Spring 1-1-2017

Numerical Simulations of Two-Phase Flow in Rigid Porous Media

Rebecca Scheetz
University of Colorado at Boulder, resc5112@colorado.edu

Numerical Simulations of Two-Phase Flow
in Rigid Porous Media

by

REBECCA LOUISE SCHEETZ

B.S., University of Colorado, 2017

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirement for the degree of

Master of Science

Civil Engineering

2017

This page intentionally left blank.

This thesis entitled:

Numerical Simulations of Two-Phase Flow in Rigid Porous Media

written by Rebecca Louise Scheetz

has been approved for the Civil Environmental and Architectural Engineering Department

____

Yida Zhang

____

Harihar Rajaram

____

Roseanna Neupauer

Date_____________

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above-mentioned discipline.

This page intentionally left blank.

Scheetz, Rebecca Louise (M.S., Civil Engineering)

Numerical Simulations of Two-Phase Flow in Rigid Porous Media

Thesis directed by Associate Professor Yida Zhang

This study concerns the development of a series of finite difference codes for solving one-dimensional two-phase flow problems. The ability to predict fluid movement in saturated and unsaturated soils is an important problem in many branches of science and engineering, including soil science, agricultural engineering, environmental engineering and groundwater hydrology. The research performed for this thesis is motivated by three main areas of study: blast densification in saturated sand, enhanced oil recovery and geothermal energy harvesting. This study models imbibition fronts in rigid porous skeleton resulting from varying boundary and initial conditions by solving governing equations for two-phase flow using the Picard and fourth-order Runge Kutta methods with finite difference spatial approximations. The numerical results were validated using experimental data from Melean et al. (2003) and Touma and Vauclin (1986). Results indicate that the numerical approximations yield accurate and practical estimations of the infiltration variables of interest.

This page intentionally left blank.

ACKNOWLEDGEMENTS

The completion of this project would not have been possible without the support and guidance from many people. I owe a great deal of thanks to my advisor, Yida Zhang. He has supported and guided me through this process, and I am grateful for his time, teaching, patience and commitment. I also greatly appreciate the advice, construction comments and help I received from Professor Harihar Rajaram. It is largely due to Yida and Hari's enthusiasm and persistence that I was able to finish this project, and I could not have asked for a better group of engineers or people with whom to collaborate.

I would also like to acknowledge my colleagues who contributed valuable insight and suggestions for this paper, including my Kristen Hess, Sarah Welsh-Huggins, Samantha Grey, Polly Murray, and my research group among others.

I also owe a special thanks to my boyfriend Tom Mallon and my sister Kara Scheetz for their unwavering encouragement and support. Supai, thank you for your company during those long days and late nights, and for your ever trustful and reassuring eyes.

Finally, I would like to thank my parents who have always inspired and supported me.

This page intentionally left blank.

CONTENTS

LIST OF TABLES

This page intentionally left blank.

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Research Objectives

1.1.1 Numerical Simulators

This thesis focuses on developing a series of finite difference codes for solving one-dimensional two-phase flow problems. The research is motivated by three main areas of study: blast densification in saturated sand, enhanced oil recovery, and geothermal energy harvesting.

The goal of this project is subdivided as follows: (a) to develop and validate a numerical solver capable of simulating 1-D two phase incompressible flow problems with a constant flux boundary at the inflow; (b) to develop and validate a numerical solver capable of simulating 1-D two phase incompressible flow problems with constant pressure boundaries, and (c) to develop and validate a numerical solver capable of simulating 1-D two phase compressible flow problems with varying boundary conditions.

1.1.2 Supporting Experimental Procedures

This research is performed in support of a larger project investigating the dynamics of soil densification and multiphase flow under blast loading. The models produced in this study will support the interpretation of experimental data to be collected in the 2017-2018 academic year. The goal of these experiments is to design and conduct a series of one-dimensional experiments on high-rate gas injection (i.e. blasting) in saturated sand to validate the governing equations and computational results. A rudimentary schematic of the test setups under development is shown in Figure 1.1.

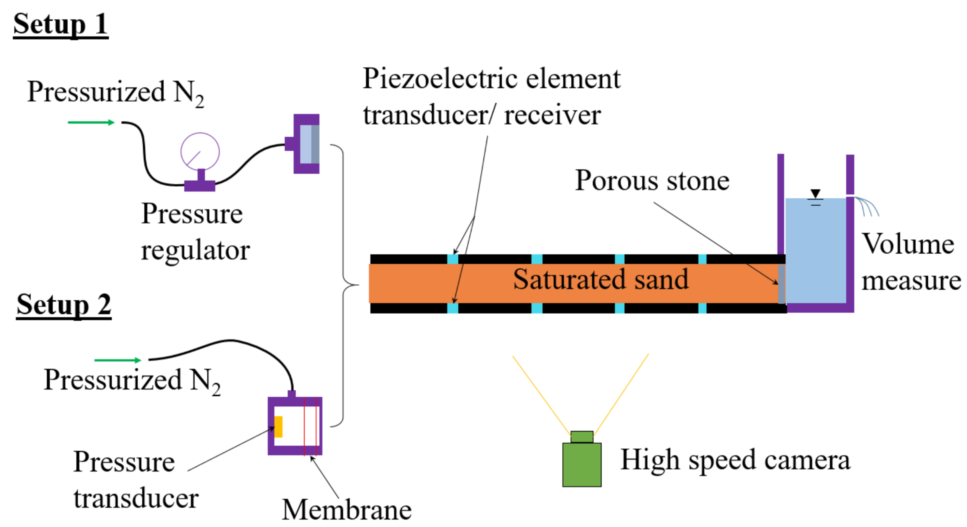


Figure 1.1 Experimental Setups

The set up consists of a thick-walled acrylic tube filled with clean sand (e.g. the Ottawa 20/40 sand) saturated by de-aired water and capped by porous stones. Nitrogen gas will be the gas agent in the injection or explosion study so that the assumption of zero mass exchange between phases remains valid. Along the tube, multiple compressive piezoelectric elements will be installed to measure the pressure wave (p-wave) velocity at different cross sections. This data will be used to track the propagation of the gas front during the test.

1.2 Scope of the Study

The research completed within the scope of the study is organized into three different objectives to accomplish the threefold purpose outlined in section 1.1.

1.2.1 Objective 1

Objective I, discussed in Chapter 3, addresses part (a) of the purpose by examining the critical aspects of a two-phase incompressible flow system subject to a constant flux boundary at the inflow. The system's governing equations are summarized and then organized into a numerical simulator based on the modified Picard method. The solver is validated using the oil-water invasion data reported by Melean et al. (2003).

1.2.2 Objective II

Objective II, discussed in Chapter 4, addresses part (b) of the purpose by examining the critical aspects of a two-phase incompressible flow system subject to constant pressure boundaries. Again, the governing equations are summarized and then organized into a coupled numerical solver based on the modified Picard method. This numerical solver is validated by a comprehensive examination of the simulated results.

1.2.3 Objective III

Objective III, discussed in Chapter 5, addresses part (c) of the purpose by examining the critical aspects a two-phase compressible flow system. Once again, the governing equations are summarized, and then implemented into two separate numerical simulators based on the fourth-order Runge-Kutta method and the modified Picard method. Finally, these final simulators are validated using the experimental data by Touma and Vauclin (1986).

1.3 Arrangement of the Thesis

The thesis is organized into six chapters, as follows. Chapter 1 includes the purpose and outlines the scope of the study. Chapter 2 reviews the problem statement, discusses relevant background information and motivates the development of the three solvers. This chapter also highlights the main approaches used by researchers in the past to model two phase flow problems, as well as the approaches used in this study to model two-phase flow problems and the advantages and disadvantages of each. Chapter 3 describes in detail the development and validation of the numerical solver capable of simulating a 1-D two-phase incompressible flow system subject to a constant flux boundary at the inflow. This chapter includes a parametric study and discussion analyzing the solver’s response to varying several model parameters. This solver is validated by fitting the numerical solutions to experimental data given by Melean et al.

in 2003. Chapter 4 describes in detail the development and validation of the coupled numerical solver capable of simulating a 1-D two phase incompressible flow system subject to constant pressure boundaries. Chapter 5 describes in detail the development and validation of the numerical solver capable of simulating a 1-D two-phase compressible flow system subject to varying boundary conditions based on Binning and Celia's 1992 formulation. To validate these results, the numerical solver is compared to the experimental solutions given by Touma and Vauclin in 1986. The computational performance of the Backward-Euler Picard iteration scheme is also compared to an explicit RK4 formulation. Chapter 6 summarizes the accomplishment of the threefold purpose for the study and discusses future directions of the research.

CHAPTER 2
BACKGROUND AND MOTIVATION

2.1 Current Research Direction

The concurrent, unsteady flow of two fluids in porous media has long been of interest in Earth science. The ability to predict fluid movement in saturated and unsaturated soils is an important problem in many branches of science and engineering, including soil science, agricultural engineering, environmental engineering and groundwater hydrology (Celia, Bouloutas, & Zarba, 1990). Specifically, the dynamics of the simultaneous flow of two fluids is important in numerous engineering problems such as the extraction of hydrocarbon in deep geological reservoirs, geothermal energy harvesting, enhanced oil recoveries, hydraulic fracking, as well as ground improvement using gas products in geotechnical engineering. Depending on the field of research and its application, researchers continue to focus primarily on the behavior of both water-oil and water-air systems.

2.2 Motivation

Researchers at the University of Colorado Boulder wish to model the imbibition fronts in a rigid porous media resulting from varying boundary conditions. These predictions will eventually be used as a comparison point for novel experimental data to be collected in the next stage of the project.

2.3 Problem Statement

The goal of this project is to produce accurate and practical estimations of the infiltration variables of interest during imbibition processes resulting from varying boundary conditions.

2.3.1 Idealized Examples of the Invasion of Wetting and Nonwetting Fluids

Suppose a rigid porous skeleton with some initial porosity φ is initially fully saturated by a nonwetting fluid, like oil. At time $t_0 = 0$ the skeleton is suddenly subjected to the invasion of a wetting fluid, like water. The saturation profile within the porous skeleton will resemble Figure 2.1 below at times t_1 and t_2, assuming $t_0 < t_1 < t_2$.

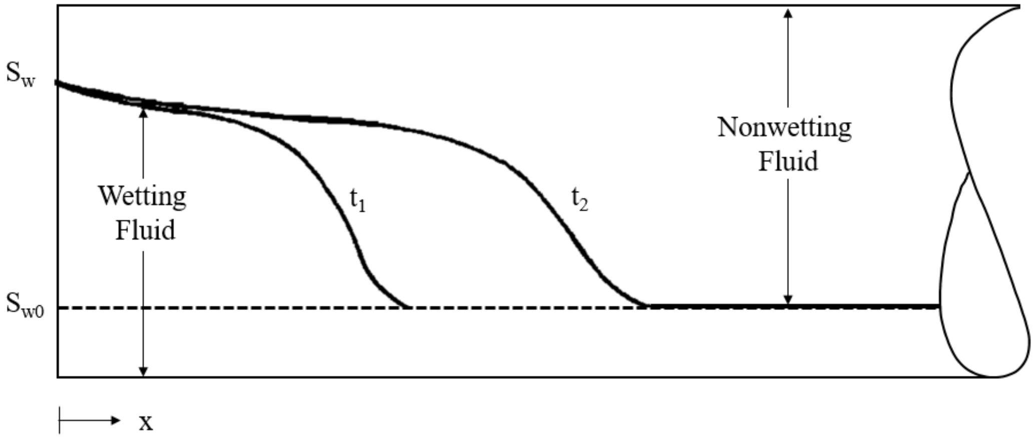


Figure 2.1 Schematic of the invasion of a nonwetting fluid.

As the wetting front moves through the soil skeleton, the nonwetting fluid is simultaneously displaced. This infiltration front moving through the porous media may also be called an imbibition front.

Similarly, one can also imagine the invasion of a nonwetting fluid as seen in Figure 2.2.

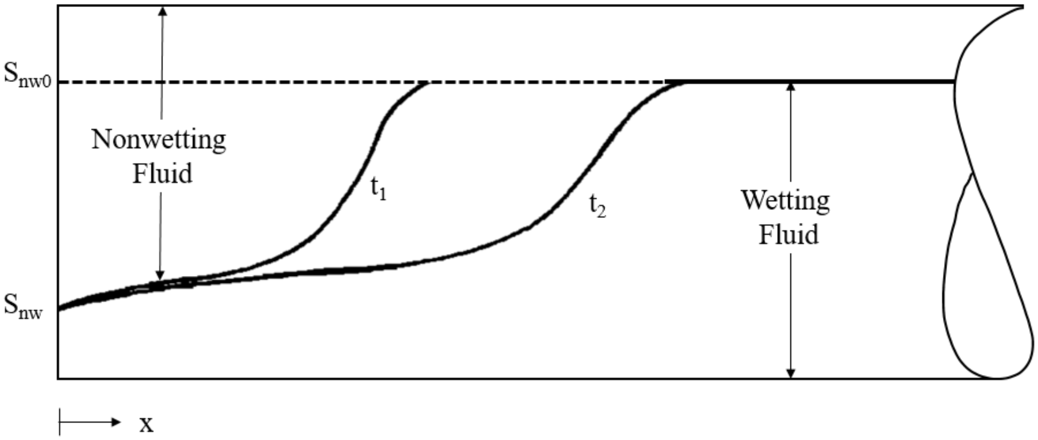


Figure 2.2 Schematic of the invasion of a wetting fluid.

2.3.2 Boundary Conditions

The same numerical simulators can be used to model the invasion of wetting and nonwetting fluids by imposing different initial and boundary conditions.

While there are a few different ways to enforce both wetting and nonwetting invasions, for this project, the only methods considered were varying the flux at the inflow boundary and varying the pressure at the inflow boundary. For example, one can enforce the invasion of a wetting fluid by imposing the flux of the wetting fluid at the inflow, or by enforcing a low suction pressure at the inflow relative to the rest of the porous medium. Similarly, the invasion of a nonwetting fluid can be enforced by varying pressure and flux conditions at the inflow boundary.

In most of the models developed for this research, the outflow boundary is considered as a constant pressure boundary to resemble the typical open-end condition in most experimental studies.

2.4 Governing Principles of Two-Phase Flow Problems

2.4.1 Darcy's Law

Darcy's Law is perhaps the most common model used to describe fluid flow through a saturated porous medium. For a 1-D system, Darcy's Law can be written in terms of pressure as follows:

$$Q = -\frac{\kappa}{\mu}\frac{dP}{dx} \tag{2.1}$$

where Q is the total flow per unit area through the porous medium, κ is the intrinsic permeability of the porous medium, μ is the dynamic viscosity of the fluid and dP/dx is the pressure gradient, or potential, which is defined as the change of pressure over the length of interest (FracFocus, 2017).

If gravity is considered and the 1-D column is placed in a vertical configuration, Darcy's Law should be modified to incorporate a gravity term:

$$Q = \frac{-\kappa}{\mu}\left(\frac{\partial P}{\partial x} - \rho G\right) \tag{2.2}$$

where Q is the rate of flow per unit area in the vertical direction, ρ is the density of the fluid and G is the gravitational constant.

When faced with problems considering the flow of two immiscible fluids flowing simultaneously, it is necessary to distinguish between the flow of the wetting and nonwetting fluids. A wetting fluid is defined as a fluid capable of maintaining a surface contact angle less than 90° with the solid surface. A nonwetting fluid is defined as any fluid that equilibrates with a solid surface with a contact angle greater than 90°. As an example, water can be considered as a wetting fluid and oil or air can be considered as nonwetting fluids. See Figure 2.3 for reference.

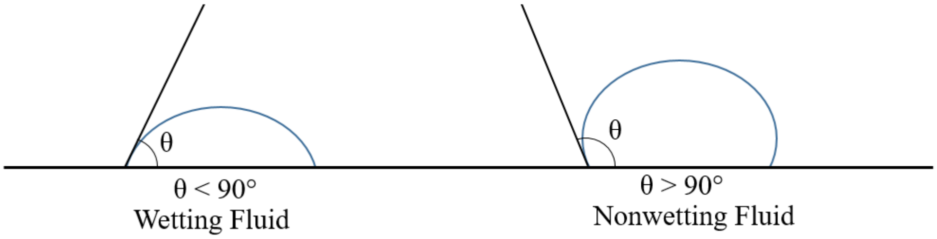


Figure 2.3 Distinguishing between wetting and nonwetting fluids based on their contact angle with the solid surface.

The total flow in a two-phase flow system can be separated into the flow of the wetting and nonwetting fluids. In a saturated two-phase flow system, the total flow and degree of saturation can be defined as follows:

$$Q = q_w + q_{nw} \tag{2.3}$$

$$S_w + S_{nw} = 1 \tag{2.4}$$

where Q is the total flow within the porous medium, q_w and q_{nw} are the flow of the wetting and nonwetting fluids respectively, and S_w and S_{nw} are the degrees of saturation of the wetting and nonwetting fluids respectively. Note that for a two-phase system, the system is always completely saturated by the two phases (see equation (2.4) for reference).

For incompressible fluid phases, the total flux Q is a constant throughout the domain and becomes only a function of time, which is controlled by the boundary condition. Note that this conclusion is only valid for 1-D scenarios, and that for 2-D and 3-D systems, only a much weaker conclusion, $\nabla \cdot \mathbf{Q} = 0$, can be drawn from assuming incompressible fluid phases

In the case of two-phase flow, Darcy's Law for fractional flows of wetting and nonwetting fluids must be modified by a relative permeability term as follows:

$$q_m = \frac{-\kappa k_{rm}(S_m)}{\mu_m}\left(\frac{\partial P_m}{\partial x} - \rho_m G\right) \rightarrow \begin{cases} q_w = \dfrac{-\kappa k_{rw}(S_w)}{\mu_w}\left(\dfrac{\partial P_w}{\partial x} - \rho_w G\right) \\ q_{nw} = \dfrac{-\kappa k_{rnw}(S_{nw})}{\mu_{nw}}\left(\dfrac{\partial P_{nw}}{\partial x} - \rho_{nw} G\right) \end{cases} \tag{2.5}$$

where the subscript *m* can be substituted as subscripts *w* and *nw* denote the wetting and nonwetting fluids, respectively. In the above equations, the variables P_w and P_{nw} are defined as the pressures of the wetting and nonwetting fluids respectively. The relative permeability, k_r, is a function of the degree of saturation and is discussed further in section 2.4.3.

