

S_table = (linspace(0,1,Nxx))';
h = S_table(2);
for i=1:length(S_table)
krw_table(i) = krw(S_table(i)) ;            % arbitrary ...
    analytic function
kro_table(i) = kro(S_table(i));
end
f_table = krw_table./(krw_table+kro_table/M); %i have to ...
    check after liunch

%% find inflection point
dff_table = zeros(Nxx,1);
for n =1:Nxx-2
    dff_table(n,1) = ...
        (f_table(n+2)-2*f_table(n+1)+f_table(n))/h^2;
end
 idx = min(find(dff_table <0))  % more than one dimention ...
     its okay but a lot of problem
 S_inf =(S_table(idx)+S_table(idx+1))/2  ...
     %(S_table(idx)*S_table(idx+1))^0.5  0.3870 ;
  Rel_table= table(S_table, krw_table,kro_table,f_table, ...
      dff_table);%
%% Boundary conditions and initial conditions:
S = zeros(Nx,2);
Sw= zeros(Nx,1);
dS = zeros(Nx,1);
S(:,1) = 0;
S(1,:) =(1-sor-swi)/(1-sor-swi);
f = zeros(Nx,1);
df = zeros(Nx,1);
f(:,1) = 1;
df(:,1) = 0;
r = zeros(Nx,1);
J = zeros(Nx,Nx);
J(1,1) = 1;

%% Newton Iteration:
tol = 1e-6;
for n = 1:Nt-1   % n=1 is initial condition, but at each ...
    step n+1 is calculated --> start from n=1 (n+1=2) ...
    untiln = Nt-1
    S(:,2) = S(:,1);
    for n_iter = 1:50       % choose the max number of ...
        iterations (=50)
```

```
[dS(:,1),J,r] = Newoton_Raphson(S,tol,S_table,f_table, dt(1));
%  Global Convergence using inflection Point
 for i = 1:Nx
 if (S(i,2) < S_inf && S(i,2)+dS(i,1) > S_inf)
    S(i,2) = S_inf + tol*10;
 elseif  S(i,2) > S_inf && S(i,2)+dS(i,1) < S_inf
     S(i,2) = S_inf - tol*10;
 else
     S(i,2) = S(i,2) + dS(i,1);
 end
 end
        if (norm(dS(:,1)) < tol)
            break
        end

    end
   Sw(:,1) = swi + (1-swi-sor)*S(:,2);
    figure(1)
    plot(x,Sw(:,1))

    axis([0 1 0 1]);
    pause(0.001)
    S(:,1) = S(:,2);
end
norm(dS(:,1))

%% only evaluate at one time step after t1
%S(:,1) =  0;

%S(1,:) = (1-sor-swi)/(1-sor-swi);
niter =zeros(length(dt),1);
for n=1
    for t=1:length(dt)
   % Initial guess selection (three choices that has to be ...
       chosen manually)
        % Preconditioning (using inflection point as an ...
            initial guess)
         S(2:end,n+1) =   S_inf;
          % Using initial condition as an initial guess
        %S(:,n+1) = S(:,n);
          % Using the last update for S(where dt=0.001 and ...
              t=0.3) as an initial guess
         %S(:,n+1) = S(:,end);
    for n_iter = 1:600
        [dS(:,1),J,r] = ...
            Newoton_Raphson(S,tol,S_table,f_table, dt(t));
% Inflection Point
 for i = 1:Nx
if (S(i,2) < S_inf && S(i,2)+dS(i,1) > S_inf)
   S(i,2) = S_inf + tol*10;
   elseif  S(i,2) > S_inf && S(i,2)+dS(i,1) < S_inf
    S(i,2) = S_inf - tol*10;
else
    S(i,2) = S(i,2) + dS(i,1);
```

