Abstract
A new approach to model multiphase flow in porous media with phase appearance and disappearance is to use nonlinear complementarity constraints. In addition to the standard mass balance equations for multiphase flow, a set of complementarity constraints are added in order to handle the phase transitions. This results in a larger system of nonlinear equations, which is non-smooth in general. The system is subsequently solved using a semi-smooth Newton method. In this project, we aim to investigate this new approach and implement it in Amanzi, which is a part of the Advanced Simulation Capability for Environmental Management (ASCEM) program.
1. Background

Modeling multiphase flow in porous media is important for many applications, such as groundwater storage, oil and gas extraction, contaminant transport, and CO2 sequestration. At the same time, developing numerical methods for multiphase flow is a challenging task, especially when phase transitions needs to be taken into account. The system of equations describing multiphase flow consists of difficult nonlinear conservation laws, constitutive laws (e.g., Darcy’s law), and algebraic constraints (saturation, molar fractions, etc.). Usually, one can use the constraints to eliminate some of the unknowns to get a reduced set of primary variables. For example, in a two-phase flow involving gas and liquid phases, using the constraint that the gas and liquid saturations sum to 1, we can eliminate the gas saturation and choose the liquid saturation as a primary variable. However, if the liquid phase disappears, this reduced set of equations consists of variables that are not well defined.

The traditional approach to overcome this problem is primary variable switching (PVS). The primary variables may be switched depending on the conditions of the phases. One drawback of PVS is that its global convergence behavior is irregular, especially if the solution is locally close to a point where fluid phase appears or vanishes [Class and Helmig 2002]. Another drawback is that the solver for PVS relies on a serial algorithm that may not perform well on emerging architecture, especially at exascale. A scalable solver such as multigrid, however, cannot be applied directly, and needs extensive customization for PVS. Thus, our goal is to find a robust algorithm that can seamlessly handle phase transitions and is more readily adapted to multilevel solvers. Besides PVS, other approaches have been developed to address phase transitions. For example, one can formulate the balance equations using gas concentration in the liquid phase and saturation [Bourgeat et al. 2009], or total concentration and saturation [Bourgeat et al., 2013], instead of phase pressure and saturation formulation. One can also extend the saturation of one phase and allow it to be negative or greater than 1 [Abadpour and Panfilov, 2009]. A more recent approach is to use nonlinear complimentarity constraint (NCP) in additions to the balance equations [Lauser et al. 2011]. This is the method we want to implement in this project.

2. Project Goals

In this project, we aim to implement the NCP approach in a high-performance computing framework. By the end of the project, we want to provide the capability for modeling fully coupled, two-phase, two-component, non-isothermal, miscible flow. We will develop unit tests, and run the code on large and realistic problems if possible. We also aim to perform code verification and benchmark against existing codes and PVS approach.

3. Approach

Since the governing equations for multiphase, multi-component, non-isothermal flow with miscibility effects and phase transitions are quite complicated, we have to start code development with a couple of simplifications, and then add complexity over time. For phase I, we
assume the following:

- Immiscibility, no phase transition.
  - No molar fractions.
  - No additional constraints for local thermal dynamic equilibrium.

- Isothermal
  - No energy equation.
  - Densities, porosity, and viscosities independent of temperature.

- Incompressibility
  - The fluid and solid structure (rock matrix) is incompressible.
  - Densities and porosity constant in space and time.

Based on these assumptions, we formulate a set of coupled equations for pressure and saturation. We then discretize these equations using finite volume method with two point flux approximation (TPFA) in space, and implicit backward Euler method in time. The resulting discrete system of linear and nonlinear equations are solved using iterative methods such as preconditioned conjugate gradient and Newton method.

For phase II, we will start with adding miscibility effect. This requires formulating a more detailed mass balance equation that includes two components inside a phase. The consequence of having two components within a phase is that we increase the number of variables per cell, i.e. mass/molar fractions of the components. Initially we will keep the fractions constant, and the components can only advect or diffuse within the phase. Then, we will formulate the nonlinear complimentarity constraints (NCP) and introduce thermal dynamic equilibrium laws for phase transitions and mass transfer between the phases. Along with the NCP, we need to develop the semi-smooth Newton method, in order to solve the system resulting from incorporating the NCP equations into the system of mass conservation. Finally, we will include the energy equation for thermal effect.

### 3.1. Continuum Model

#### 3.1.1. Conservation Equations

For a control volume, we have the general balance equation:

$$\frac{\partial \xi}{\partial t} + \nabla \cdot \psi = f$$

(1)

where $\xi$ is the storage term, $\psi$ the flux term, and $f$ the source term.

For each phase $\alpha$, the above terms for mass conservation read:

$$\xi = \phi \rho_\alpha S_\alpha$$

(2)

$$\psi = \rho_\alpha q_\alpha$$

(3)

$$f = f_\alpha$$

(4)
in which
φ is the porosity.
S_α is the phase saturation.
ρ_α is the mass density of the phase
q_α is the Darcy’s velocity.