The difference in the pressure of the nonwetting and wetting fluids is defined as the capillary pressure, or suction pressure. Often, it is this difference in pressure that drives the flow of the wetting or nonwetting fluids, the so-called capillary drive. The capillary pressure, P_c is a function of the degree of saturation of the wetting fluid as follows:

$$P_{nw} - P_w = P_c = \Pi(S_w) \tag{2.6}$$

where Π*(S*$_w$*)* represents a relationship between the capillary pressure and the saturation of the wetting fluid, known as the soil water retention curve, which is discussed further in section 2.4.2.

2.4.2 Soil Water Retention Curve

The soil water retention curve is a relationship between the water content and the water potential that can be used to estimate various parameters describing the behavior of unsaturated soil. There are several models that can be used to obtain an estimation of the soil water retention curve. Each model results in a slightly different estimation, and unfortunately no unified model exists to date which can be used for all soil types (Rahimi, Rahardjo, & Leong, 1997). Several researchers have proposed different models, including Brooks & Corey in 1964, Brutsaert in 1966, van Genuchten in 1980 and Fredlund et al. in 1994 (Rahimi, Rahardjo, & Leong, 1997).

For this research, the van Genuchten model was used to predict the degree of saturation based on the capillary pressure at that point. The relationship between the degree of saturation and capillary pressure as defined by van Genuchten is given as follows:

$$S_e = \frac{S_w - S_r}{1 - S_r} = \left(\frac{1}{(1 + P_c / p_0)^N} \right)^M \tag{2.7}$$

where S_e is the effective degree of saturation. Solving for the degree of saturation of the wetting fluid in equation (2.7) gives the following equation:

$$S_w = \left(\frac{1}{(1 + P_c / p_0)^N} \right)^M (1 - S_r) + S_r \tag{2.8}$$

where S_r is the residual saturation, p_0 defines the suction air entry value, N is a model parameter that is related to the pore size distribution of the porous medium, and M is defined as follows:

$$M = 1 - 1 / N \tag{2.9}$$

In general, larger *N* values reflect a wider pore size distribution, and larger p_0 values correspond to smaller average pore sizes. For example, the value of *N* is generally larger for sand-like soils than clay-like soils, and p_0 values for clays are generally larger for clay-like soils than sand-like soils. A van Genuchten model representation of the proposed soil water retention curves can be seen in Figure 2.4.

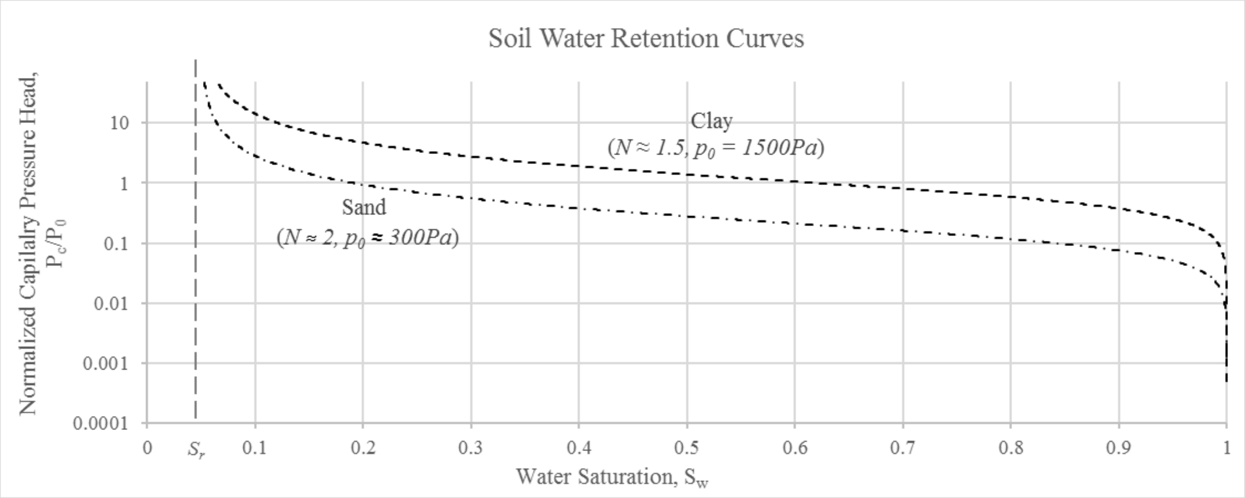


Figure 2.4 Example soil water retention curves for sand and clay based on the van Genuchten model. In this figure, the residual degree of saturation S_r is 5%.

The highly nonlinear nature of the capillary drive is the primary source of many mathematical difficulties in deriving close-form solutions to two-phase flow problems (McWhorter & Sunada, 1990). In the past, realistic behavior of the capillary drive has been captured through numerical solutions (Fokas & Yortsos, 1982), and only limited analytical solutions with simplifying assumptions have been obtained (Buckley & Leverett, 1942), (McWhorter & Sunada, 1990).

2.4.3 *Relative Permeability*

The relative permeability of a fluid is a function of its corresponding volume fraction. One may think of the relative permeability as a type of scaling factor as follows:

$$\kappa_{effective} = k_r \cdot \kappa \tag{2.10}$$

At intermediate wetting fluid contents, the effective intrinsic permeability of the wetting fluid will assume some intermediate value between zero and κ (Morel-Seytoux, 1973), scaled by the value of the relative permeability, k_r, which lies between zero and one. For example, if a wetting fluid flows alone in a porous medium and occupies the entire pore space, the effective permeability should be identical to the intrinsic permeability of the porous medium κ (i.e. $k_{rw} = 1$). Similarly, when the porous medium is completely occupied by a nonwetting fluid, the pathways for the transporting wetting fluid are all lost and the effective permeability of the wetting fluid approaches to zero (i.e. $k_{rw} = 0$).

For this study, the relative permeabilities of the wetting and nonwetting fluids are defined as a function of the saturation of the wetting fluid based on the Mualem's 1976 model as follows:

$$\begin{cases} k_{rw} = \sqrt{S_w}(1-(1-S_w{}^{1/M})^M)^2 \\ k_{rnw} = \sqrt{1-S_w}(1-S_w{}^{1/M})^{2M} \end{cases} \tag{2.11}$$

where M is a model parameter between zero and one that varies based on the type of soil making up the porous media, as given in equation (2.9). Figure 2.5 shows how the relative permeabilities of wetting and nonwetting fluids vary as a function of water saturation with varying M (or N) values based on Mualem's 1976 model.

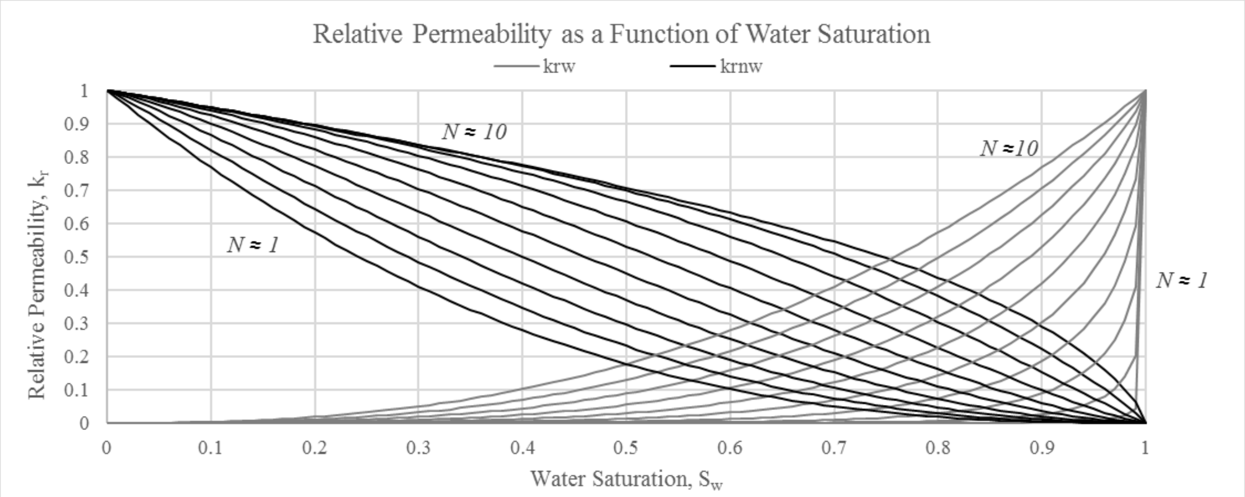


Figure 2.5 Relative permeability of wetting and nonwetting fluids as a function of the water saturation based on Maulem's 1976 model.

2.4.4 Richard's Equation

In studying vadose zone hydrology, the classical Richard's equation is often invoked to model the transport of fluids (Hillel, 1980). Richard's equation is a very specific form of the mass balance equation where the compressibility of the fluid is neglected, solid deformation is ignored and the air pressure is assumed to be constant. These assumptions are valid for this research in the specific case of constant P_{nw} conditions. Even though the governing equations for the more general case of two-phase flow are not strictly the same as "Richard's Equation," they do bear prominent similarities, and it is for this reason that the following discussion is included.

Richard's equation governing unsaturated flow can be written in several different forms, with either moisture content, θ, or pressure head, h, acting as the dependent variable. Three standard forms exist and can be identified as the "h-based" form, the "θ-based" form and the "mixed" form, where in the mixed both moisture content and pressure head are dependent variables (Celia, Bouloutas, & Zarba, 1990). These equations are given as follows:

h-based:

$$C(h)\frac{\partial h}{\partial t} - \nabla \cdot K(h)\nabla h - \frac{\partial K}{\partial x} = 0 \tag{2.12}$$

θ-based:

$$\frac{\partial \theta}{\partial t} - \nabla \cdot D(\theta)\nabla \theta - \frac{\partial K}{\partial x} = 0 \tag{2.13}$$

Mixed:

$$\frac{\partial \theta}{\partial t} - \nabla \cdot K(h)\nabla h - \frac{\partial K}{\partial x} = 0 \tag{2.14}$$

where *C(h)* ≡ *dθ/dh* is the soil-water retention relationship discussed in section 2.4.2, *K(h)* is the unsaturated hydraulic conductivity, *D(h)* ≡ *K(θ)/C(θ)* is the unsaturated diffusivity and *x* denotes the length of the sample (Celia, Bouloutas, & Zarba, 1990). Before the 1990s, many researchers in hydrology used either the h-based or *θ*-based formulation to approximate the numerical solutions (Celia, Bouloutas, & Zarba, 1990). However, beginning in the 1990s, researchers began to realize that *h*-based solutions to Richard's equation tend to exhibit mass balance problems (Celia, Bouloutas, & Zarba, 1990). Researcher also observed that the *θ*-based formulation, while able to conserve mass, can be problematic because the diffusivity *D(θ)* increases to infinity when the degree of saturation of the wetting fluid is near zero, and that the water content *θ* is not constant across interfaces between different mediums. The mixed formulation does not exhibit any of these problems, and for this reason, the mixed-form formulation of the mass balance equation was used in this project.

For this project, the mass balance equation is written with the pressure, *P*, as the independent variable and the degree of saturation, S_w, as the dependent variable. An intermediate step is to convert the water content and pressure head to degree of saturation and pressure using the following relationships:

$$\theta_m = \varphi S_m \tag{2.15}$$

$$P_m = \rho_m g h \tag{2.16}$$

where φ is the porosity of the soil.

2.4.5 The Mass Balance Equation

The most general form of mass balance equation for phase *m* can be written as:

$$\frac{\partial}{\partial t}\left[\rho_m \varphi S_m\right] + \frac{\partial}{\partial x}\left[\rho_m q_m\right] = 0 \tag{2.17}$$

By substituting Darcy's Law as in equation (2.5), the mass balance equation becomes:

$$\frac{\partial}{\partial t}\left[\rho_m \varphi S_m\right] + \frac{\partial}{\partial x}\left[\rho_m \left(\frac{-\kappa k_{rm}(S_m)}{\mu_m}\left(\frac{\partial P_m}{\partial x} - \rho_m G\right)\right)\right] = 0 \tag{2.18}$$

This version of the mass balance equation governs the vertical flow of compressible fluids in a deformable solid skeleton. However, depending on the characteristics of the system being analyzed, the mass balance equation may be simplified. The version of the mass balance equation required for the numerical solvers is derived at the beginning of each chapter.

2.5 Model Development using Numerical Methods

Solutions to ordinary and partial differential equations can be achieved via mathematical techniques (often referred to as analytical methods) such as method of separation of variables and transform methods. However, for practical boundary conditions or for highly nonlinear systems, the closed form solution often cannot be obtained, and thus numerical methods are necessarily invoked to approximate their solutions. The nonlinear nature of Richard's equation, specifically the soil water retention relationship and the relative permeabilities of the fluids, is what has steered researchers to apply numerical methods to solve the unsaturated flow equation (Celia, Bouloutas, & Zarba, 1990). However, even with numerical methods, Richard's equation can display singularities under certain formulation and difficult to solve in many cases.

2.5.1 Finite Difference Method

Typically, the spatial domain is approximated using the finite difference methods or the finite element method coupled with a simple one-step Euler time-marching algorithm. Research has shown that, when solving Richard's equation for unsaturated flow, finite element methods are generally inferior to finite difference methods, as finite element methods may result in oscillatory solutions even when mass is conserved (Celia, Bouloutas, & Zarba, 1990). For this reason, a finite difference approximation with the implicit Euler backward algorithm was applied to approximate the spatial domain in all numerical solvers developed for this project.

2.5.2 Implicit Modified Picard Method

Throughout the development of numerical solutions for unsaturated flow, the implicit Picard scheme has attracted the most attention due to its relatively straightforward implementation procedure (i.e. no need to derive Jacobian matrix as compared to the Newton-Raphson method), and it has the advantage of being unconditionally stable (i.e. no limitation on the size of time step as compare to explicit methods).

For three of the four solvers developed for this project, the specific method used to approximate the solution was developed by Celia and coworkers in 1987 (Celia, Ahuja, & Pinder, 1987), a method that they referred to as an implicit "modified Picard" method. This method, in contrast to the original Picard method, represents the time derivative of water content (or saturation) such that a small change in this value can be represented in terms of pressure head (or pressure), the so called "mixed form." This method demonstrates excellent mass balance in the numerical solutions with both finite difference and finite element approximations in space.

2.5.3 Explicit Fourth-Order Runge-Kutta Method

The final solver developed for this project is based on the explicit fourth-order Runge-Kutta method. In general, explicit methods tend to be much easier to implement, but have a limited maximum time step and thus lead to more computationally expensive solutions. The explicit fourth-order Runge-Kutta method was selected to provide a comparison point for the implicit solver of the compressible two-phase flow problem.

This page intentionally left blank.

CHAPTER 3

TWO-PHASE HORIZONTAL FLOW SYSTEM OF INCOMPRESSIBLE FLUIDS SUBJECT TO CONSTANT FLUX BOUNDARIES

3.1 Overview

3.1.1 Objective

A numerical solver was developed using MATLAB to model two-phase incompressible flow problems subjected to a constant flow rate boundary condition at the inflow with varying initial conditions. Comparison with the experimental data by Melean et al. (2003) shows that the approximation methods used yield accurate and practical estimations of the infiltration variables of interest.

3.1.2 Governing Equation for the Invasion of a Wetting Fluid

This derivation models the invasion of a wetting fluid assuming both fluids are incompressible fluids in a rigid porous medium.

As in section 2.4.5, the mass balance equation in its most general form is written as follows:

$$\frac{\partial}{\partial t}\left[\rho_m \varphi S_m\right] + \frac{\partial}{\partial x}\left[\rho_m q_m\right] = 0 \tag{3.1}$$

Since the fluids are assumed to be incompressible, the density of the fluids can cancel out of both terms. Since the fluids are assumed to be in a rigid porous medium, the porosity is constant with respect to time and can be removed from the derivative. Based on these assumptions, the mass balance equation is simplified as follows for wetting and nonwetting fluids:

$$\varphi \frac{\partial S_w}{\partial t} + \frac{\partial q_w}{\partial x} = 0 \tag{3.2}$$

$$\varphi \frac{\partial S_{nw}}{\partial t} + \frac{\partial q_{nw}}{\partial x} = 0 \tag{3.3}$$

Since the sum of the degrees of saturation of the wetting and nonwetting fluids is always 1, adding equations (3.2) and (3.3) gives the following:

$$\frac{\partial(q_w + q_{nw})}{\partial x} = 0 \tag{3.4}$$

where the total flux is expressed as follows:

$$q_t = q_w + q_{nw} \tag{3.5}$$

From equation (3.4), it can be concluded that the total flux is constant with respect to the spatial domain. Assuming the flow is in the horizontal direction, there is no gravitational effect and the flux of the wetting and nonwetting fluids can be defined as in section 2.4.1 by Darcy's Law as follows:

$$q_w = \frac{-\kappa k_{rw}}{\mu_w} \frac{\partial P_w}{\partial x} \tag{3.6}$$

$$q_{nw} = \frac{-\kappa k_{rnw}}{\mu_{nw}} \frac{\partial P_{nw}}{\partial x} \tag{3.7}$$

Substituting equations (3.6) and (3.7) into (3.2) and (3.3), one obtains:

$$\varphi \frac{\partial S_w}{\partial t} + \frac{\partial}{\partial x}\left[\frac{-\kappa k_{rw}}{\mu_w} \frac{\partial P_w}{\partial x}\right] = 0 \tag{3.8}$$

$$\varphi \frac{\partial S_{nw}}{\partial t} + \frac{\partial}{\partial x}\left[\frac{-\kappa k_{rnw}}{\mu_{nw}} \frac{\partial P_{nw}}{\partial x}\right] = 0 \tag{3.9}$$

A third governing equation is given based on the total degree of saturation in the medium as follows:

$$S_w + S_{nw} = 1 \tag{3.10}$$

At this point, there are four unknowns, S_w, P_w, S_{nw} and P_{nw}, but only three equations, so a fourth equation is needed to complete the system. The fourth equation is the van Genuchten model for the soil water retention curve, which is discussed in section 2.4.2 to provide a relationship between the capillary pressure and the degree of saturation of the wetting fluid as follows:

$$S_w = \left(\frac{1}{(1+P_c/p_0)^N} \right)^M \left(1-S_r\right)+S_r \tag{3.11}$$

where constants *M* and *N* are model parameters based on the pore size distribution of the porous medium.