```
end
end

        if (norm(dS(:,1)) < tol)
            break
        end
    end
    Sw(:,1) = swi + (1-swi-sor)*S(:,2);
niter(t,1) = n_iter;
figure(2)
%plot(x,S2(:,1),'b'),
plot(x,Sw(:,1),'c') , hold on
    end
    figure(3)
    plot(dt,niter)
end
norm(dS(:,1))
%
%% Appleyard chop
S(:,1) = 0; %swi;
S(1,:) = (1-sor-swi)/(1-sor-swi);

for n = 1:Nt-1
     S(:,2) = S(:,1);
for n_iter = 1:500
[dS(:,1),J,r] = Newoton_Raphson(S,tol,S_table,f_table, dt(1));
 %modifying the solver appleyard chop
         for i =1: Nx
     if dS(i,1)>0.2
         dS(i,1) = 0.2;
     end
         end
        if (norm(dS(:,1)) < tol)
            break
        end
        S(:,2) = dS(:,1) + S(:,2);
end
Sw(:,1) = swi + (1-swi-sor)*S(:,2);
    figure(4)
    plot(x,Sw(:,1))
    axis([0 1 0 1]);
    pause(0.001)
    S(:,1) = S(:,2);
end


%% function def

function [dS, J, r] = Newoton_Raphson(S,tol,S_table,f_table,dt)
%global Ut Phi dx Nxx
Nx = length(S);
f = zeros(Nx,1);
df = zeros(Nx,1);
h = S_table(2) - S_table(1);
```

```
f(:,1) = 1;
df(:,1) = 0;
r = zeros(Nx,1);
J = zeros(Nx,Nx);
J(1,1) = 1;
for m = 2:Nx          % pay attention that m-1 is alos used ...
    --> start from m = 2, and m = 1 is BC
    % !!pay attention to introduce the f at m = 1
    % !! pay attention to introduce df at m = 1
    j = ceil(S(m,2)/S_table(2));
    if j == 0 || j<0
        j =1;
    elseif j≥ Nxx
        j= Nxx-1;
    else
        j=j;
    end
     % !! pay attention to introduce df at m = 1

     f(m,1) = f_table(j) + (f_table(j+1)-f_table(j))/h * ...
         (S(m,2)-S_table(j)) ;
    df(m,1) =  (f_table(j+1)-f_table(j))/h ;
%  if f(m,1) < 10*tol  %this constraint makes our solver ...
    perform similar as analytical solution
%  f(m,1) = 0;
%  df(m,1) =0;
%  end
r(m) = (S(m,2) - S(m,1))/dt + Ut/Phi * (f(m,1) - ...
    f(m-1,1))/dx;    %!! pay attention to evaluate r at m = 1

  J(m,m) = 1/dt + Ut/(Phi*dx)*(df(m,1));   %!! pay ...
      attention to introduce the J at m = 1
  J(m,m-1) = -Ut/(Phi*dx)*(df(m-1,1));
end
dS(:,1) = - J\r;
end
```

Appendix B

AD-GPRS

The Automatic Differentiation General Purpose Research Simulator (AD-GPRS) is a flexible and extensible multiphysics simulation platform. It employs automatic differentiation to construct the Jacobian allowing for an easy extension to new physics and constitutive relations, as well as for complete flexibility in the specification of independent variables, which leads to a unified simulator for different formulations and solution strategies. There are no assumptions about the underlying grid structure thus unstructured grids are supported for accurate representation of the complex structure and heterogeneity of subsurface formations. Fully implicit or sequentially implicit time-discretization schemes are available. The latter is designed for handling different physical sub-problems with flexible coupling strategies.

AD-GPRS can be used, for example, to simulate enhanced oil recovery (EOR) processes, CO2 sequestration in saline aquifers and depleted oil reservoirs, shale gas/oil production, and enhanced steam injection.

Currently, AD-GPRS consists of the following modules:

Flow General component-based reservoir simulation tool. Several widely used variable formulations (natural and molar) and solution strategies are incorporated, including Black Oil, fully EoS (two- and three-phase, as well as support for external libraries), and K-value formulation.

Thermal Adds support for thermal-compositional flows, for example, for simulation of steam injection or geothermal reservoirs.

Geomechanics Simulates complex mechanical behavior including plastic deformation and thermal effects. Helps to understand physical processes for fractured and faulted reservoirs. Incorporates poroelastic, thermo-elastic, and general poro-thermo-plastic models with complete flexibility in adding new constitutive relations. Can be used, for instance, for coupled simulations of gas production from a naturally fractured reservoir, steam-assisted gravity drainage (SAGD).

Chemical reactions Capable of modeling kinetic and equilibrium reactions. Supports both natural and overall-composition variable formulations. Adds a key component for simulation of in-situ upgrading of oil-shale or CO2 sequestration in saline aquifers.

Wells The wellbore flow and the near-well flow behavior have an important impact on well performance. Different techniques have been developed to model multiphase flow through the wellbore. We account for thermal and compositional effects. Currently supported well models include standard well, multi-segment well, and heater model.

Appendix C

Inflection point detection from OBL table code

Here is my code for finding the inflection point numerically by second derivative analysis of my operators. The code is written in a way it can be extended easily to compositional problems unconditional to the number of the components. However, we shoud be carefull by increasing the number of composition we should add the corresponding outer loop as well.

Matlab Code