3.1.2. Constitutive Relations

We assume the phase velocity q_α follows Darcy’s law, and can be expressed in terms of the phase pressure P_α as

\[ q_α = -\frac{K_α}{μ_α} (\nabla P_α - ρ_α g) \]  

Here, K_α is the phase permeability, which depends on the phase saturation S_α

\[ K_α = k_{rα}(S_α) K \]  

in which k_{rα} is the relative permeability of phase α and K is the absolute permeability, which is independent of the fluid.

Substituting equation (6) into (5), we have

\[ q_α = -K k_{rα}(S_α) \frac{μ_α}{μ_α} (\nabla P_α - ρ_α g) \]  

The ratio \( λ_α = k_{rα}/μ_α \) is usually called phase mobility.

Moreover, the mass balance equations for the phases are coupled through the saturation constraint and capillary pressure.

\[ \sum_{α=1}^{M} S_α = 1 \]  

\[ P_n = P_w + P_c(S_w) \]

where the subscript w, n means “wetting” and “non-wetting” respectively. For example, in a flow with gas and liquid phases, the liquid phase is the “wetting phase” and the gas phase is the “non-wetting” phase. Various models can be used to calculate P_c(S_w), such as Van Genuchten and Brooks-Corey. Since capillary pressure is one main source of nonlinearity and produces a strong coupling between the equations at the same time, a more detailed treatment of capillary pressure is presented in a later section.

3.1.3. Relative Permeability

One source of nonlinearity comes from the relative permeability k_{rα} parametrized by the saturation S_α. Common models for relative permeability are
• Power law (Brooks-Corey type)

\[ S_{ea} = \frac{S_\alpha - S_{r\alpha}}{1 - \sum_\beta S_{r\beta}} \]  
\[ k_{r\alpha} = \left(S_{ea}\right)^n \]

where \(S_{ea}\) is the effective saturation of phase \(\alpha\).

• Van Genuchten [Genuchten 1980]

\[ k_{rw}(S_w) = \sqrt{S_{ew}\left(1 - \left(1 - S_{e1/m}\right)^m\right)^2} \]  
\[ k_{rn}(S_w) = \sqrt{1 - S_{ew}\left(1 - S_{e1/m}\right)^{2m}} \]

3.1.4. Capillary Pressure

Van Genuchten capillary pressure curve

\[ P_c = P_r(S_{we}^{-1/m} - 1)^{-1/n} \]
\[ S_{we} = \frac{S_w - S_{wr}}{1 - S_{wr} - S_{nr}} \]
\[ m = 1 - 1/n \]

where the porous medium dependent parameters are
\(P_r\) is the entry pressure,
\(S_{wr}\) is the residual saturation of the wetting phase,
\(S_{nr}\) is the residual saturation of the non-wetting phases,
\(n\) is the Van Genuchten parameter.

Brooks-Corey capillary pressure curve

\[ P_c = P_dS_{we}^{-1/\lambda} \]

where \(\lambda\) is the Brooks-Corey parameter and \(P_d\) is the entry pressure.

3.1.5. Pressure Equation

Here we are going to derive the pressure equation used in simulation of two-phase fluid flow. With incompressibility assumption, we can rewrite the mass balance equations as

\[ \phi \rho_w \frac{\partial S_w}{\partial t} - \rho_w \nabla \cdot \left(\lambda_w K (\nabla P_w - \rho_w g)\right) = f_w \]  
\[ \phi \rho_n \frac{\partial S_n}{\partial t} - \rho_n \nabla \cdot \left(\lambda_n K (\nabla P_n - \rho_n g)\right) = f_n \]

Divide both sides of each equation by the density and sum them together we get

\[ \phi \frac{\partial(S_w + S_n)}{\partial t} - \nabla \cdot \left(\lambda_w K (\nabla P_w - \rho_w g) + \lambda_n K (\nabla P_n - \rho_n g)\right) = \frac{f_w}{\rho_w} + \frac{f_n}{\rho_n} \]
Invoke the constraint $S_w + S_n = 1$, the first derivative vanishes and let us introduce total velocity

$$ q = q_w + q_n = -\left(\lambda_w K (\nabla P_w - \rho_w g) + \lambda_n K (\nabla P_n - \rho_n g)\right) $$

(17)

and total source term

$$ f_{tot} = \frac{f_w}{\rho_w} + \frac{f_n}{\rho_n} $$

(18)

The equation (16) becomes an elliptic PDE of the form

$$ \nabla \cdot q = f_{tot} $$

(19)

Further, let us introduce fractional flow quantity

$$ f_\alpha = \frac{\lambda_\alpha}{\lambda} $$

(20)

where $\lambda = \lambda_w + \lambda_n$ is the total mobility. Using this, we can rewrite the total velocity as follows

$$ q = -K \lambda \left(\nabla P_w + f_n \nabla P_c(S_w) - G\right) $$

(21)

in which we have substituted $P_n = P_w + P_c(S_w)$, and the modified gravity term is defined as

$$ G = \frac{\lambda_w \rho_w + \lambda_n \rho_n}{\lambda} g $$

(22)

### 3.1.6. Global Pressure

Notice that the formulation for total velocity in equation (21) will have the form of Darcy’s law in equation (5), if we can find a pressure $P$ such that

$$ \nabla P = \nabla P_w + f_n \nabla P_c(S_w) $$

(23)