Based on the definition of capillary pressure ($P_c = P_{nw} - P_w$), Darcy's Law for wetting fluid can be rearranged as follows:

$$\frac{1}{-\kappa}\frac{q_w \mu_w}{k_{rw}} = \frac{\partial P_w}{\partial x} = \frac{\partial (P_{nw}-P_c)}{\partial x} \tag{3.12}$$

$$\frac{1}{-\kappa}\frac{q_{nw} \mu_{nw}}{k_{rnw}} = \frac{\partial P_{nw}}{\partial x} \tag{3.13}$$

Subtracting (3.13) from (3.12) gives the following equation:

$$\frac{1}{-\kappa}\left(\frac{q_w \mu_w}{k_{rw}} - \frac{q_{nw} \mu_{nw}}{k_{rnw}} \right) = -\frac{\partial P_c}{\partial x} \tag{3.14}$$

Since the sum of the flux of the wetting and nonwetting fluids is equal to the total flux, the flux of the nonwetting fluid can be defined as follows:

$$q_{nw} = q_t - q_w \tag{3.15}$$

With this information, the mass balance equation can be written as follows:

$$\varphi \frac{\partial S_w}{\partial t} + q_t \frac{\partial}{\partial x}\left[\frac{\mu_{nw}/k_{rnw}}{\mu_w/k_{rw} + \mu_{nw}/k_{rnw}} \right] + \frac{\partial}{\partial x}\left[\frac{\kappa}{\mu_w/k_{rw} + \mu_{nw}/k_{rnw}} \frac{\partial P_c}{\partial x} \right] = 0 \tag{3.16}$$

Equivalently, equation (3.16) can be expressed as follows:

$$\varphi \frac{\partial S_w}{\partial t} + q_t \frac{\partial}{\partial x}\left[\frac{\mu_{nw} k_{rw}}{\mu_w k_{rnw} + \mu_{nw} k_{rw}} \right] + \frac{\partial}{\partial x}\left[\frac{\kappa k_{rw} k_{rnw}}{\mu_w k_{rnw} + \mu_{nw} k_{rw}} \frac{\partial P_c}{\partial x} \right] = 0 \qquad (3.17)$$

This is the governing equation for incompressible two-phase horizontal flow subjected to a constant flux boundary conditions. The following notations are introduced to simplify the implementation:

$$h_w(S_w) = \frac{\mu_{nw} k_{rw}}{\mu_w k_{rnw} + \mu_{nw} k_{rw}} \qquad (3.18)$$

$$g_w(S_w) = \frac{\kappa k_{rw} k_{rnw}}{\mu_w k_{rnw} + \mu_{nw} k_{rw}} \qquad (3.19)$$

$$H_w = q_t h_w(S_w) \qquad (3.20)$$

$$G_w = g_w(S_w) \qquad (3.21)$$

Equations (3.20) and (3.21) are defined for later generalizations of the same code to solve a nonwetting invasion system.

With these simplifications, the mass balance equation can be written as follows:

$$\varphi \frac{\partial S_w}{\partial t} + \frac{\partial}{\partial x}\left[H_w \right] + \frac{\partial}{\partial x}\left[G_w \frac{\partial P_c}{\partial x} \right] = 0 \qquad (3.22)$$

Equation (3.22) is the version of the mass balance equation that was discretized and approximated using a modified Picard approximation.

3.2 Model Development

3.2.1 Discretization

A finite difference approximation with the implicit Euler backward algorithm was applied to discretize the governing equation as follows:

$$\varphi\frac{(S_{wj}^{\,n+1}-S_{wj}^{\,n})}{\Delta t}+\frac{H_{wj+1/2}^{\,n+1}-H_{wj-1/2}^{\,n+1}}{\Delta x}+\frac{1}{\Delta x}\left[G_{wj+1/2}^{\,n+1}\frac{P_{cj+1}^{n+1}-P_{cj}^{n+1}}{\Delta x}-G_{wj-1/2}^{\,n+1}\frac{P_{cj}^{n+1}-P_{cj-1}^{n+1}}{\Delta x}\right]=0 \tag{3.23}$$

where the superscript *n* represents the current time step and the subscript *j* represents the current node. Subscripts *j+1/2* and *j-1/2* represent half nodes adjacent to the current node *j*.

3.2.2 Modified Picard Iterations

The Picard method operates by solving for the variables of interest iteratively. The modified Picard iteration for equation (3.23) is written as follows:

$$\varphi\frac{(S_{wj}^{\,n+1,m+1}-S_{wj}^{\,n})}{\Delta t}+\frac{H_{wj+1/2}^{\,n+1,m}-H_{wj-1/2}^{\,n+1,m}}{\Delta x}+\frac{1}{\Delta x}\left[G_{wj+1/2}^{\,n+1,m}\frac{P_{cj+1}^{n+1,m+1}-P_{cj}^{\,n+1,m+1}}{\Delta x}-G_{wj-1/2}^{\,n+1,m}\frac{P_{cj}^{n+1,m+1}-P_{cj-1}^{n+1,m+1}}{\Delta x}\right]=0 \tag{3.24}$$

where the superscript *m* represents the current iteration. At the *m*th iteration, all values are known, and the values at the *m+1* iteration represent the unknown values being solved for. The capillary pressure variables at the *m+1* iteration can be approximated by adding an incremental value as follows:

$$P_{cj-1}^{n+1,m+1}=P_{cj-1}^{n+1,m}+\delta P_{cj-1}^{n+1,m+1} \tag{3.25}$$

$$P_{cj}^{n+1,m+1}=P_{cj}^{n+1,m}+\delta P_{cj}^{n+1,m+1} \tag{3.26}$$

$$P_{cj+1}^{n+1,m+1}=P_{cj+1}^{n+1,m}+\delta P_{cj+1}^{n+1,m+1} \tag{3.27}$$

The saturation variable at the *m+1* iteration is approximated as follows:

$$S_{wj}^{\,n+1,m+1}=S_{wj}^{\,n+1,m}+\left.\frac{\partial S_w}{\partial P_c}\right|^{n+1,m}\delta P_{cj}^{\,n+1,m+1} \tag{3.28}$$

where the variable C is defined based on the van Genuchten model of the soil water retention curve as follows:

$$C_j^{n+1,m} = \frac{\partial S_w}{\partial P_c} = (-N+1)\left(\left(\frac{P_c}{p_0}\right)^N + 1\right)^{-2+1/N}\left(\frac{P_c}{p_0}\right)^{N-1}\frac{1}{p_0}\left(1-S_r\right) \tag{3.29}$$

Equation (3.24) can be further written in terms of the unknown, δP_c, by substituting equations (3.25), (3.26), (3.27), (3.28) and (3.29):

$$\begin{gathered}
\varphi\frac{(S_{w\,j}^{\,n+1,m} + C^{n+1,m}\delta P_{c\,j}^{n+1,m+1} - S_{w\,j}^{\,n})}{\Delta t} + \frac{H_{w\,j+1/2}^{\,n+1,m} - H_{w\,j-1/2}^{\,n+1,m}}{\Delta x}\dots \\
+\frac{1}{\Delta x}\left[\begin{array}{l}
G_{w\,j+1/2}^{\,n+1,m}\,\dfrac{P_{c\,j+1}^{n+1,m} + \delta P_{c\,j+1}^{n+1,m+1} - P_{c\,j}^{n+1,m} - \delta P_{c\,j}^{n+1,m+1}}{\Delta x} \\
-G_{w\,j-1/2}^{\,n+1,m}\,\dfrac{P_{c\,j}^{n+1,m} + \delta P_{c\,j}^{n+1,m+1} - P_{c\,j-1}^{n+1,m} - \delta P_{c\,j-1}^{n+1,m+1}}{\Delta x}
\end{array}\right] = 0
\end{gathered} \tag{3.30}$$

3.2.3 *Solving for the Variables of Interest*

This equation is then solved for the unknown δP_c term at the next iteration ($m+1$) for all nodes ($j=1, 2\dots J\text{-}1, J$) as follows:

$$\begin{gathered}
\overbrace{\begin{array}{c}
\varphi\dfrac{(S_{w\,j}^{\,n+1,m} - S_{w\,j}^{\,n})}{\Delta t} + \dfrac{H_{w\,j+1/2}^{\,n+1,m} - H_{w\,j-1/2}^{\,n+1,m}}{\Delta x} \\
+\dfrac{1}{\Delta x}\left[G_{w\,j+1/2}^{\,n+1,m}\dfrac{P_{c\,j+1}^{n+1,m} - P_{c\,j}^{n+1,m}}{\Delta x} - G_{w,\,j-1/2}^{\,n+1,m}\dfrac{P_{c\,j}^{n+1,m} - P_{c\,j-1}^{n+1,m}}{\Delta x}\right]
\end{array}}^{-R} \\
\overbrace{+\left(\frac{1}{\Delta x^2}G_{w\,j+1/2}^{\,n+1,m}\right)}^{C}\boxed{\delta P_{c\,j+1}^{n+1,m+1}} + \overbrace{\left(\varphi\frac{C^{n+1,m}}{\Delta t} - \frac{1}{\Delta x^2}\left(G_{w\,j+1/2}^{\,n+1,m} + G_{w\,j-1/2}^{\,n+1,m}\right)\right)}^{B}\boxed{\delta P_{c\,j}^{\,n+1,m+1}} \\
\overbrace{+\left(\frac{1}{\Delta x^2}G_{w\,j-1/2}^{\,n+1,m}\right)}^{A}\boxed{\delta P_{c\,j-1}^{n+1,m+1}} = 0
\end{gathered} \tag{3.31}$$

This equation is then formulated into a matrix to solve for the δP_c vector as follows:

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \boxed{A} & \boxed{B} & \boxed{C} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_J & B_J \end{bmatrix} \begin{pmatrix} \delta P_{c1} \\ \delta P_{c2} \\ \vdots \\ \boxed{\delta P_{c\,j-1}} \\ \boxed{\delta P_{c\,j}} \\ \boxed{\delta P_{c\,j+1}} \\ \vdots \\ \delta P_{c\,J-1} \\ \delta P_{c\,J} \end{pmatrix} = - \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_{j-1} \\ \boxed{R_j} \\ R_{j+1} \\ \vdots \\ R_{J-1} \\ R_J \end{pmatrix} \tag{3.32}$$

where R represents all known values in the equation, boxed in green in equation (3.31), the blue, yellow and purple boxes represent the known values in front of the variables being solved for, which are the δP_c terms boxed in red.

3.2.4 Constant Flux Boundary at the Inflow Boundary Conditions

The first and last nodes ($j = 1$ and $j = J$) are formulated slightly differently than the rest of the matrix, as they are the nodes that enforce the constant flux boundary condition.

Since this solver enforces a constant flux boundary at the inflow ($j = 1$ at $x = 0$), Darcy's Law for the flux of a wetting fluid governs the boundary node:

$$q_{w3/2}^{n+1} = q_t h_{w3/2}^{n+1} + G_{w3/2}^{n+1} \frac{P_{c2}^{n+1} - P_{c1}^{n+1}}{\Delta x} \tag{3.33}$$

At the boundary where $x = 0$, the flux of the nonwetting fluid is zero so $q_w = q_t$. Then equation (3.33) can be written as follows:

$$q_t (h_{w3/2}^{n+1} - 1) + G_{w3/2}^{n+1} \frac{P_{c2}^{n+1} - P_{c1}^{n+1}}{\Delta x} = 0 \tag{3.34}$$

From here, equation (3.34) may be approximated using the modified Picard approximation as follows:

$$q_t (h_{w3/2}^{n+1,m} - 1) + G_{w3/2}^{n+1,m} \frac{P_{c2}^{n+1,m} + \delta P_{c2}^{n+1,m+1} - P_{c1}^{n+1,m} - \delta P_{c1}^{n+1,m+1}}{\Delta x} = 0 \tag{3.35}$$

which is solved for the δP_c increments similarly to equation (3.30):

$$\boxed{\overbrace{(h_{w3/2}^{n+1,m}-1)q_t+G_{w3/2}^{n+1,m}\frac{P_{c2}^{n+1,m}-P_{c1}^{n+1,m}}{\Delta x}}^{R_1}}\\ \boxed{-\overbrace{\left(\frac{1}{\Delta x}G_{w3/2}^{n+1,m}\right)}^{B_1}}\boxed{\delta P_{c1}^{n+1,m+1}}+\boxed{\overbrace{\left(\frac{1}{\Delta x}G_{w3/2}^{n+1,m}\right)}^{C_1}}\boxed{\delta P_{c2}^{n+1,m+1}}=0 \tag{3.36}$$

The inflow boundary condition is then implemented in the code as follows:

$$\begin{bmatrix} \boxed{B_1} & \boxed{C_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A & B & C & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & C_J \end{bmatrix} \begin{pmatrix} \boxed{\delta P_{c1}} \\ \boxed{\delta P_{c2}} \\ \vdots \\ \delta P_{c\,j-1} \\ \delta P_{c\,j} \\ \delta P_{c\,j+1} \\ \vdots \\ \delta P_{c\,J-1} \\ \delta P_{c\,J} \end{pmatrix} = - \begin{pmatrix} \boxed{R_1} \\ R_2 \\ \vdots \\ R_{j-1} \\ R_j \\ R_{j+1} \\ \vdots \\ R_{J-1} \\ R_J \end{pmatrix} \tag{3.37}$$

3.2.5 *Constant Pressure Boundary Condition at the Outflow*

At the outflow ($j = J$ at $x = L$), a constant pressure boundary is maintained to resemble the typical open-end conditions in most experimental studies. Based on the input pressure out the outflow, the change in pressure δP_c at the J^{th} node is zero. This is easily enforced as follows:

$$\boxed{1}\cdot\boxed{\delta P_{cJ}^{n+1,m+1}}=\boxed{0} \tag{3.38}$$

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A & B & C & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} \end{bmatrix} \begin{pmatrix} \delta P_{c1} \\ \delta P_{c2} \\ \vdots \\ \delta P_{c\,j-1} \\ \delta P_{c\,j} \\ \delta P_{c\,j+1} \\ \vdots \\ \delta P_{c\,J-1} \\ \boxed{\delta P_{c\,J}} \end{pmatrix} = - \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_{j-1} \\ R_j \\ R_{j+1} \\ \vdots \\ R_{J-1} \\ \boxed{0} \end{pmatrix} \tag{3.39}$$

The matrices discussed in equations (3.32), (3.37) and (3.39) were combined to form one large matrix. Each diagonal in the matrix was built as an independent vector before assembling the tri-diagonal matrix to be solved using the Thomas algorithm.

3.2.6 Implementation in MATLAB

The code solves the matrices above by taking an initial guess for the capillary pressure at all nodes at the initial time $t_0 = 0$. The code iterates through once, and calculates the error as follows:

$$error = \max(abs(\delta P_c)) \tag{3.40}$$

If the calculated error is less than some set tolerance, set to be *tol =1E–04*, then two successive iterations give very little change in the calculated P_c values and convergence is reached. As an example, consider the first time step. When the change in pressure between successive iterations is smaller than the set tolerance, convergence is reached for the first time step. Following this, these values become the new initial guess for the second-time step, and the code begins to iterate until convergence is reached for the second time step. Then the P_c values determined in the second time step become the initial guess for the third time step, and so on. This process continues until the desired end time is reached.

3.3 Parametric Study

A parametric study was performed by varying different model parameters in the code. The purpose of the parametric study was two-fold: to analyze how changing various model parameters affects the shapes and velocity of the imbibition front, and to eventually produce a best-fit comparison to the Melean’s data of imbibition fronts, which is discussed in section 3.4.

3.3.1 Experimental Background

Melean and coworkers performed an experimental study on the horizontal dispersion of imbibition fronts in an incompressible fluid oil-water system. The 40cm long soil sample was made of quartzitic grains and confined in a glass cylinder with a diameter of 3cm. Injection velocities were varied, and data points were collected using X-ray tomography (Melean, Broseta, Hasmy, & Blossey, 2003). The input parameters extracted from Melean's paper are listed in Table 3.1.

Table 3.1 Input parameters given by Melean et al. (Melean, Broseta, Hasmy, & Blossey, 2003)

Parameter	Symbol	Value	Unit
Intrinsic Permeability	K_{sat} or κ	1.00E-09	m^2
Wetting Fluid Viscocity	μ_w	8.90E-04	kg/m/s
Nonwetting Fluid Viscocity	μ_{nw}	1.03E-04	kg/m/s
Inflow of Wetting Fluid	q	varies	mL/hr
Porosity	φ	0.35	-
Length	x	0.4	m

Melean and coworkers neglected to provide any information regarding the initial suction throughout the sample $P_{c,init}$, and soil water retention curve parameters, including the SWRC *N*-value, the residual saturation S_r, and the entry pressure p_0. As mentioned, a parametric study was performed by varying each of these parameters independently.