```
function [z_inf, ddBB, BB, dBB_inf] = ...
    inflection_general(tbl, p, zz, z_vec, ind_comp, use_der )
% INPUTS:
%   tbl:         Table
%   p:           Pressure
%   zz:          Vector with value for components
%   z_vec:       Vector with values for the changing component
%   ind_comp:    Index for the changing component
%   use_der:     (optional, default false) If true, computes ...
    the second
%                derivatives using the first derivatives ...
    from the table.

dz = max(diff(z_vec));
npoints = length(z_vec);
ncomp   = length(zz);

XX = zeros( npoints, ncomp+1 );
XX(:,1) = p;
for i=1:ncomp
    if i==ind_comp
```

```
        XX(:,i+1) = z_vec;
    else
        XX(:,i+1) = zz(i);
    end
end

% If the optional input use_der is not used when calling ...
    the function, it
% default to false
if nargin<6
    use_der = false;
end

if use_der
    [ ¬, dBB ] = tbl.interpolate_v2(XX);
    dBB = dBB( (1+ind_comp):(1+ncomp):npoints*(1+ncomp) );
    ddBB = ( dBB(3:end) - dBB(1:end-2) )/(2*dz);
else
    [ BB, ¬ ] = tbl.interpolate_v2(XX);
    ddBB = ( BB(3:end) - 2*BB(2:end-1) + BB(1:end-2) )/(dz*dz);
    dBB = ( BB(3:end) - BB(1:end-2) )/(2*dz);
end



%inf_indx = find( ddBB(1)*ddBB<0, 1, 'first' ); % ...
    adapptively do it (1d doesnt make sense) go by ...
    trajectory second derivative for every intervalinthe ...
    update, for the given pressure i look to the update of ...
    z , i need to check im not crossing the inflection point.
inf_indx = find( ddBB(1:end-1).*ddBB(2:end) < 0 );

if isempty(inf_indx)
    z_inf = NaN;
    dBB_inf = NaN;
else
    z_inf   = (z_vec(inf_indx+1)+z_vec(inf_indx))/2;
    dBB_inf = dBB(inf_indx);
end