In this case equation (21) turns into

$$ q = -K \lambda \left(\nabla P - G\right) $$

(24)

Following [Chavent and Jaffré, 1978], one form of $P$ is

$$ P = P_w + \pi_n $$

(25)

with

$$ \pi_n(S_w) = \int_{S_0}^{S_w} f_n(\xi) \frac{\partial P_c}{\partial \xi}(\xi) d\xi + \pi_0 $$

(26)

It is easy to check that such $P$ satisfying equations (25) and (26) also obeys (24).
3.1.7. Saturation Equation

Like the pressure equation, we start deriving an equation for saturation from one of the mass balance equations. It is common that we use the equation for saturation of the wetting phase, but there is no difference for the non-wetting phase.

\[ \phi \rho_w \frac{\partial S_w}{\partial t} + \rho_w \nabla \cdot q_w = f_w \]  

(27)

Here, we want to express the Darcy’s phase velocity \( q_w \) in terms of the global pressure \( P \) and the total velocity \( q \). Again, we have the Darcy’s velocities for the phases

\[ q_w = -\lambda_w K (\nabla P_w - \rho_w g) \]  

(28)

\[ q_n = -\lambda_n K (\nabla P_w + \nabla P_c - \rho_n g) \]  

(29)

Divide both sides by the mobility and take the difference to eliminate \( \nabla P_w \) we obtain

\[ \lambda_n q_w - \lambda_w q_n = K \lambda_w \lambda_n (\nabla P_c + (\rho_w - \rho_n) g) \]  

(30)

Substituting \( q_n = q - q_w \) and simplify we have

\[ q_w = f_w v + \lambda_n f_w K (\nabla P_c + (\rho_w - \rho_n) g) \]  

(31)

3.2. Complete System of Equations

Putting all the equations together, we want to solve the following system of coupled equations for incompressible 2-phase flow

\[ -\nabla \cdot \left( K \lambda (\nabla P - G) \right) = f_{tot} \]  

(32)

\[ \phi \frac{\partial S_w}{\partial t} + \nabla \cdot q_w = f_w / \rho_w \]  

(33)

\[ q_w = f_w v + \lambda_n f_w K (\nabla P_c + (\rho_w - \rho_n) g) \]  

(34)

\[ \lambda = \lambda_w + \lambda_n \]  

(35)

\[ \lambda_w = k_{rw} / \mu_w, \quad \lambda_n = k_{rn} / \mu_n \]  

(36)

3.3. Boundary Conditions

For the pressure equation, we have the following boundary conditions

- Dirichlet boundary condition, in which the pressure on the boundary is given.

\[ P = P_d \text{ on } \partial \Omega_D \]  

(37)

- Neumann boundary condition, in which the fluxes on the faces are given.

\[ q \cdot n = G_n \text{ on } \partial \Omega_N \]  

(38)
For the saturation equation, we have the following boundary conditions

- Dirichlet boundary condition, in which the saturation on the boundary is given.
  \[ S = S_d \text{ on } \partial \Omega_D \] (39)

- Neumann boundary condition, in which the fluxes on the boundary faces are given.
  \[ \mathbf{q}_w \cdot \mathbf{n} = \mathbf{G}_n \text{ on } \partial \Omega_N \] (40)

3.4. Discretization

We discretize both pressure and saturation equations using cell-centered finite volume method. This method has the advantage that it is a conservative method, meaning that mass is conserved locally, which is exactly what we need. Another advantage is that it can be easily formulated for unstructured meshes. For the scope of the first part of this project, we will limit ourselves to structured meshes.

3.4.1. Pressure Equation

We discretize the pressure equation as follows. First, for each control volume, integrate equation (32) and apply divergence theorem

\[ -\int_{\partial V_i} \lambda \mathbf{K} (\nabla P - \mathbf{G}) \cdot \mathbf{n} dS = \int_{V_i} f_{tot} \] (41)

Since the boundary of the control volume \( V_i \) contains a set of edges (or faces in 3D), the above equation is equivalent to

\[ -\sum_j \int_{\gamma_{ij}} \lambda \mathbf{K} (\nabla P - \mathbf{G}) \cdot \mathbf{n} dS = \int_{V_i} f_{tot} \] (42)

where \( \gamma_{ij} \) is the edge (or face) adjacent to cells \( i \) and \( j \).

The integral on the RHS is approximated using a cell-averaged value

\[ \int_{V_i} f_{tot} = V_i \bar{f}_{tot} \] (43)

The gradient of the global pressure is calculated using two-point flux approximation (TPFA), which is simply a center difference

\[ \nabla P = \frac{P_i - P_j}{\Delta x_i + \Delta x_j} \] (44)

in which \( \Delta x_i, \Delta x_j \) are the size in the x-direction of cell \( V_i, V_j \) respectively.