3.3.2 Baseline Case

A baseline case was selected as a starting point for the parametric study. The input parameters for the baseline case are presented in Table 3.2.

Table 3.2 Input parameters for the baseline case used in the parametric study

Parameter	Symbol	Value	Unit
Intrinsic Permeability	K_{sat} or κ	1.00E-09	m^2
Wetting Fluid Viscocity	μ_w	8.90E-04	kg/m/s
Nonwetting Fluid Viscocity	μ_{nw}	1.03E-04	kg/m/s
Inflow of Wetting Fluid	q_w	100	mL/hr
Porosity	φ	0.35	-
Length	x	0.4	m
N-Value	N	3.4	-
Air Entry Pressure	p_0	300	
Initial Capillary Pressure	$P_{c,init}$	1200	Pa
Residual Saturation	S_r	0.1	-

The simulated wetting front and capillary pressure profile for the baseline case are presented in Figure 3.1 and Figure 3.2.

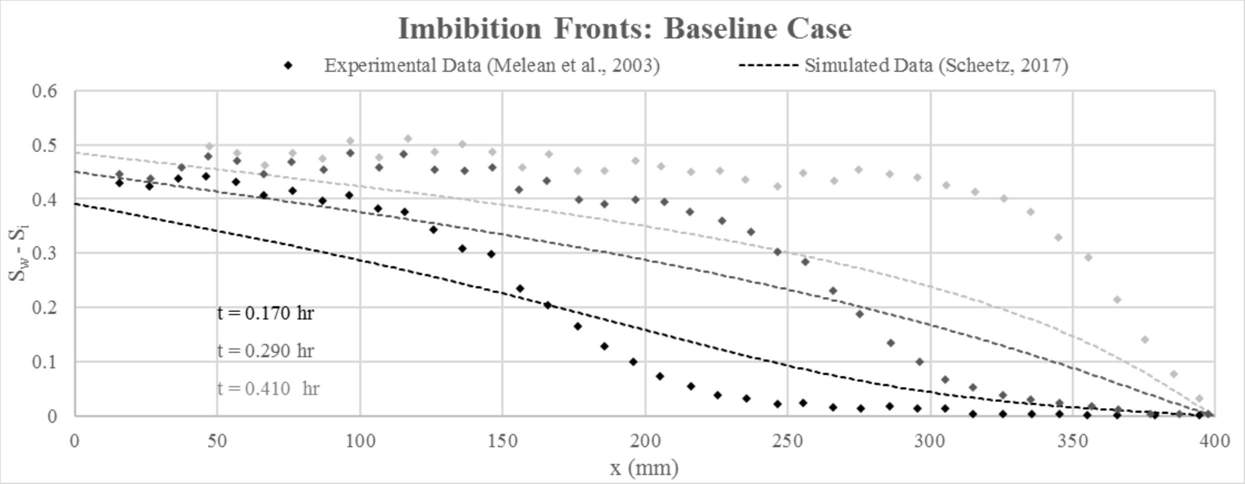


Figure 3.1 A baseline case for the imbibition fronts for a two-phase incompressible flow system subject to a constant flux boundary condition of 100 mL/hour compared against the Melean et al. 2003 experimental data.

From Figure 3.1, it is observed that, in general, the shape of the fronts is correct. However, with the listed input parameters, the peak degree of saturation is lower and the wetting front is travelling too fast as compared to the experimental data by Melean et al.

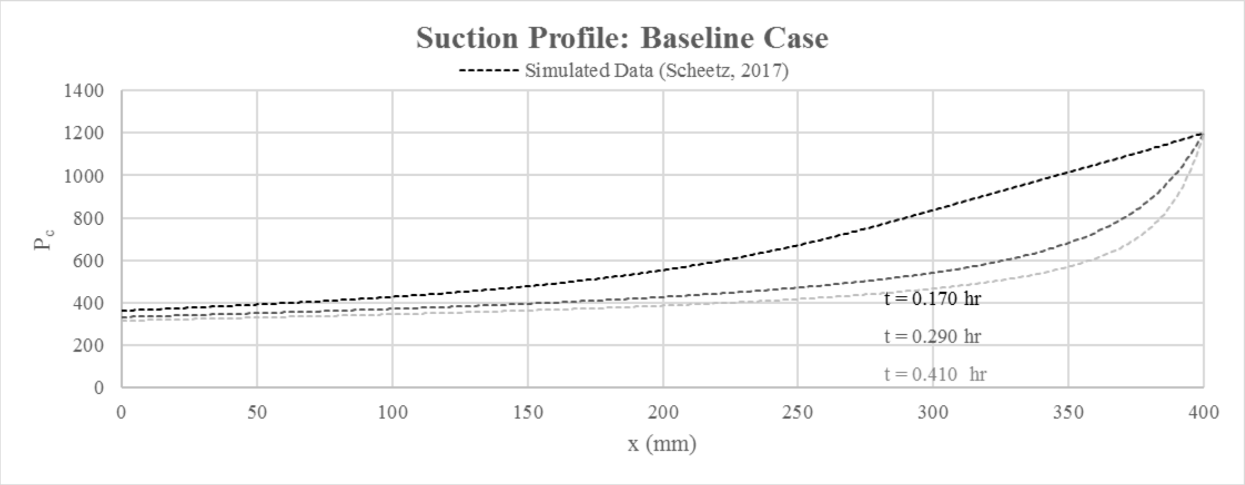


Figure 3.2 A corresponding baseline case for the pressure profiles for a two-phase incompressible flow system subject to a constant flux boundary condition of 100 mL/hour.

Figure 3.2 presents the suction profiles corresponding to the wetting fronts presented in Figure 3.1.

3.3.3 Vary Soil Water Retention Curve (SWRC) Parameter N

The first parameter studied in the parametric study was the soil water retention curve *N*-value discussed in section 2.4.2.

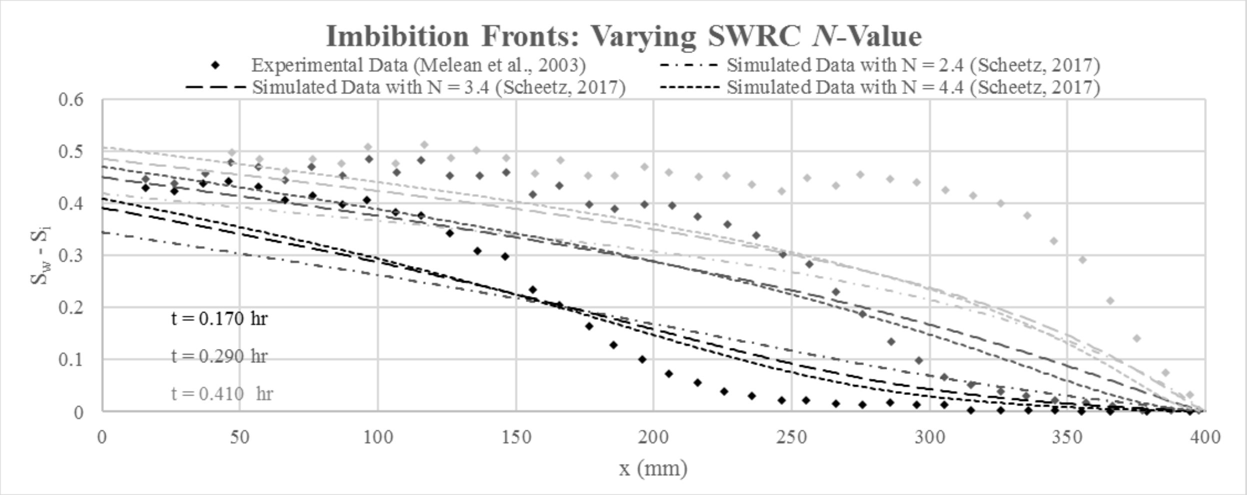


Figure 3.3 Varying SWRC N-values to understand the effects on the imbibition fronts for a two-phase incompressible flow system subject to a constant flux boundary condition of 100 mL/hour. Fronts are compared against the Melean et al. 2003 experimental data.

From Figure 3.3, it is observed that higher *N*-values create more gradual wetting front through the porous medium. Also, higher *N*-values result in a higher degree of saturation at the entry boundary. This difference in degree of saturation is directly related to pressure difference throughout the porous medium via the SWRC. For larger *N*-values, the difference of S_w either sides of the front is related to the traveling speed of the front. Recall that the system is subjected to constant flux boundary, which means that for a given time the injected volume of fluid is fixed and the area underneath the $S_w(x)$ profiles at a given time should be the same. This 'constant area' constraint implies that the faster the front it travelling, the less saturated the tailing area is.

One might conclude, based on Figure 3.3, that water can travel more easily through clay-like soils than sand-like soils. This, however, is not correct because more variables need to be altered to reflect true sand-like soils and clay-like soils, including the entry pressure (p_0) and the intrinsic permeability κ.

One can conclude, however, that higher *N*-values indicate a lower effective permeability during the invasion of wetting phase, which means a slower travelling speed of the front through the porous medium.

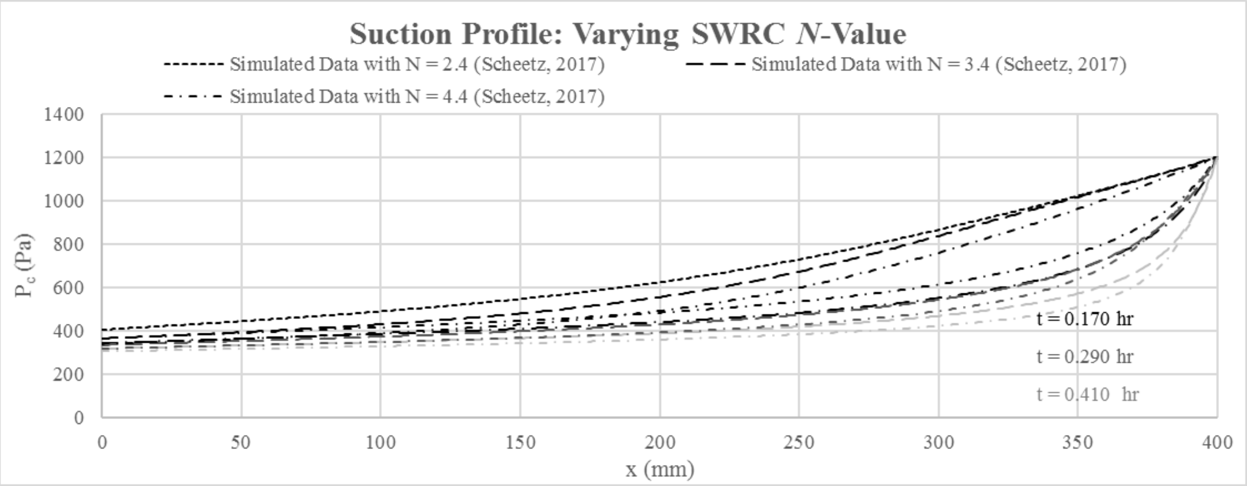


Figure 3.4 Corresponding pressure profiles to varying SWRC N-values.

Figure 3.4 shows the pressure profiles corresponding to the wetting fronts shown in Figure 3.3

3.3.4 Vary Initial Suction present throughout the Porous Media

The second parameter analyzed during the parametric study was the initial suction value, which represents the suction present in the porous media prior to the injection of the wetting fluid.

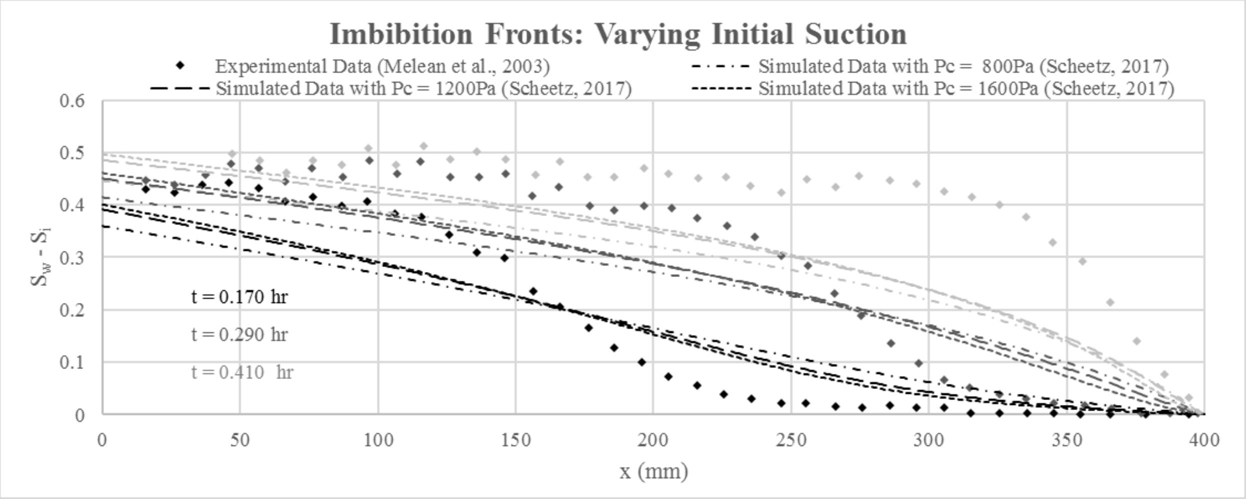


Figure 3.5 Varying initial suction, P_c, input values to understand the effects on the imbibition fronts for a two-phase incompressible flow system subject to a constant flux boundary condition of 100 mL/hour. Fronts are compared against the Melean et al. 2003 experimental data.

From Figure 3.5, it is observed that even relatively large differences in initial suction have very little effect on the shapes of the imbibition fronts. It can be observed, though, that larger initial suction values slow down the traveling of wetting phase through the porous medium. This makes sense, as larger initial suction values indicate an initially drier soil sample. If the soil is initially very dry, it is more difficult for the front to move through the medium. This is because higher initial suction values mean a smaller relative permeability of the wetting front, which reduces the ease and traveling velocity of the wetting front in the medium.

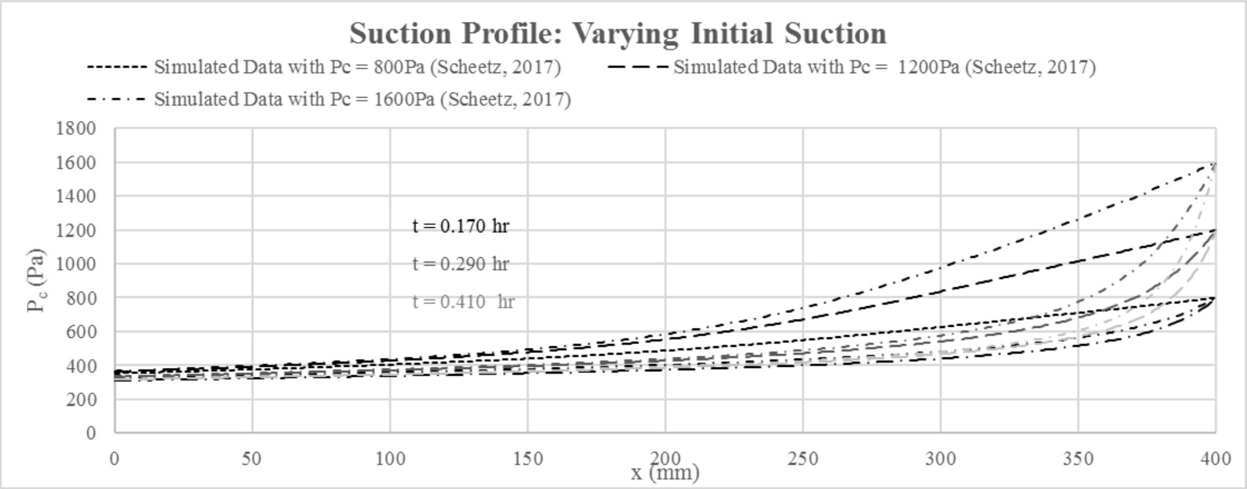


Figure 3.6 Corresponding pressure profiles to varying initial suction, P_c, values.

Figure 3.6 presents the pressure profiles corresponding to the wetting fronts shown in Figure 3.5. Note that the pressure differences between the profiles are caused by the imposed initial conditions.

3.3.5 Vary the Residual Saturation within the Porous Media

The next parameter analyzed in the parametric study was the residual saturation, which represents the degree of saturation of the wetting fluid present in the porous media prior to the injection of the wetting fluid. Note that a nonzero value for the residual degree of saturation means that, even if you have an infinitely high initial suction making the media as dry as possible, there will still be some initial nonzero degree of saturation in the porous media.