end
```

Bibliography

[1] K. Aziz and A. Settari. *Petroleum Reservoir Simulation.* K. Aziz & A. Settari, 2002.

[2] Hui Cao. Development of techniques for genenral purpose simulators,. 18:264–273, 12 202.

[3] Rustem; Obi Eguono Cao, Hui; Zaydullin. Nonlinear convergence for near-miscible problem: A mystery unveiled for natural variable simulator. 18:264–273, 12 202.

[4] Keith H. Coats. An equation of state compositional model,. 18:264–273, 12 202.

[5] K.H. Coats. Impes stability: Selection of stable timesteps. s 82–83:101–111, 02 2003.

[6] Li YK Grabonstotter JE Collins, Nghiem. An efficient approach to adaptive- implicit compositional simulation with an equation of state. 83:111, 02 1992.

[7] P. Deuflhard. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms.* Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2011.

[8] P.A. Forsyth and P.H. Sammon. Practical considerations for adaptive implicit methods in reservoir simulation. *Journal of Computational Physics*, 62(2):265 – 281, 1986.

[9] William D. Gropp, Dinesh K. Kaushik, David E. Keyes, and Barry F. Smith. High-performacne parallel implicit cfd. *Parallel Comput.*, 27(4):337–362, March 2001.

[10] M. D. Hebden. An algorithm for minimization using exact second derivatives. 18, 12 1973.

[11] Patrick Jenny, Hamdi A. Tchelepi, and Seong H. Lee. Unconditionally convergent nonlinear solver for hyperbolic conservation laws

with s-shaped flux functions. *Journal of Computational Physics*, 228(20):7497 – 7512, 2009.

[12] L.J.; Aziz K. Karimi-Fard, M.; Durlofsky. An efficient discrete fracture model applicable for general purpose reservoir simulators. *Journal of Computational Physics*, 62(2):265 – 281, 1986.

[13] Herbert Bishop Keller, Bangalore Indian Institute of Science, and T.I.F.R.-I.I.Sc. Programme in Applications of Mathematics. *Lectures on numerical methods in bifurcation problems.* Berlin : Springer-Verlag, published for the TATA Institute of Fundamental Research, 1987. "Lectures delivered at the Indian Institute of Science, Bangalore, under the T.I.F.R.-I.I.Sc. Programme in Applications of Mathematics".

[14] David Keyes. Terascale implicit methods for partial differential equations. 01 2002.

[15] Mark Khait and Denis V. Voskov. Operator-based linearization for general purpose reservoir simulation. *Journal of Petroleum Science and Engineering*, 157:990 – 998, 2017.

[16] Felix Kwok and Hamdi Tchelepi. Potential-based reduced newton algorithm for nonlinear multiphase flow in porous media. *Journal of Computational Physics*, 227(1):706 – 727, 2007.

[17] Kenneth Levenberg. A method for the solution of certain nonlinear problems in least squares. 18, 12 1973.

[18] Michael L. Michelsen. The isothermal flash problem. part ii. phase-split calculation. *Fluid Phase Equilibria*, 9(1):21 – 40, 1982.

[19] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1970.

[20] T. Plewa, T. Linde, and V.G. Weirs. *Adaptive Mesh Refinement - Theory and Applications: Proceedings of the Chicago Workshop on Adaptive Mesh Refinement Methods, Sept. 3-5, 2003.* Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2005.

[21] S.; Moridis G.; Oldenburg C.; Wu Y. Pruess, K.; Fensterle. General purpose reservoir simulators: the tough2 family. *Journal of Computational Physics*, 62(2):265 – 281, 1997.

[22] J R. Wallis, Richard Kendall, and Todd Little. Constrained residual acceleration of conjugate residual methods. 02 1985.

[23] Kristian; Michelsen Michael L.; BjurstrÃ¸m Kersti E. Rasmussen, Claus P.; Krejbjerg. Increasing the computational speed of flash calculations with applications for compositional, transient simulations. 18:264–273, 12 202.

[24] T.F. Russell. Stability analysis and switching criteria for adaptive implicit methods based on the cfl. 18, 12 2012.

[25] J.G.; Stone H.L. Spillette, A.G.; Hillestad. A high-stability sequential solution approach to reservoir simulation. s 82–83:101–111, 02 1973.

[26] Denis Voskov and Hamdi Tchelepi. Comparison of nonlinear formulations for two-phase multi-component eos based simulation. s 82–83:101–111, 02 2012.

[27] Denis V. Voskov. Operator-based linearization approach for modeling of multiphase multi-component flow in porous media. *Journal of Computational Physics*, 337:275 – 288, 2017.

[28] Khait Voskov. Adaptive parameterization for solving of thermal/compositional nonlinear flow and transport with buoyancy. s 82–83:101–111, 02 2018.

[29] Tchelepi Voskov. Compositional nonlinear solver based on trust regions of the flux function along key tie-lines. 18:264–273, 12 2011.

[30] Xiaochen Wang. Trust-region newton solver for multiphase flow and transport in porous media. 18, 12 2012.

[31] Rami Younis. 2011.

[32] Rami Younis, Hamdi Tchelepi, and Khalid Aziz. Adaptively localized continuation-newton method–nonlinear solvers that converge all the time. 15:526–544, 06 2010.

[33] Rustem Zaydullin, Denis Voskov, and Hamdi Tchelepi. Nonlinear formulation based on an equation-of-state free method for compositional flow simulation. 18:264–273, 12 2012.