The value of \( \lambda \mathbf{K} \) on the face is computed using harmonic average

\[ (\lambda \mathbf{K})_{ij} = (\Delta x_i + \Delta x_j) \left( \frac{(\lambda \mathbf{K})_{ij} (\lambda \mathbf{K})_{ji}}{\Delta x_i (\lambda \mathbf{K})_{ji} + \Delta x_j (\lambda \mathbf{K})_{ij}} \right) \] (45)

which ensures that the flux over the edge (or face) \( \gamma_{ij} \) is continuous.
3.4.2. Saturation Equation

We use finite volume in space and backward Euler in time. For each control volume,

\[ V_\phi \frac{S_w^{n+1} - S_w^n}{\Delta t} + \int_{\partial V} f_w(S_w^m) \mathbf{q} \cdot \mathbf{n} = V \frac{q_w^m}{\rho_w} \]

In IMPES approach, \( m = n \), and for fully implicit approach, \( m = n + 1 \).

Since this is a nonlinear hyperbolic equation, in which sharp front and shock waves can occur, one has to be careful in choosing numerical scheme that avoids unphysical oscillations. One such scheme is to use upwinding for the fractional flow as follows

\[ f_w(S_w)_{ij} = \begin{cases} 
  f_w(S_w)_i & \text{if } \mathbf{q}_{ij} \cdot \mathbf{n} \geq 0 \\
  f_w(S_w)_j & \text{if } \mathbf{q}_{ij} \cdot \mathbf{n} < 0
\end{cases} \]

The total velocity \( \mathbf{v} \) is calculated from the solution of the pressure equation.

4. Moving Forward

The equations described above serve as our starting point for the project, and get us closer to a point where we can tackle the full system of equations for 2-phase, 2-component, non-isothermal, miscible flow. However, moving forward, we will have to slowly remove our simplifying assumptions and add more complexity. In particular, we will have to formulate more detailed mass balance equations. To do so, we will first introduce a couple of fractional quantities that are useful in analyzing the composition of the flow.

4.1. Flow Compositions

Component volume fraction

\[ C^k_\alpha = \frac{\text{Volume of component } k \text{ in phase } \alpha}{\text{Volume of phase } \alpha} \quad (46) \]

Mass fraction of a component

\[ X^k_\alpha = \frac{\text{Mass of component } k \text{ in phase } \alpha}{\text{Mass of phase } \alpha} \quad (47) \]

Intrinsic mass density of a component

\[ \rho^k_\alpha = \frac{\text{Mass of component } k \text{ in phase } \alpha}{\text{Volume of component } k \text{ in phase } \alpha} \quad (48) \]

These fractional quantities are related to each other by the equation

\[ \rho_\alpha X^k_\alpha = \rho^k_\alpha C^k_\alpha \quad (49) \]
It is also obvious that we have the following constraints

\[ \sum_{k=1}^{N} X_{\alpha}^k = 1, \quad \sum_{k=1}^{N} C_{\alpha}^k = 1 \]  

(50)

We further introduce another fractional quantity, which is the molar fraction of a component

\[ x_{\alpha}^k = \frac{\text{Mole of component } k \text{ in phase } \alpha}{\text{Total moles of phase } \alpha} \]  

(51)

Then, the terms in the mass balance equations for an M-phase and N-component system can be rewritten as

\[ \xi^K = \phi \sum_{\alpha=1}^{M} \rho_{\text{mol, } \alpha} x_{\alpha}^K S_{\alpha} \]

\[ \psi^K = - \sum_{\alpha=1}^{M} \left( \rho_{\text{mol, } \alpha} x_{\alpha}^K q_{\alpha} + D_{\text{pm, } \alpha} \rho_{\text{mol, } \alpha} \nabla x_{\alpha}^K \right) \]

\[ f = f^K \]

For more details, see [Lauser et al., 2011; Bastian, 1999].

### 4.2. Mass Conservation Equations

For a control volume, we have the general balance equation:

\[ \frac{\partial \xi}{\partial t} + \nabla \cdot \psi = f \]  

(52)

where \( \xi \) is the storage term, \( \psi \) the flux term, and \( f \) the source term.

For each phase \( \alpha \), the above terms for mass conservation read:

\[ \xi = \phi \rho_{\text{mass, } \alpha} S_{\alpha} \]  

(53)

\[ \psi = -\rho_{\text{mass, } \alpha} q_{\alpha} \]  

(54)

\[ f = f_{\alpha} \]  

(55)

in which

\( \phi \) is the porosity.

\( S_{\alpha} \) is the phase saturation.

\( \rho_{\text{mass, } \alpha} \) is the mass density of the phase.

\( q_{\alpha} \) is the Darcy’s velocity.
4.3. Energy Equation

For non-isothermal flow, we will need to add the the energy conservation equation, which has the same form as (52). However, the storage, flux, and source terms are

\[ \phi \sum_{\alpha=1}^{M} \rho_{\text{mass,}\alpha} u_{\alpha} S_{\alpha} + (1 - \phi) \rho_{s} c_{s} T = \xi^{K} \]  

\[ - \sum_{\alpha=1}^{M} \left( \rho_{\text{mass,}\alpha} h_{\alpha}^{K} q_{\alpha} \right) + \sum_{K=1}^{N} \sum_{\alpha=1}^{M} \left( D_{pm,\alpha}^{K} \rho_{\text{mol,}\alpha} h_{\alpha}^{K} M^{K} \nabla x_{\alpha}^{K} \right) - \lambda_{pm} \nabla T = \psi^{K} \]  

\[ f = f^{h} \]  

in which

\( T \) is the temperature.  
\( \rho_{s} \) is the density of the solid phase.  
\( c_{s} \) is the specific heat capacity of the solid phase.  
\( h_{\alpha} \) is the specific enthalpy of phase \( \alpha \).  
\( \lambda_{pm} \) is the heat conduction coefficient.  
\( M^{K} \) is the atomic mass of component \( K \).