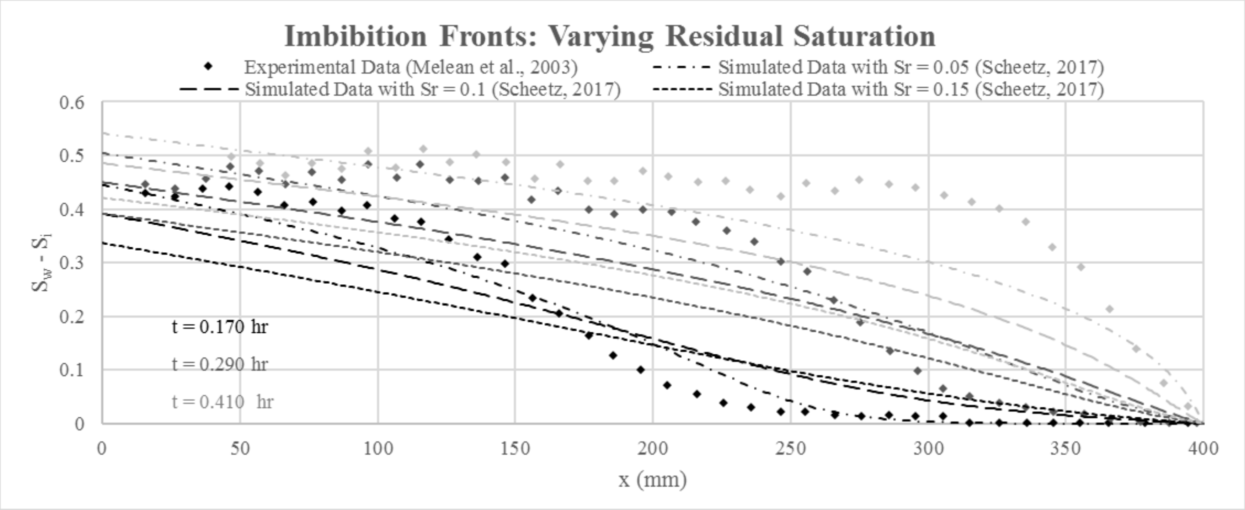


Figure 3.7 Varying residual saturation, S_r, input values to understand the effects on the imbibition fronts for a two-phase incompressible flow system subject to a constant flux boundary condition of 100 mL/hour. Fronts are compared against the Melean et al. 2003 experimental data.

From Figure 3.7, it is observed that higher values of residual saturation result in a wetting front with a much shallower slope and faster travelling speed. Similar to the arguments made in section 3.3.4, an initially wet sample improves the ease at which the front can travel through the media and thus increases the travelling speed of the front. A higher degree of residual saturation leads to a higher relative permeability, which leads to a faster travelling speed of the front.

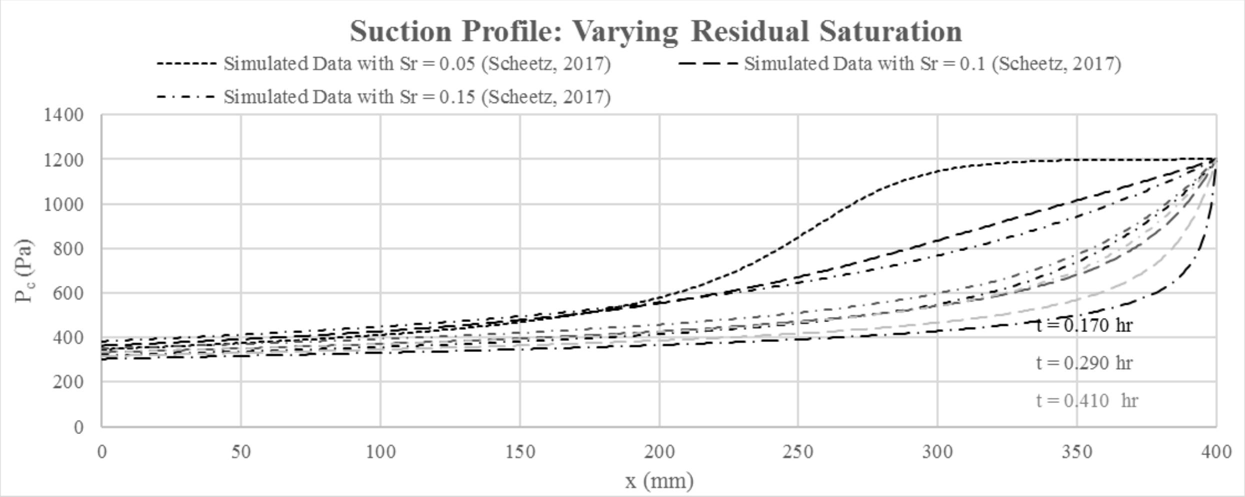


Figure 3.8 Corresponding pressure profiles to varying residual saturation, S_r, values.

Figure 3.8 presents the pressure profiles corresponding to the saturation profiles shown in Figure 3.7.

3.3.6 Vary the Entry Pressure

The final parameter analyzed during the parametric study was the entry pressure, p_0. The entry pressure, similar to the SWRC *N*-value, reflects the type of soil present in the porous media. Typically, higher entry pressures resemble clay-like soils and lower entry pressures resemble sand-like soils.

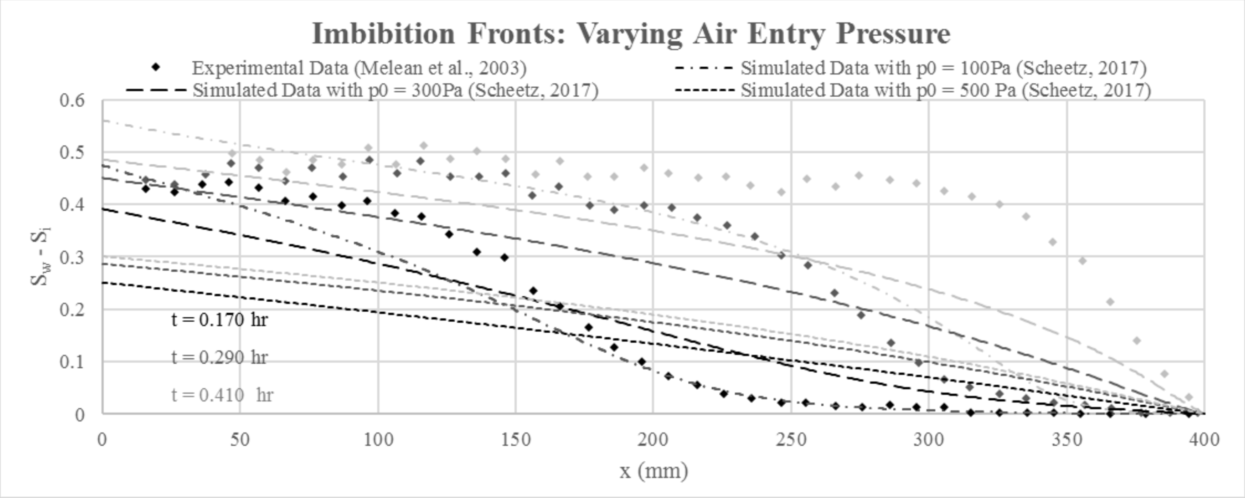


Figure 3.9 Varying entry pressure, p_0, input values to understand the effects on the imbibition fronts for a two-phase incompressible flow system subject to a constant flux boundary condition of 100 mL/hour. Fronts are compared against the Melean et al. 2003 experimental data.

From Figure 3.9, it is observed that higher entry pressures result in wetting fronts travelling relatively quickly with shallow slopes. With a higher entry pressure, it is easier for the soil to get wet during the wetting process. This indicates that the relative permeability of the water relatively increases with a high entry pressure, which increases the speed of the front.

Looking at Figure 3.9, one might conclude that water can travel more easily through clayey soils than sandy soils. Again, this conclusion is incorrect. The reason the front travels faster through soils with a higher entry pressure is because varying the entry pressure alone is not enough to represent a clay-like soil, as other parameters including the intrinsic permeability and SWRC *N*-values need to be altered as well.

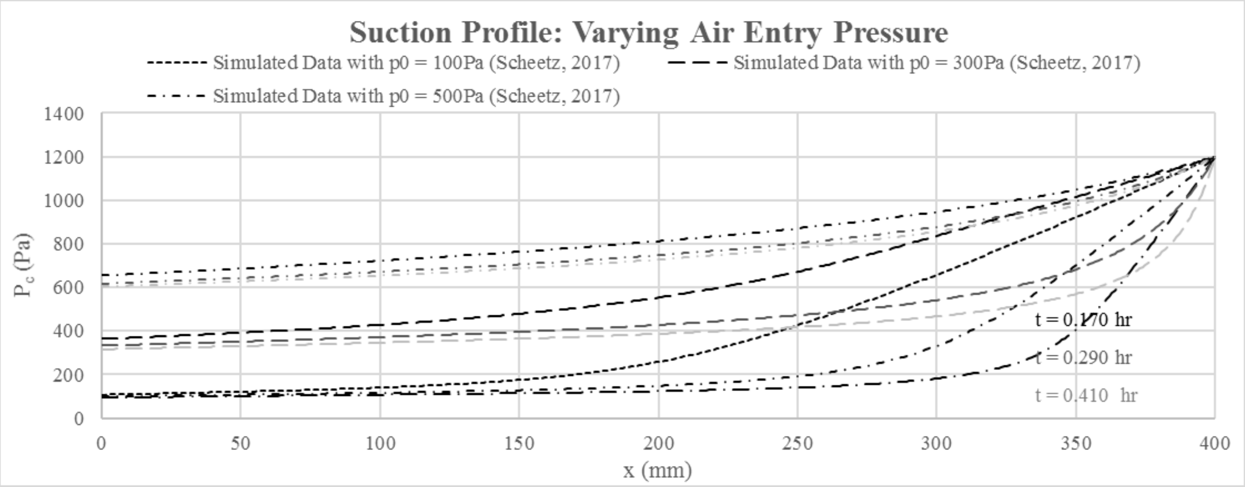


Figure 3.10 Corresponding pressure profiles to varying entry pressure, p_0, values.

Figure 3.10 presents the pressure profiles corresponding to the saturation profiles shown in Figure 3.9.

3.4 Model Validation

The model was validated by synthesizing the results from the parametric study in section 3.3 to produce a "best fit" to the experimental data captured by Melean et al. in 2003 with an inflow rate of 100 mL/hr. Table 3.3 presents the "best fit" parameters used to match the experimental data captured by Melean et al. when $q_w = 100$mL/hr.

Table 3.3 Best fit parameters based on experimental data from Melean et al. in 2003.

Parameter	Symbol	Value	Unit
Intrinsic Permeability	K_{sat} or κ	1.00E-09	m^2
Wetting Fluid Viscocity	μ_w	8.90E-04	kg/m/s
Nonwetting Fluid Viscocity	μ_{nw}	1.03E-04	kg/m/s
Inflow of Wetting Fluid	q_w	varies	mL/hr
Porosity	φ	0.35	-
Length	x	0.4	m
N-Value	N	4.0	-
Air Entry Pressure	p_0	300	
Initial Capillary Pressure	$P_{c,init}$	1700	Pa
Residual Saturation	S_r	0.05	-

Figure 3.11 and Figure 3.12 present the best fit imbibition fronts and the corresponding pressure profiles.

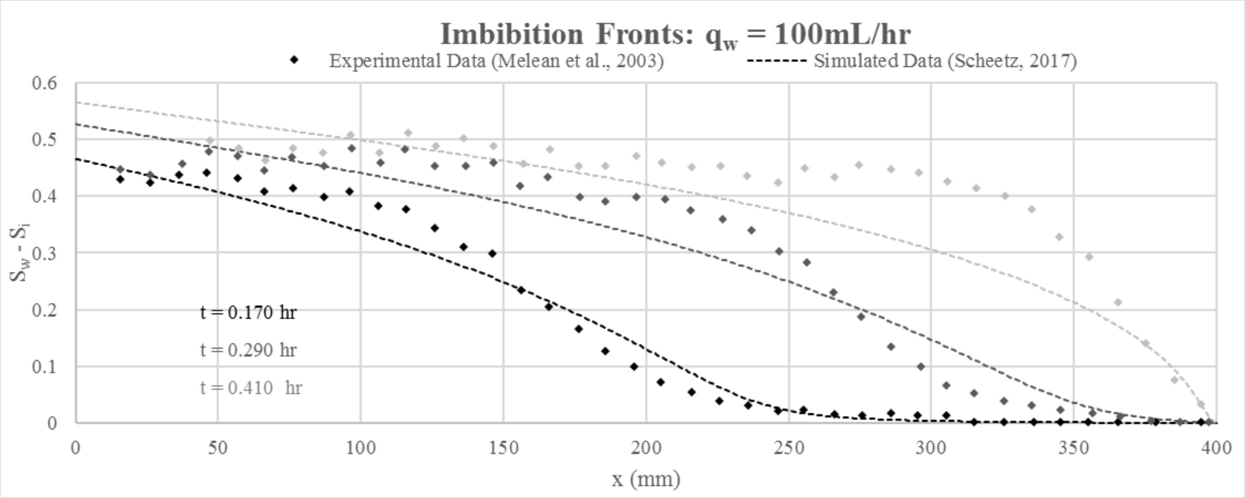


Figure 3.11 Input parameters were selected to produce the best fit to the Melean et al. 2003 experimental data for the imbibition fronts of a two-phase incompressible flow system subject to a constant flux boundary condition of 100 mL/hour.

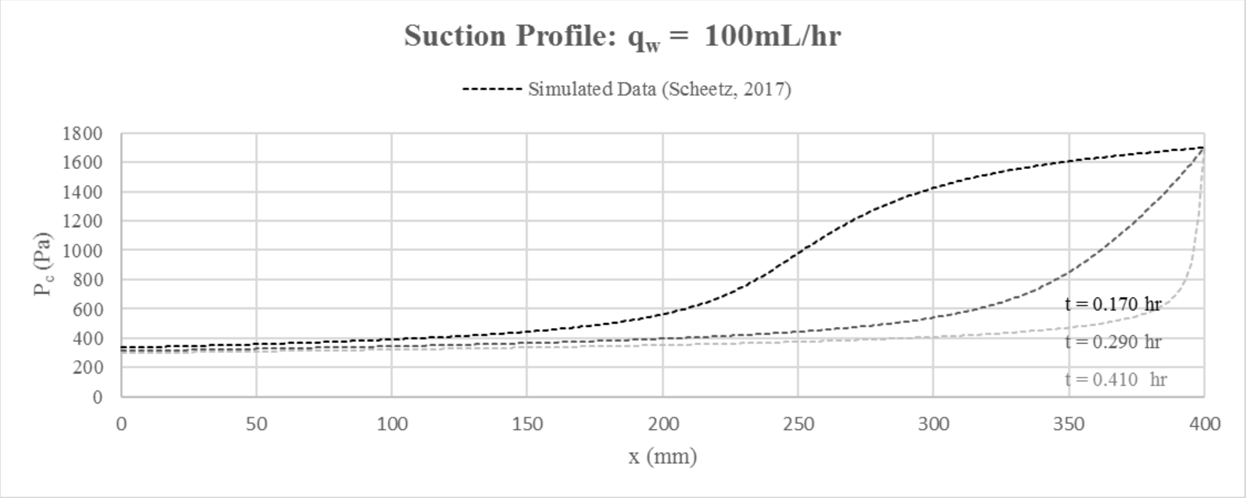


Figure 3.12 Corresponding pressure profiles for the imbibition fronts' best fit to the Melean et al. 2003 experimental data with an inflow rate of 100 mL/hr.

From Figure 3.11 and Figure 3.12, it is observed that the simulated solution matches the experimental data captured by Melean et al. moderately well when the inflow rate is 100 mL/hr, but the simulated fronts are travelling a little bit too quickly to match the data. Unfortunately, with such large flow rates, the shapes of the curves change very little with varying input parameters, as observed in the parametric study in section 3.3. Input parameters would need to be varied greatly to produce a “better best fit” to Melean’s data with an injection rate of 100mL/hr. However, researchers found that the solver is extremely sensitive to the input parameters (discussed further in section 3.6.2), and time constraints limited researchers from finding the best fit possible.

Despite this, model validation continued, and the next step in model validation was to use the same parameters listed in Table 3.3 to try to capture the experimental response with a different input flowrate, e.g. comparing the predicted saturation profiles with the data collected under an injection flow rate of 300 mL/hr and 3 mL/hr.

Figure 3.13 and Figure 3.14 show the imbibition fronts and the corresponding pressure profiles resulting from changing the flow rate of the wetting fluid q_w to 300 mL/hr.

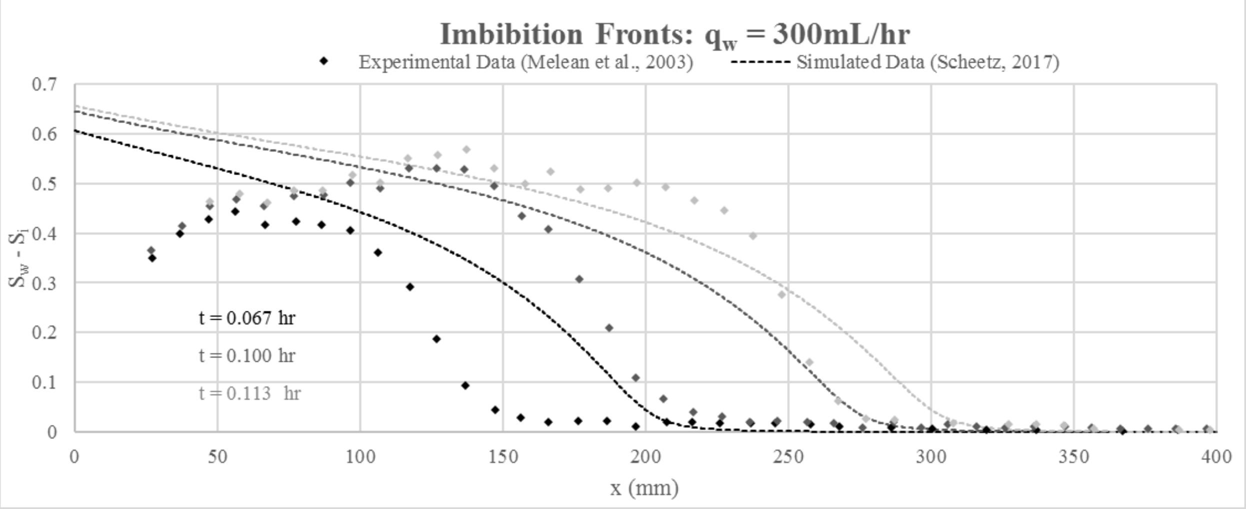


Figure 3.13 Input parameters were selected to produce the best fit to the Melean et al. 2003 experimental data for the imbibition fronts of a two-phase incompressible flow system subject to a constant flux boundary condition of 300 mL/hour.