4.4. Complete System of Equations

Putting all the equations together, we want to solve the following system for fully coupled 2-phase, 2-component, non-isothermal, miscible flow:

\[ \frac{\partial}{\partial t} \left( \phi \sum_{\alpha=l,g} \rho_{\text{mol,}\alpha} x_{\alpha}^{K} S_{\alpha} \right) + \nabla \cdot \left( - \sum_{\alpha=l,g} \left( \rho_{\text{mol,}\alpha} x_{\alpha}^{K} q_{\alpha} + D_{pm,\alpha}^{K} \rho_{\text{mol,}\alpha} \nabla x_{\alpha}^{K} \right) \right) = f^{K} \]  

\[ \frac{\partial}{\partial t} \left( \phi \sum_{\alpha=l,g} \rho_{\text{mass,}\alpha} u_{\alpha} S_{\alpha} + (1 - \phi) \rho_{s} c_{s} T \right) + \nabla \cdot \left( - \sum_{\alpha=1}^{M} \left( \rho_{\text{mass,}\alpha} h_{\alpha}^{K} q_{\alpha} \right) + \sum_{K=1,2}^{M} \sum_{\alpha=l,g} \left( D_{pm,\alpha}^{K} \rho_{\text{mol,}\alpha} h_{\alpha}^{K} M^{K} \nabla x_{\alpha}^{K} \right) - \lambda_{pm} \nabla T \right) = f^{h} \]  

\[ q_{\alpha} = \frac{K_{\text{ref}}}{\mu_{\alpha}} K (\nabla P_{\alpha} - \rho_{\text{mass,}\alpha} g) \]  

\[ P_{c}(S_{l}) = P_{g} - P_{l}, \]  

\[ S_{l} + S_{g} = 1, \quad 0 \leq S_{\alpha} \leq 1 \]  

\[ \sum_{K=1,2}^{M} x_{\alpha}^{K} = 1, \quad 0 \leq x_{\alpha}^{K} \leq 1 \]  

4.5. Finite Volume Method for Extended System

We discretize the balance equations using cell-centered finite volume method. For simplicity, we assume a uniform grid. For each cell \( i \), integrate the mass conservation equations and
invoke the divergence theorem gives

\[ \frac{\partial}{\partial t} \int_{C_i} \xi + \sum_{j \in \eta} \int_{\gamma_{ij}} \psi \cdot n = \int_{C_i} f \]

We introduce some approximation of the volume integrals

\[ \bar{\xi} = \frac{1}{V_{C_i}} \int_{C_i} \xi, \quad F = \frac{1}{V_{C_i}} \int_{C_i} f \]

For the surface integrals, using two-point flux approximation (TPFA)

\[ \int_{\gamma_{ij}} \psi \cdot n = \gamma_{ij} (\rho_{\text{mol},a} x_\alpha^K K_{i+j+1/2} (\omega_i - \omega_j)) \]

\[ \omega_i = p_i - (\rho_{\text{mass}})_{i+1/2} g \]

The index \( ij + 1/2 \) signifies an appropriate averaging of properties at the interface between cell \( i \) and \( j \).

Discretize in time using backward Euler method, we have a fully discrete system of \( N \) nonlinear equations.

\[ \mathbf{H}(x) = (\bar{\xi}(x))_{i}^{n+1} - (\xi(x))_{i}^{n} - \frac{\Delta t}{V_{C_i}} \sum_{j \in \eta} \gamma_{ij} (\rho_{\text{mol},a} x_\alpha^K K_{i+j+1/2} (\omega_i^{n+1} - \omega_j^{n+1})) - F^{n+1} = 0 \]

where \( x = \{p_1, X_1^1, \ldots, X_1^N, S_2, \ldots, S_M, T\} \) is the set of \( N + M + 1 \) primary variables.

The choice of \( x \) is not unique, however. For example, it is convenient to choose \( x = \{p_1, f g^1, \ldots, f g^N, S_2, \ldots, S_M, T\} \) in which \( f g^K \) are the component fugacities defined by

\[ f g^K = \Phi^K_\alpha X^K_\alpha p_\alpha, \quad \text{for} \ K \in \{1, 2, \ldots, N\} \]

\( \Phi^K_\alpha \) is the fugacity coefficient. As noted in [Lauser et al., 2011], the intuition to choose fugacities instead of molar fractions is that they allows us to describe both the gas and liquid phases through equation (66).

### 4.6. Nonlinear Complementarity Constraints

#### 4.6.1. General Formulation

In its simplest form, a nonlinear complementarity constraint (NCP) problem can be written as follows

\[ x \geq 0 \] (67)

\[ f(x) \geq 0 \] (68)

\[ x^T f(x) = 0 \] (69)
where $x \in \mathbb{R}^n$, and $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a smooth function. A slightly more general form of equation (69) reads

$$g(x)^T f(x) = 0$$

where $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is another smooth function. The complementarity conditions can be expressed in an equivalent form via complementarity function (C-function). For example, consider

$$\phi : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$$

$$\phi(x, y) = \min(x, y)$$

$$\phi(x, y) = \sqrt{x^2 + y^2} - x - y$$

$$\phi(x, y) = -xy + \min^2(0, x) + \min^2(0, y)$$

These are C-functions in the sense that

$$\phi(x, y) = 0 \iff x \geq 0, \ y \geq 0, \ x^T y = 0$$

Along with the discretized balance equations, we still need another $M$ equations to complete the system.

### 4.6.2. Multiphase Flow with NCP

Recall the constraints for molar fractions

$$\sum_{K=1}^{N} X^K_{\alpha} \leq 1, \quad \sum_{K=1}^{N} X^K_{\alpha} = 1 \text{ if phase } \alpha \text{ is present.}$$

Combining with the saturation constraints, we get the complementarity conditions

$$1 - \sum_{K=1}^{N} X^K_{\alpha} \geq 0, \quad S_{\alpha} \geq 0, \quad S_{\alpha}(1 - \sum_{K=1}^{N} X^K_{\alpha}) = 0.$$  

The nonlinear complementarity constraints of saturation and molar fractions can be written in compact form as

$$\min(S_{\alpha}, 1 - \sum_{K=1}^{N} X^K_{\alpha}) = 0$$

Combining with the discretized mass balance equations, we have the full system for each time step:

$$\mathbf{R}(x) = \begin{cases} \mathbf{H}(x) = 0 \quad &\text{(from the PDE)} \\ \mathbf{\Psi}(x) = \min(\mathbf{F}, \mathbf{G}) = 0 \quad &\text{(from the constraints)} \end{cases}$$

with $\mathbf{F}$, and $\mathbf{G}$ are discretized functions of $S_{\alpha}$ and $1 - \sum_{K=1}^{N} X^K_{\alpha}$ respectively.
4.7. Semi-smooth Newton Method

We want to solve the system

\[ \mathbf{R}(x) = \begin{pmatrix} \mathbf{H}(x) \\ \Psi(x) \end{pmatrix} = 0 \]

where \( \Psi \) differentiable almost everywhere, \( \mathbf{R}(x) \) is the residual function. The procedure for semi-smooth Newton method is similar to Newton method, except that we use subdifferential for where the function is non-differentiable. Since \( \Psi \) is not differentiable everywhere, we solve the system using a semi-smooth Newton method as in [Gharbia and Jaffré 2014].

**Algorithm:** Semi-smooth Newton Method

while \( k < \text{max}_\text{iter} \) and res > \( \epsilon \) do

1. Define the index sets \( A^k \) and \( I^k \):
   
   \[ A^k := \{ j : \mathbf{F}_j(x^k) \geq \mathbf{G}_j(x^k) \} \]
   \[ I^k := \{ j : \mathbf{F}_j(x^k) < \mathbf{G}_j(x^k) \} \]

2. Select an element \( J^k \in \partial \Psi(x^k) \) such that its \( j \)th line is equal to
   \[ \mathbf{F}_j'(x^k) \text{ if } j \in I^k, \mathbf{G}_j'(x^k) \text{ if } j \in A^k \]

3. Solve the system
   \[ \mathbf{H}'(x^k) \Delta x^k = -\mathbf{H}(x^k) \]
   \[ J^k \Delta x^k = -\Psi(x^k) \]

4. Update \( x^{k+1} = x^k + \Delta x^k \)

The general semi-smooth Newton method has superlinear convergence for semi-smooth functions, and quadratic for strongly semi-smooth functions. For definitions of semi-smooth and strongly semi-smooth, we refer to [?]. For a more complete treatment of semi-smooth Newton method with active set strategy, see [Hintermüller et al. 2002].

5. Implementation

5.1. Amanzi

5.1.1. Introduction

The simulator of our choice is Amanzi [ASCEM 2009], which is a part of the Advanced Simulation Capability for Environmental Management (ASCEM) program. This program is supported by the US Department of Energy (DOE) and its primary objective is to provide a suite of state-of-the-art tools and approach to meet the challenge of minimizing the environmental, safety, and health risks associated with legacy nuclear waste across the DOE complex. ASCEM toolsets are integrated in Akuna and include data management, model setup, visualization, sensitivity analysis, and uncertainty quantification. Amanzi is ASCEMs computational engine, providing a state-of-the-art simulator for flow and reactive transport.
Using Amanzi for this course project has many advantages. Amanzi, is a multi-lab open-source C++ code that is actively being developed and the team is interested in collaboration. It uses an object oriented design that mirrors the hierarchy of processes being modeled to provide both run time flexibility and significant extensibility for those interested in adding new models (process kernels). Amanzi provides parallel distributed memory infrastructure for unstructured and structured AMR meshes, advanced discretizations such as the Mimetic Finite Difference (MFD) method, and linear and nonlinear solvers. Where possible, Amanzi leverages existing open-source libraries such as Trilinos and HYPRE, rather than duplicating capability and effort. Amanzi provides a User Guide and a growing suite of verification and benchmark problems for testing.