It is observed from Figure 3.13 that, again, the simulated fronts are travelling a little bit too fast to match the data collected by Melean and coworkers when the injection rate is 300mL/hr. This is likely due to the same factors affecting the imbibition fronts with an injection rate of 100mL/hr.

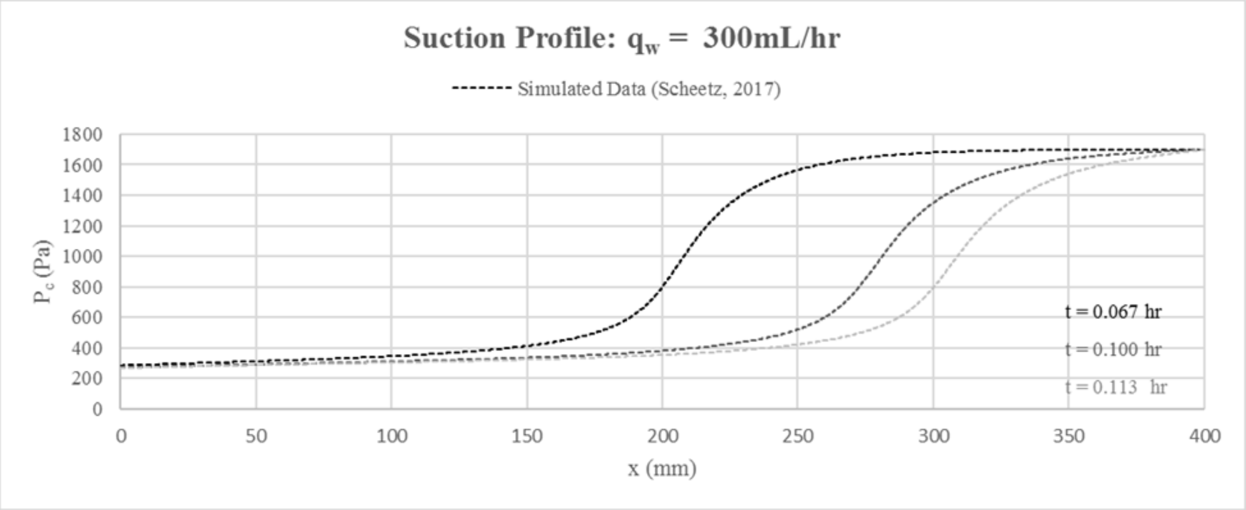


Figure 3.14 Corresponding pressure profiles for the imbibition fronts' best fit to the Melean et al. 2003 experimental data with an inflow rate is 300 mL/hr.

Figure 3.14 presents the suction profiles corresponding to the wetting fronts shown in Figure 3.13.

The paper by Melean et al. also gives experimental data corresponding to an injection rate of 3 mL/hr. Figure 3.15 and Figure 3.16 show the wetting fronts and the corresponding pressure profiles resulting from changing the flow rate *q* to 3 mL/hr.

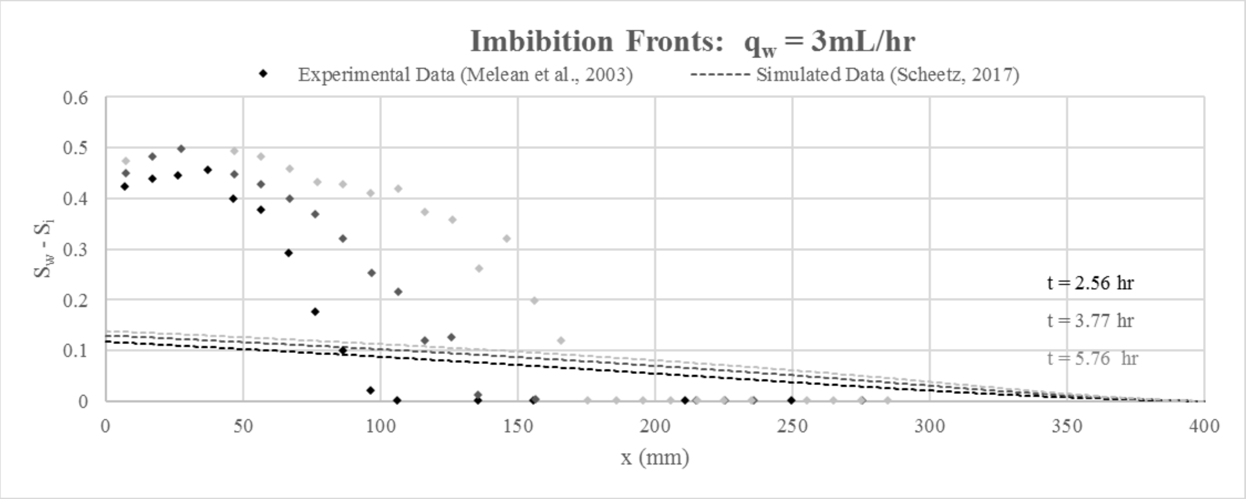


Figure 3.15 Input parameters were selected to produce the best fit to the Melean et al. 2003 experimental data for the imbibition fronts of a two-phase incompressible flow system subject to a constant flux boundary condition of 3 mL/hour.

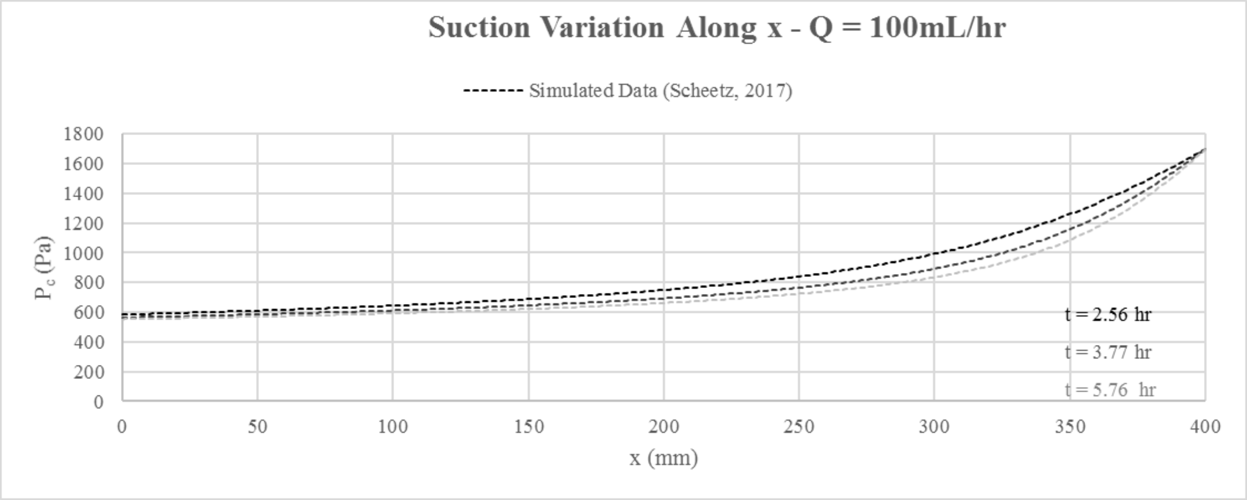


Figure 3.16 Corresponding pressure profiles for the imbibition fronts' best fit to the Melean et al. 2003 experimental data with an inflow rate of 3 mL/hr.

From Figure 3.15, it observed that, with the input parameters listed in Table 3.3, researchers were unable to match Melean's experimental data for an inflow rate of 3mL/hr. This is discussed further in section 3.6.1.

3.5 Nonwetting invasion

The numerical simulator capable of modelling the two-phase horizontal flow system of incompressible fluids subject to constant flux boundaries was formulated so that it could model both wetting and nonwetting invasions by changing one variables in the inputs. The code was written such that changing the variable *flag* to 1 or 2 changes the invasion from wetting to nonwetting, respectively. The MATLAB code for the invasion of a nonwetting fluid is functions exactly as it does for the invasion of a wetting fluid. Additionally, though the development is not discussed in this thesis, the code can also be modified slightly to solve the classical Richard's equation.

3.5.1 Governing Equation for the Invasion of a Nonwetting Fluid

A similar derivation as discussed in section 3.1.2 can be performed for the invasion of a nonwetting fluid in a rigid porous medium. The resulting governing equation is given as follows:

$$\varphi\frac{\partial S_w}{\partial t}-q_t\frac{\partial}{\partial x}\left[\frac{\mu_w k_{rnw}}{\mu_w k_{rnw}+\mu_{nw}k_{rw}}\right]+\frac{\partial}{\partial x}\left[\frac{\kappa k_{rw}k_{rnw}}{\mu_w k_{rnw}+\mu_{nw}k_{rw}}\frac{\partial P_c}{\partial x}\right]=0 \tag{3.41}$$

where the following substitutions were made to simplify the model:

$$h_{nw}=\frac{\mu_w k_{rnw}}{\mu_w k_{rnw}+\mu_{nw}k_{rw}} \tag{3.42}$$

$$g_{nw}=\frac{\kappa k_{rw}k_{rnw}}{\mu_w k_{rnw}+\mu_{nw}k_{rw}} \tag{3.43}$$

$$H_{nw}=-q_t h_{nw} \tag{3.44}$$

$$G_{nw}=g_{nw} \tag{3.45}$$

With these substitutions, the mass balance equation modeling the invasion of a nonwetting fluid can be written as follows:

$$\varphi\frac{\partial S_w}{\partial t}-\frac{\partial}{\partial x}\left[H_{nw}\right]+\frac{\partial}{\partial x}\left[G_{nw}\frac{\partial P_c}{\partial x}\right]=0 \tag{3.46}$$

This is the equation that is then discretized and approximated using the modified Picard approximation to give the solution resulting from the invasion of a nonwetting fluid.

3.5.2 Discretization

The discretization of equation (3.46) using a finite difference approximation with the implicit Euler backward algorithm is performed as follows:

$$\begin{aligned}&\varphi\frac{(S_{w\,j}^{\,n+1}-S_{w\,j}^{\,n})}{\Delta t}+\frac{H_{nw\,j+1/2}^{\,n+1}-H_{nw\,j-1/2}^{\,n+1}}{\Delta x}\cdots\\&+\frac{1}{\Delta x}\left[G_{nw\,j+1/2}^{\,n+1}\frac{P_{c\,j+1}^{n+1}-P_{c\,j}^{n+1}}{\Delta x}-G_{nw\,j-1/2}^{\,n+1}\frac{P_{c\,j}^{n+1}-P_{c\,j-1}^{n+1}}{\Delta x}\right]=0\end{aligned} \tag{3.47}$$

where the superscript *n* represents the current time step and the subscript *j* represents the current node. Subscripts *j+1/2* and *j-1/2* represent half nodes adjacent to the current node *j*.

3.5.3 Modified Picard Method

The modified Picard iteration for equation (3.47) is written as follows:

$$\varphi\frac{(S_{wj}^{n+1,m}+C_j^{n+1,m}\delta P_{cj}^{n+1,m+1}-S_{wj}^{n})}{\Delta t}+\frac{H_{nw\,j+1/2}^{n+1,m}-H_{nw\,j-1/2}^{n+1,m}}{\Delta x}+\frac{1}{\Delta x}\left[\begin{array}{l}G_{nw\,j+1/2}^{n+1,m}\dfrac{P_{c\,j+1}^{n+1,m}+\delta P_{c\,j+1}^{n+1,m+1}-P_{cj}^{n+1,m}-\delta P_{cj}^{n+1,m+1}}{\Delta x}\\ -G_{nw\,j-1/2}^{n+1,m}\dfrac{P_{cj}^{n+1.m}+\delta P_{cj}^{n+1,m+1}-P_{c\,j-1}^{n+1,m}-\delta P_{c\,j-1}^{n+1,m+1}}{\Delta x}\end{array}\right]=0 \qquad (3.48)$$

where the superscript *m* represents the current iteration. At the *m*th iteration, all values are known, and the values at the *m+1* iteration represent the unknown values being solved for. From this point, the variables of interest in equation (3.48) are put into a large matrix and solved for the same way as discussed in section 3.2.3.

3.5.4 Constant Flux Boundary Condition at the Inflow

The inflow boundary node for the invasion of a nonwetting fluid is governed by Darcy's Law for the flux of a nonwetting fluid as follows:

$$q_{nw3/2}^{n+1}=q_t h_{nw3/2}^{n+1}-G_{nw3/2}^{n+1}\frac{P_{c2}^{n+1}-P_{c1}^{n+1}}{\Delta x} \qquad (3.49)$$

At the inflow boundary where $x = 0$, the flux of the wetting fluid is zero so $q_{nw} = q_t$. Then equation (3.49) can be written as follows:

$$q_t(h_{nw3/2}^{n+1}-1)-G_{nw3/2}^{n+1}\frac{P_{c2}^{n+1}-P_{c1}^{n+1}}{\Delta x}=0 \qquad (3.50)$$

From here, equation (3.50) may be approximated using the modified Picard approximation as follows:

$$q_t(h_{nw3/2}^{n+1,m}-1)-G_{nw3/2}^{n+1,m}\frac{P_{c2}^{n+1,m}+\delta P_{c2}^{n+1,m+1}-P_{c1}^{n+1,m}-\delta P_{c1}^{n+1,m+1}}{\Delta x}=0 \qquad (3.51)$$

The boundary for the invasion of a nonwetting fluid are implemented into the code exactly as discussed for the invasion of a wetting fluid in section 3.2.4.

3.5.5 Constant Pressure Boundary Condition at the Outflow

At the outflow, ($j = J$ at $x = L$), a constant pressure boundary is maintained to resemble the typical open-end conditions in most experimental studies. This means that δP_c at the J^{th} node is zero. This is easily enforced exactly the same as in section 3.2.5.

3.5.6 Simulated Invasion of a Nonwetting Fluid

Table 3.4 presents the input parameters used to model a nonwetting invasion in a porous medium.

Table 3.4 Input parameters used to model a nonwetting invasion

Parameter	**Symbol**	**Value**	**Unit**
Intrinsic Permeability	K_{sat} or κ	1.00E-09	m^2
Wetting Fluid Viscocity	μ_w	8.90E-04	kg/m/s
Nonwetting Fluid Viscocity	μ_{nw}	1.03E-04	kg/m/s
Inflow of Wetting Fluid	q_{nw}	100	mL/hr
Porosity	φ	0.35	-
Length	x	0.4	m
N-Value	N	2.0	-
Air Entry Pressure	p_0	600	
Initial Capillary Pressure	$P_{c,init}$	700	Pa
Residual Saturation	S_r	0.1	-

Figure 3.17 and Figure 3.18 show the simulated response of the system during the invasion of a nonwetting fluid.

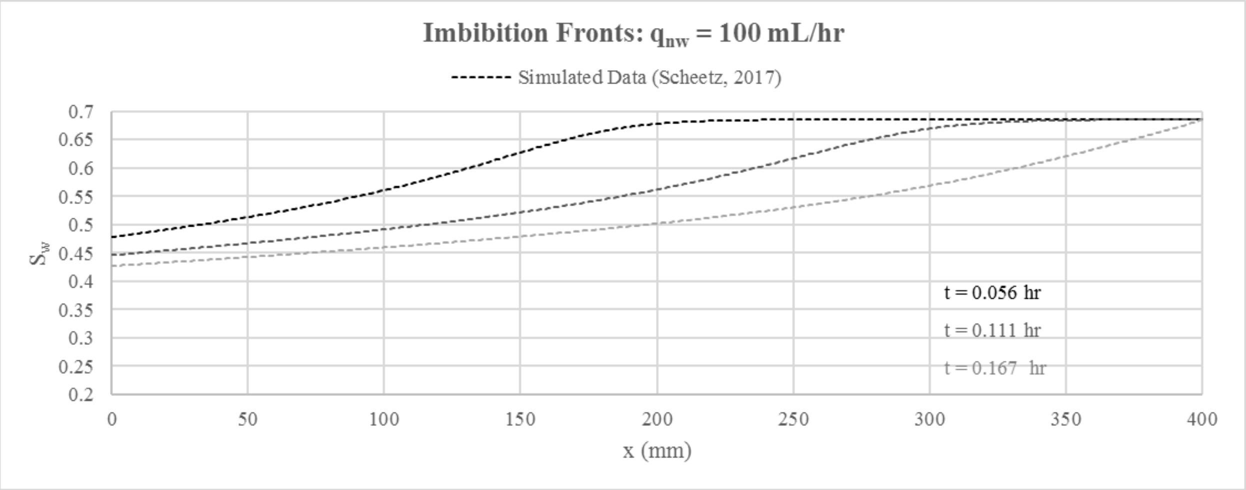


Figure 3.17 Imbibition fronts for the invasion of a nonwetting fluid resulting from the two-phase incompressible flow system subject to a constant flux boundary condition of 15 mL/hr.

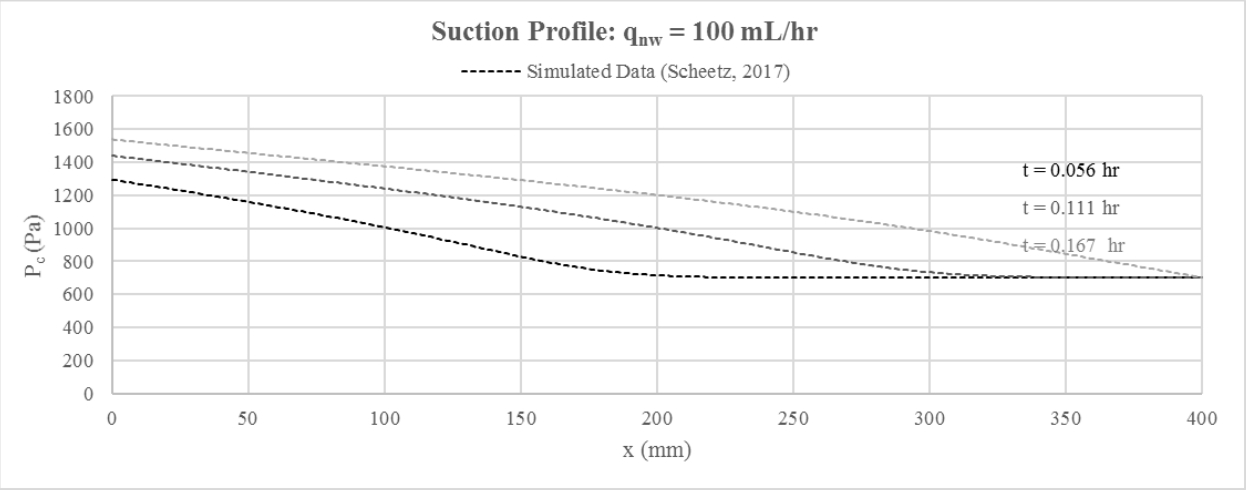


Figure 3.18 Corresponding pressure profiles for the simulated invasion of a nonwetting fluid.

3.6 Discussion

3.6.1 Fitting parameters given by Melean et al.

With the set of input parameters listed in Table 3.3, the numerical simulator developed for incompressible two-phase flow subject to a constant flux boundary condition at the inflow produces a moderately good match to the experimental data when injection rates of 100 and 300mL/hr is considered. However, at much smaller injection rates, the shapes of the experimental and simulated imbibition fronts are extremely different. There may be a few different reasons for this.

During an initial parametric study, the simulator was used to match Melean's data for an injection rate of 3mL/hr. This study revealed that extremely – perhaps unrealistically – low values for entry pressure (p_0 = *4*Pa) and initial suction ($P_{c,init}$ = *10*Pa) were necessary in order to match Melean's data for an injection rate of 3 mL/hr,. This value of p_0 = *4*Pa is extremely low, as it should be closer to p_0 = *200-300*Pa to model the quartzitic grains with *radii* ≈ *100*μm used in Melean's experimental study. Such a low entry pressure value is capable of influencing the shape of the soil water retention curve drastically. See Figure 3.19 for reference.

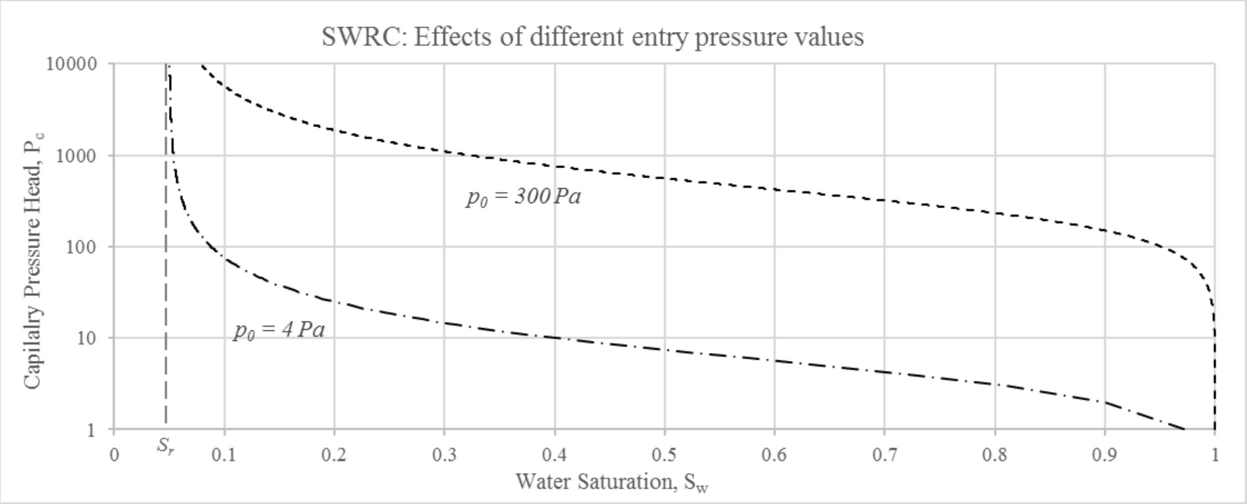


Figure 3.19 The effects of entry pressure on the shape of the soil water retention curve.

An extremely low value for the entry pressure creates a much sharper soil water retention curve. It is believed that this low entry pressure influenced through the solution greatly.

See Table 3.5 for the input parameters selected based on the initial parametric study to produce a best fit to Melean's experimental data with an injection rate of 3mL/hr in Figure 3.20.

Table 3.5 Alternative set of best fit parameters based on experimental data from Melean et al. in 2003

Parameter	Symbol	Value	Unit
Intrinsic Permeability	K_{sat} or κ	1.00E-09	m^2
Wetting Fluid Viscocity	μ_w	8.90E-04	kg/m/s
Nonwetting Fluid Viscocity	μ_{nw}	1.03E-04	kg/m/s
Inflow of Wetting Fluid	q_w	varies	mL/hr
Porosity	φ	0.35	-
Length	x	0.4	m
N-Value	N	1.6	-
Air Entry Pressure	p_0	4	
Initial Capillary Pressure	$P_{c,init}$	400	Pa
Residual Saturation	S_r	0.1	-

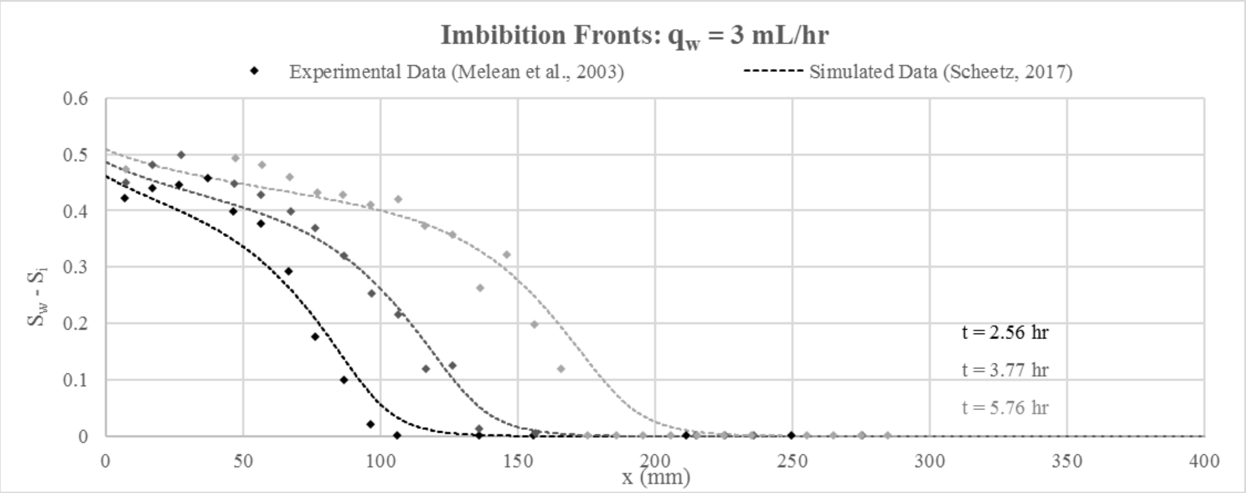


Figure 3.20 New set of input parameters selected to match the experimental data given for an injection rate of 3mL/hr given by Melean et al. in 2003. System is a two-phase incompressible flow system subject to a constant flux boundary of 3 mL/hr.

From Figure 3.20, it is exciting to see a very good match the Melean's experimental data. Unfortunately, this new set of input parameters does not produce a suitable match for the higher flow rates of 100 and 300mL/hr. See Figure 3.21 and Figure 3.22 for reference.

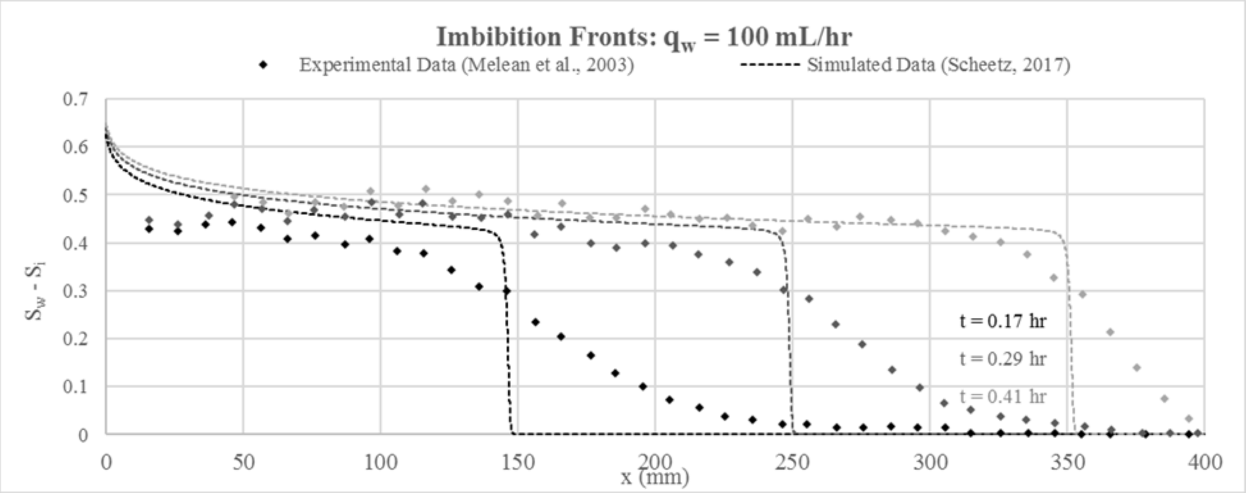


Figure 3.21 New set of input parameters selected to match the experimental data given for an injection rate of 3mL/hr given by Melean et al. in 2003. System is a two-phase incompressible flow system subject to a constant flux boundary of 3 mL/hr.

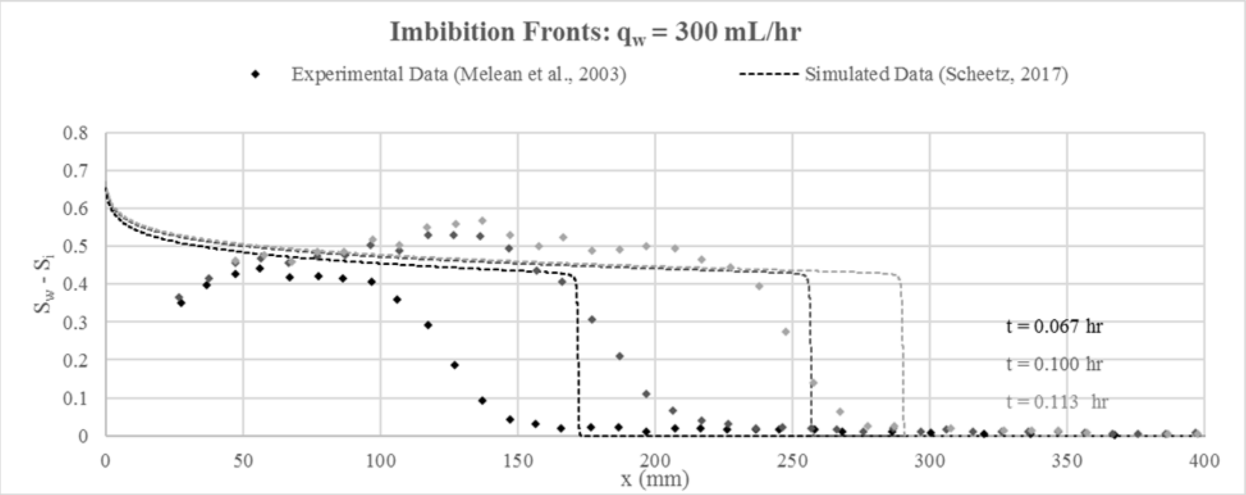
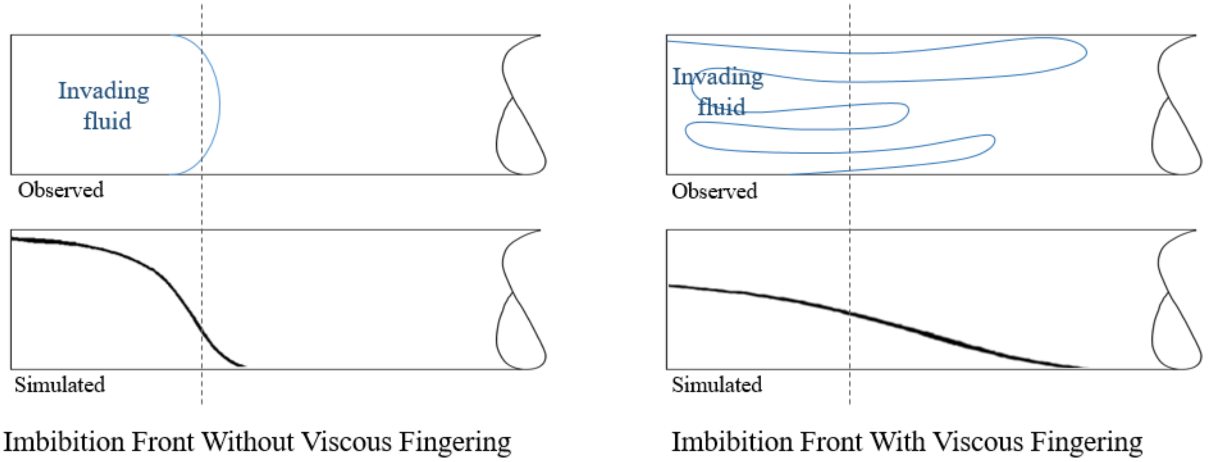


Figure 3.22 New set of input parameters selected to match the experimental data given for an injection rate of 3mL/hr given by Melean et al. in 2003. System is a two-phase incompressible flow system subject to a constant flux boundary of 3 mL/hr.

One possible reason why these imbibition fronts do not match well with higher flow rates could be due to the aforementioned unrealistically low air entry pressure.

Another possible reason why these experimental and simulated imbibition fronts do not match well with higher flow rates could be due to the effects of viscous fingering in the experimental results from Melean and coworkers. Viscous fingering is a phenomenon that occurs when a more viscous fluid (i.e. oil) is displaced by a less viscous fluid (i.e. water). In the case of viscous fingering, the more viscous fluid initially fills the voids in a porous medium is driven forwards by the pressure of the less viscous driving fluid, and an unstable interface may develop between them (Saffman & Taylor, 1958). See Figure 3.23 and Figure 3.24 for reference.



Imbibition Front Without Viscous Fingering

Imbibition Front With Viscous Fingering

Figure 3.23 1-D Effects of Viscous Fingering

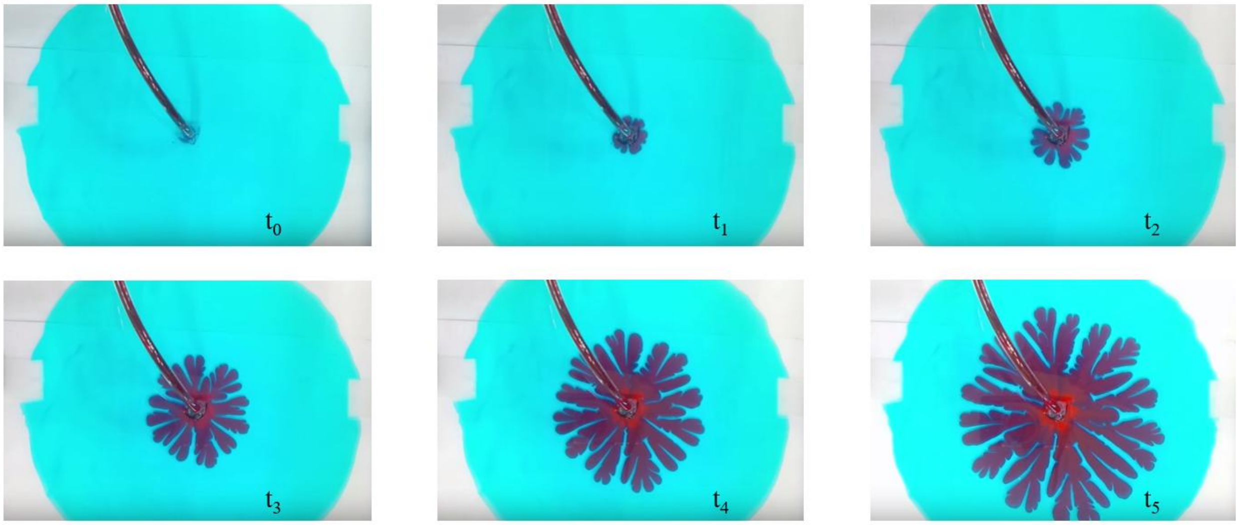


Figure 3.24 2-D effects of viscous fingering in a Hele-Shaw cell (youtube.com, 2013)

3.6.2 Convergence Issues

It should be noted that the convergence of this numerical simulator is extremely sensitive. Relative to those parameters presented in Table 3.3, higher SWRC *N*-values, higher injection rates, and higher entry pressures all require extremely small spatial and temporal step sizes to converge, which greatly increases the computational cost. Though not investigated through this research, it is possible that updating all variables used in the parametric study could result in a more stable and robust simulator.

As discussed in section 2.5, the used of the implicit modified Picard method to approximate the solution should have led to a relatively stable and robust numerical solution. However, this was not the case for this special scenario. At this point, it was suspected that explicit RK4 method could provide equivalent or superior computational performance than implicit Picard scheme in this special case. It was this realization that led researchers to compare this scheme to the explicit RK4 formulation for the compressible solver discussed in Chapter 5.

This page intentionally left blank.