5.1.2. Design and Features

As can be seen in figure (1), Amanzi features a “process-kernel tree”. Each leaf of a “process-kernel tree” is a process-kernel (PK), which represents a single differential equation. This makes it easy to describe physical models through a hierarchy of mathematical models (or partial differential equations). One can choose as many PKs as necessary for a model at run-time. This which allows for models with a wide range of complexity. This is possible through Directed Acyclic Graph (DAG), that allows dynamic plug-and-play PKs via self-registering factories and automated Multi-Process Couplers. The goal of this design is to ensure domain scientists can easily understand, extend, and develop PK implementations. For more details on Amanzi features and design philosophy, we refer to [Coon et al., 2014].

5.2. Pressure Equation

We implement the pressure equation as a process kernel (PK) in Amanzi. This means that we need to conform to the standard Amanzi interface, in order for this new feature to interact...
correctly with existing codes. The main methods for a typical PK is shown in the snippet below

```cpp
// New interface for a PK
virtual void Initialize();
virtual bool AdvanceStep(double t_old, double t_new);
virtual void CommitStep(double t_old, double t_new){};
virtual void CalculateDiagnostics(){};

// Main methods of this PK
void InitializeFields();
void InitializePressure();
void InitTimeInterval(Teuchos::ParameterList& ti_list);
void CommitState(const Teuchos::Ptr<State>& S);

// Time integration interface new_mpc, implemented in Pressure_PK_TI.cc
// computes the non-linear functional f = f(t,u,udot)
virtual void Functional(double t_old, double t_new,
    Teuchos::RCP<TreeVector> u_old,
    Teuchos::RCP<TreeVector> u_new,
    Teuchos::RCP<TreeVector> f);

// applies preconditioner to u and returns the result in Pu
virtual void ApplyPreconditioner(
    Teuchos::RCP<const TreeVector> u,
    Teuchos::RCP<TreeVector> Pu);

// updates the preconditioner
virtual void UpdatePreconditioner(double t,

Every PK is understood as a single discrete PDE that can be solved by a linear or nonlinear solver. In order to interact with the Multi-process Coordinator (MPC), a PK must have standard interface methods such as

- **Initialize** create the variables needed for this equation, the methods for space and time discretization, construct the operators, etc.

- **AdvanceStep** Advance the equation in time. For time-dependent problem, this entails a time discretization and integration. For steady-state problem, it simply solves the steady-state problem.

- **CommitStep** Write the solution back to the State for this time step.

- **CalculateDiagnostics** Calculate diagnostics to help understand whether the PK is doing what it is supposed to.

Beside the standard interface methods, each PK can also have its own local methods. Particularly in this Pressure_PK, these are
• InitializeFields
• InitializePressure
• InitTimeInterval
• CommitState

Other required methods for a PK are those needed by the linear and nonlinear solvers:
• Functional basically computes the residual of the system of algebraic equations.
• UpdatePreconditioner updates the preconditioner for solving the system.
• ApplyPreconditioner solves the the algebraic system of equations.

We have implemented all the above methods, and other necessary methods for processing XML input files for the pressure equation as well. More specifically, for the pressure equation, the main component is the diffusion operator, which is provided by an Amanzi class called OperatorDiffusion. However, customization is needed to adapt this class to our specific problem. The default diffusion operator in Amanzi uses upwinding for the relative permeability. In our problem, the relative permeability is replaced by the total mobility. Furthermore, the total mobility is computed using harmonic average. Thus, we have to loop through the faces, look for the cells adjacent to the faces, compute the relative permeability and total mobility for the cells, multiply by the absolute permeability tensor, and then take the harmonic average. This is done efficiently and already parallelized thanks to Amanzi infrastructure inside the method InitTimeInterval.

After discretization, we have a system of linear equations that needs to be solved. Unlike in Matlab, setting up and solving a linear system in highly parallelized packages with a wide range of preconditioner choices such as Trilinos or Hypre can be quite difficult. Amanzi provides easy access to these solvers and preconditioners, provided that we conform to Amanzi datastructures and interfaces. For the test problems, we use the algebraic multigrid preconditioner from the Hypre package, and the preconditioned conjugate gradient method from the Trilinos package.

5.3. Code Verification for Pressure Equation

We have the following unit test for the pressure equation.

\[-\nabla \cdot (\lambda \nabla P) = f \text{ on } \Omega = [0, 1] \times [0, 1]\]
\[P = P_d \text{ on } \partial \Omega\]

Example problem

\[\lambda = 1, \ P_d = 0\]
\[f = 5\pi^2 \sin (\pi x) \sin (2\pi y)\]
\[P = \sin (\pi x) \sin (2\pi y)\]
Compared to the analytic solution, the numerical solution looks correct (figure 2). In order to be certain, we perform a convergence test for the same problem. Since for the pressure equation, we use a center difference to compute the values on the faces from the cell values (see equation (44)), the errors should be proportional to $h^2$. This is exactly what is shown in figure 3.

Beside the above test, we also have various tests to verify the correctness of important components of the Pressure_PK such as tests for Functional and UpdatePreconditioner methods. All the tests are done using the UnitTest++ framework, which is supported by CMake build system.
5.4. Saturation Equation

Similar to the pressure equation, we implement the saturation equation as a separate PK in Amanzi. The Saturation_PK has been implemented completely. It has exactly the same interface methods mentioned above in the Pressure_PK. Only the implementation for each method is different. The main component of the saturation equation is an advection operator, as opposed to the diffusion operator in the pressure equation. Again, we have to modify the default operator for our problem. We implemented a separate class for fractional flow, which provide a method for computing fractional flow in cells. Then the cell values are mapped on the the faces using upwinding. Unlike the Pressure_PK, since the saturation equation is a nonlinear time-dependent problem, we have to implement a method for time integration. For each time step in Saturation_PK, the discretization results in a system of nonlinear equations which is solved using Newton method. Again, Amanzi interface provides access to the nonlinear solvers. The code for Saturation_PK builds and runs successfully. A verification test is presented in the next section.

5.5. Code Verification for Saturation Equation

We propose the following unit tests for verifying the implementation of the Saturation_PK.

- Unit test for 1D advection.
  \[ u_t + f(u)_x = 0 \]

- Buckley-Leverett problem for one-dimensional 2-phase, 2-component, immiscible, incompressible flow.
  \[ S_t = U(S)S_x \]
  \[ U(S) = \frac{Q}{\phi A} \frac{df}{dS} \]

where \( S \) is the saturation, \( Q \) is the flux, \( A \) is the cross section area, \( \phi \) is the porosity, and \( f \) is the fractional flow. The Buckley-Leverett equation is a nonlinear hyperbolic equation with non-convex flux \( f \).

For simplicity, we can test the code for unit flux \( Q = 1 \), unit cross-sectional area \( A = 1 \), unit porosity \( \phi = 1 \), and the flux

\[ f(S) = \frac{S^2}{S^2 + (1 - S)^2} \quad \text{(70)} \]

The initial condition is as follows

\[ S_0(t = 0) = \begin{cases} 
1 & \text{for } x \leq 2 \\
0 & \text{for } x > 2 
\end{cases} \quad \text{(71)} \]

For boundary condition, we have Dirichlet boundary conditions \( U_L(x = 0) = 1 \) and \( U_R(x = 10) = 0 \). The analytic and numerical solutions to this problem is plotted in the figure below.
Figure 4: Solutions of the Buckley-Leverett equation for different time steps using first order upwind scheme: $t = 0.2$ (blue), $t = 0.4$ (green), and $t = 0.6$ (red).

From figure (4), it is obvious that the implicit upwind scheme suffers from numerical diffusion, which smears out the sharp front observed in the analytic solution. Although an explicit upwind scheme can avoid this problem, we choose the implicit scheme for stability reason. In an explicit scheme, one has to prescribe a CFL condition to ensure that the solution does not blow up. Furthermore, this condition can sometimes make the timesteps very small, which affects the overall performance of the method.

5.6. Coupling Pressure and Saturation

We have coupled the pressure and saturation equations by solving them sequentially. First, providing an initial saturation, we can calculate the total mobility, which makes the pressure equation becomes a linear elliptic equation with constant coefficient. Once we solve for pressure, we compute the velocity from the pressure, which is subsequently substituted into the saturation equation to calculate the fluxes on the faces by upwinding. Then we solve the saturation equation with an initial saturation and some time step $\Delta t$.

6. Validation

In general, since it is difficult to obtain experimental data for flow in porous media, and most of the data is not publicly available, we cannot validate the code. However, we can compare the results of our code to those published in the literature with other codes. In particular, we might want to benchmark our code with the Open Porous Media - eWoms \cite{Lauser2013} package. Besides, we can also compare the results of the NCP approach to those of PVS near the end of the project if time permits.
7. Project Schedule and Milestones

We plan to follow these general incremental steps in our development of the new process kernel (PK):

Phase I

- October
  - Gain confidence working with Amanzi coding standard, build tools, interfacing with other libraries, etc. ✓
  - Get this new PK to interface correctly with both the multi-process coordinator (MPC) and input files. ✓
  - Start with building a simple solver for the pressure equation. ✓

- November
  - Implement a simulator for incompressible 2-phase flow. ✓
  - Fully coupled approach for incompressible 2-phase flow. ✗
  - Unit tests for pressure equation. ✓
  - Unit tests for saturation equation. ✗

- December
  - Add nonlinear complementarity constraints for phase transitions. (In progress)
  - Prepare mid-year report and presentation. ✓

Phase II

- January: Develop active sets and semi-smooth Newton method.

- February: Add miscibility effect.

- March: Incorporate energy equation for thermal effect.

- April
  - Collect the unit tests and make a test suite.
  - Benchmark with existing codes or pursue realistic problems, such as desiccation if time permits.

- May: Prepare final report and presentation.
8. **Deliverables**

At the end of the course, I will be able to deliver the following:

- An implementation of the NCP approach for a two-phase, two-component, non-isothermal, miscible flow as a PK in Amanzi.
- A semi-smooth Newton solver for the resulting system.
- Unit tests for verification.
- Results of validation and testing.
- Documentation for all the codes.
- User guide for multiphase flow PK.
- Weekly reports.
- Project proposal and presentation.
- Mid-year presentation and report.
- Final presentation and report.
References