CHAPTER 4

TWO-PHASE HORIZONTAL FLOW SYSTEM OF INCOMPRESSIBLE FLUIDS SUBJECT TO CONSTANT PRESSURE BOUNDARIES

4.1 Overview

4.1.1 Objective

A coupled numerical simulator was developed using MATLAB to model two-phase incompressible flow problems subjected to constant pressure boundary conditions at the inflow and the outflow and varying initial conditions. This solver was validated by analyzing the results obtained for the infiltration variables of interest, specifically the total flux throughout the medium.

4.1.2 Governing Equations

Like the previous derivation in section 3.23.1.2, this derivation models the invasion of a wetting fluid assuming both fluids are incompressible fluids in a porous medium. This difference between this derivation and the previous derivation, is that this derivation models a constant pressure boundary at the inflow. Therefore, the assumption that the total flux q_t is independent of time is no longer valid, and one has to solve two equations simultaneously. For this reason, a coupled solver capable of solving two equations was developed.

The beginning of this derivations is identical to the derivation in section 3.1.2 up until the handling of the mass balance equations for the wetting and nonwetting fluids with Darcy's Law substituted in as shown in equations (4.1) and (4.2)

$$\varphi \frac{\partial S_w}{\partial t} + \frac{\partial}{\partial x}\left[\frac{-\kappa k_{rw}}{\mu_w}\frac{\partial P_w}{\partial x}\right] = 0 \tag{4.1}$$

$$\varphi \frac{\partial S_{nw}}{\partial t} + \frac{\partial}{\partial x}\left[\frac{-\kappa k_{rnw}}{\mu_{nw}}\frac{\partial P_{nw}}{\partial x}\right] = 0 \tag{4.2}$$

For the nonwetting phase equation (4.2), it is convenient to take advantage of the following two relations:

$$S_w + S_{nw} = 1 \tag{4.3}$$

$$P_{nw} = P_c + P_w \tag{4.4}$$

Following this, the saturation and pressure of the nonwetting phase can be eliminated from equation (4.2) as follows:

$$-\varphi \frac{\partial S_w}{\partial t} + \frac{\partial}{\partial x}\left[\frac{-\kappa k_{rnw}}{\mu_{nw}} \frac{\partial (P_c + P_w)}{\partial x}\right] = 0 \tag{4.5}$$

The following substitutions were made to simplify model development:

$$A_w = \frac{\kappa k_{rw}}{\mu_w} \tag{4.6}$$

$$A_{nw} = \frac{\kappa k_{rnw}}{\mu_{nw}} \tag{4.7}$$

Then the governing equations (4.1) and (4.5) can be written as follows:

$$\varphi \frac{\partial S_w}{\partial t} - \frac{\partial}{\partial x}\left[A_w \frac{\partial P_w}{\partial x}\right] = 0 \tag{4.8}$$

$$-\varphi \frac{\partial S_w}{\partial t} - \frac{\partial}{\partial x}\left[A_{nw} \frac{\partial (P_c + P_w)}{\partial x}\right] = 0 \tag{4.9}$$

Equations (4.8) and (4.9) are the two mass balance equations that govern the model. These equations were discretized and approximated using a modified Picard approximation.

4.2 Model Development

4.2.1 Discretization

The governing equations were discretized using a finite difference approximation with the implicit Euler backward algorithm as follows:

$$\varphi\frac{(S_{wj}^{n+1}-S_{wj}^{n})}{\Delta t}-\frac{1}{\Delta x}\left[A_{wj+1/2}^{n+1}\frac{P_{wj+1}^{n+1}-P_{wj}^{n+1}}{\Delta x}-A_{wj-1/2}^{n+1}\frac{P_{wj}^{n+1}-P_{wj-1}^{n+1}}{\Delta x}\right]=0 \tag{4.10}$$

$$-\varphi\frac{(S_{wj}^{n+1}-S_{wj}^{n})}{\Delta t}-\frac{1}{\Delta x}\left[\begin{array}{l} A_{nwj+1/2}^{n+1}\dfrac{P_{cj+1}^{n+1}+P_{wj+1}^{n+1}-P_{cj}^{n+1}-P_{wj}^{n+1}}{\Delta x} \\ -A_{nwj-1/2}^{n+1}\dfrac{P_{cj}^{n+1}+P_{wj}^{n+1}-P_{cj-1}^{n+1}-P_{wj-1}^{n+1}}{\Delta x}\end{array}\right]=0 \tag{4.11}$$

where the superscript *n* represents the current time step and the subscript *j* represents the current node. Subscripts *j+1/2* and *j-1/2* represent half nodes adjacent to the current node *j*.

4.2.2 Modified Picard Iterations

The modified Picard iterations for equations (4.10) and (4.11) are written as follows:

$$\varphi\frac{(S_{wj}^{n+1,m+1}-S_{wj}^{n})}{\Delta t}-\frac{1}{\Delta x}\left[A_{wj+1/2}^{n+1,m}\frac{P_{wj+1}^{n+1,m+1}-P_{wj}^{n+1,m+1}}{\Delta x}-A_{wj-1/2}^{n+1,m}\frac{P_{wj}^{n+1,m+1}-P_{wj-1}^{n+1,m+1}}{\Delta x}\right]=0 \tag{4.12}$$

$$-\varphi\frac{(S_{wj}^{n+1,m+1}-S_{wj}^{n})}{\Delta t}-\frac{1}{\Delta x}\left[\begin{array}{l} A_{nwj+1/2}^{n+1,m}\dfrac{P_{cj+1}^{n+1,m+1}+P_{wj+1}^{n+1,m+1}-P_{cj}^{n+1,m+1}-P_{wj}^{n+1,m+1}}{\Delta x} \\ -A_{nwj-1/2}^{n+1,m}\dfrac{P_{cj}^{n+1,m+1}+P_{wj}^{n+1,m+1}-P_{cj-1}^{n+1,m+1}-P_{wj-1}^{n+1,m+1}}{\Delta x}\end{array}\right]=0 \tag{4.13}$$

where the superscript *m* represents the current iteration. At the *m*th iteration, all values are known, and the values at the *m+1* iteration represent the unknown values being solved for. The capillary and wetting fluid pressure variables at the *m+1* iteration can be approximated by adding an incremental value as follows:

$$P_{j-1}^{n+1,m+1}=P_{j-1}^{n+1,m}+\delta P_{j-1}^{n+1,m+1} \tag{4.14}$$

$$P_{j}^{n+1,m+1}=P_{j}^{n+1,m}+\delta P_{j}^{n+1,m+1} \tag{4.15}$$

$$P_{j+1}^{n+1,m+1}=P_{j+1}^{n+1,m}+\delta P_{j+1}^{n+1,m+1} \tag{4.16}$$

The saturation variable at the *m+1* iteration is approximated as follows:

$$S_{w\,j}^{\;n+1,m+1} = S_{w\,j}^{\;n+1,m} + \left.\frac{\partial S_w}{\partial P_c}\right|^{n+1,m} \delta P_{c\,j}^{\,n+1,m+1} \tag{4.17}$$

where the variable C is defined based on the van Genuchten model of the soil water retention curve as follows:

$$C_j^{n+1,m} = \frac{\partial S_w}{\partial P_c} = (-N+1)\left(\left(\frac{P_c}{p_0}\right)^N + 1\right)^{-2+1/N} \left(\frac{P_c}{p_0}\right)^{N-1} \frac{1}{p_0}\left(1-S_r\right) \tag{4.18}$$

Using the incremental approximations in equations (4.14), (4.15), (4.16) and (4.17), equations (4.12) and (4.13) can be approximated as follows:

$$\begin{aligned} &\varphi\frac{(S_{w\,j}^{\;n+1,m+1} + C_j^{n+1,m}\delta P_{c\,j}^{\,n+1,m+1} - S_{w\,j}^{\;n})}{\Delta t} \\ &-\frac{1}{\Delta x}\begin{bmatrix} A_{w\,j+1/2}^{\;n+1,m}\,\dfrac{P_{w\,j+1}^{\;n+1,m} + \delta P_{w\,j+1}^{\;n+1,m+1} - P_{w\,j}^{\;n+1,m} - \delta P_{w\,j}^{\;n+1,m+1}}{\Delta x} \\ -A_{w\,j-1/2}^{\;n+1,m}\,\dfrac{P_{w\,j}^{\;n+1,m} + \delta P_{w\,j}^{\;n+1,m+1} - P_{w\,j-1}^{\;n+1,m} - \delta P_{w\,j-1}^{\;n+1,m+1}}{\Delta x} \end{bmatrix} = 0 \end{aligned} \tag{4.19}$$

$$\begin{aligned} &-\varphi\frac{(S_{w\,j}^{\;n+1,m} + C_j^{n+1,m}\delta P_{c\,j}^{\,n+1,m+1} - S_{w\,j}^{\;n})}{\Delta t} \\ &-\frac{1}{\Delta x}\begin{bmatrix} A_{nw\,j+1/2}^{\;n+1,m}\,\dfrac{\begin{matrix} P_{c\,j+1}^{\,n+1,m} + \delta P_{c\,j+1}^{\,n+1,m+1} + P_{w\,j+1}^{\;n+1,m} + \delta P_{w\,j+1}^{\;n+1,m+1} \\ -P_{c\,j}^{\,n+1,m} - \delta P_{c\,j}^{\,n+1,m+1} - P_{w\,j}^{\;n+1,m} - \delta P_{w\,j}^{\;n+1,m+1} \end{matrix}}{\Delta x} \\ -A_{nw\,j-1/2}^{\;n+1,m}\,\dfrac{\begin{matrix} P_{c\,j}^{\,n+1,m} + \delta P_{c\,j}^{\,n+1,m+1} + P_{w\,j}^{\;n+1,m} + \delta P_{w\,j}^{\;n+1,m+1} \\ -P_{c\,j-1}^{\,n+1,m} - \delta P_{c\,j-1}^{\,n+1,m+1} - P_{w\,j-1}^{\;n+1,m} - \delta P_{w\,j-1}^{\;n+1,m+1} \end{matrix}}{\Delta x} \end{bmatrix} = 0 \end{aligned} \tag{4.20}$$

with the unknowns being δP_w and δP_c.

4.2.3 Solving for the Variables of Interest

Equations (4.19) and (4.20) are then solved for the unknown increments δP_w and δP_c as follows:

$$
\begin{aligned}
&\overbrace{\varphi\frac{(S_{w\,j}^{\,n+1,m+1}+-S_{w\,j}^{\,n})}{\Delta t}
-\frac{1}{\Delta x}\left[A_{w\,j+1/2}^{\,n+1,m}\frac{P_{w\,j+1}^{\,n+1,m}-P_{w\,j}^{\,n+1,m}}{\Delta x}-A_{w\,j-1/2}^{\,n+1,m}\frac{P_{w\,j}^{\,n+1,m}-P_{w\,j-1}^{\,n+1,m}}{\Delta x}\right]}^{-R_{2\,j-1}}\\
&\overbrace{+\left(\frac{1}{\Delta t}\varphi C_{j}^{n+1,m}\right)}^{\substack{d_{2\,j-1}\\ c}}\boxed{\delta P_{c\,j}^{\,n+1,m+1}}\overbrace{+\left(\frac{1}{\Delta x^{2}}A_{w\,j-1/2}^{\,n+1,m}\right)}^{c_{2\,j-1}}\boxed{\delta P_{w\,j-1}^{\,n+1,m+1}}\\
&\overbrace{-\left(\frac{1}{\Delta x^{2}}A_{w\,j+1/2}^{\,n+1,m}+\frac{1}{\Delta x^{2}}A_{w\,j-1/2}^{\,n+1,m}\right)}^{e_{2\,j-1}}\boxed{\delta P_{w\,j}^{\,n+1,m+1}}\overbrace{+\left(\frac{1}{\Delta x^{2}}A_{w\,j+1/2}^{\,n+1,m}\right)}^{g_{2\,j-1}}\boxed{\delta P_{w\,j+1}^{\,n+1,m+1}}=0
\end{aligned}
\tag{4.21}
$$

$$
\begin{aligned}
&\overbrace{-\varphi\frac{(S_{w\,j}^{\,n+1,m}-S_{w\,j}^{\,n})}{\Delta t}
-\frac{1}{\Delta x}\left[A_{nw\,j+1/2}^{\,n+1,m}\frac{P_{c\,j+1}^{\,n+1,m}+P_{w\,j+1}^{\,n+1,m}-P_{c\,j}^{\,n+1,m}-P_{w\,j}^{\,n+1,m}}{\Delta x}-A_{nw\,j-1/2}^{\,n+1,m}\frac{P_{c\,j}^{\,n+1,m}+P_{w\,j}^{\,n+1,m}-P_{c\,j-1}^{\,n+1,m}-P_{w\,j-1}^{\,n+1,m}}{\Delta x}\right]}^{-R_{2\,j}}\\
&\overbrace{-\left(\frac{1}{\Delta x^{2}}A_{nw\,j-1/2}^{\,n+1,m}\right)}^{a_{2\,j}}\boxed{\delta P_{c\,j-1}^{\,n+1,m+1}}\overbrace{+\left(-\frac{1}{\Delta t}\varphi C_{j}^{n+1,m}+\frac{1}{\Delta x^{2}}A_{nw\,j+1/2}^{\,n+1,m}+\frac{1}{\Delta x^{2}}A_{nw\,j-1/2}^{\,n+1,m}\right)}^{c_{2\,j}}\boxed{\delta P_{c\,j}^{\,n+1,m+1}}\\
&\overbrace{-\left(\frac{1}{\Delta x^{2}}A_{nw\,j+1/2}^{\,n+1,m}\right)}^{e_{2\,j}}\boxed{\delta P_{c\,j+1}^{\,n+1,m+1}}\overbrace{-\left(\frac{1}{\Delta x^{2}}A_{nw\,j-1/2}^{\,n+1,m}\right)}^{b_{2\,j}}\boxed{\delta P_{w\,j-1}^{\,n+1,m+1}}\\
&\overbrace{+\left(\frac{1}{\Delta x^{2}}A_{nw\,j+1/2}^{\,n+1,m}+\frac{1}{\Delta x^{2}}A_{nw\,j-1/2}^{\,n+1,m}\right)}^{d_{2\,j}}\boxed{\delta P_{w\,j}^{\,n+1,m+1}}\overbrace{-\left(\frac{1}{\Delta x^{2}}A_{nw\,j+1/2}^{\,n+1,m}\right)}^{f_{2\,j}}\boxed{\delta P_{w\,j+1}^{\,n+1,m+1}}=0
\end{aligned}
\tag{4.22}
$$

These two equations are then formulated into a matrix to solve for the increments δP_w and δP_c as follows:

$$\begin{bmatrix} d_1 & e_1 & f_1 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_2 & d_2 & e_2 & f_2 & g_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \boxed{c_{2j-2}} & \boxed{d_{2j-1}} & \boxed{e_{2j-1}} & 0 & \boxed{g_{2j-1}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \boxed{a_{2j}} & \boxed{b_{2j}} & \boxed{c_{2j}} & \boxed{d_{2j}} & \boxed{e_{2j}} & \boxed{f_{2j}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{K-1} & b_{K-1} & c_{K-1} & d_{K-1} & e_{K-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_K & b_K & c_K & d_K \end{bmatrix} \begin{pmatrix} \delta P_{c1} \\ \delta P_{w1} \\ \vdots \\ \boxed{\delta P_{c\,j-1}} \\ \boxed{\delta P_{w\,j-1}} \\ \boxed{\delta P_{c\,j}} \\ \boxed{\delta P_{w\,j}} \\ \boxed{\delta P_{c\,j+1}} \\ \boxed{\delta P_{w\,j+1}} \\ \vdots \\ \delta P_{cJ} \\ \delta P_{wJ} \end{pmatrix} = - \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ \vdots \\ \vdots \\ \boxed{R_{2j-1}} \\ \boxed{R_{2j}} \\ \vdots \\ \vdots \\ \vdots \\ R_{K-1} \\ R_K \end{pmatrix} \qquad (4.23)$$

where *R* represents all known values in the equations, and the vectors *a*, *b*, *c*, *d*, *e*, *f*, and *g* represent the known values in front of the variables being solved for, which are the δP_w and δP_c terms boxed in red. Note that the first and last nodes ($j = 1$ and $j = K$) are formulated slightly differently than the rest of the matrix, as they are the nodes that enforce constant flux boundary condition

4.2.4 Constant Pressure Boundary Condition at the Inflow

Based on the input pressure out the inflow, the change in pressure δP_c at the first node is zero. Constant pressure boundary conditions were enforced at the inflow ($j = 1$ at $x = 0$) according to the following governing equations:

$$\boxed{1}\cdot\boxed{\delta P_{c1}} = \boxed{0} \qquad (4.24)$$

$$\boxed{1}\cdot\boxed{\delta P_{w1}} = \boxed{0} \qquad (4.25)$$

These are easily enforced in the solver as follows:

$$\begin{bmatrix} \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{2j-1} & b_{2j-1} & c_{2j-2} & d_{2j-1} & e_{2j-1} & f_{2j-1} & g_{2j-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{2j} & b_{2j} & c_{2j} & d_{2j} & e_{2j} & f_{2j} & g_{2j} & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{K-1} & b_{K-1} & c_{K-1} & d_{K-1} & e_{K-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_K & b_K & c_K & d_K \end{bmatrix} \begin{pmatrix} \boxed{\delta P_{c1}} \\ \boxed{\delta P_{w1}} \\ \vdots \\ \delta P_{c\,j-1} \\ \delta P_{w\,j-1} \\ \delta P_{c\,j} \\ \delta P_{w\,j} \\ \delta P_{c\,j+1} \\ \delta P_{w\,j+1} \\ \vdots \\ \delta P_{cJ} \\ \delta P_{wJ} \end{pmatrix} = - \begin{pmatrix} \boxed{0} \\ \boxed{0} \\ \vdots \\ \vdots \\ \vdots \\ R_{2j-1} \\ R_{2j} \\ \vdots \\ \vdots \\ \vdots \\ R_{K-1} \\ R_K \end{pmatrix} \qquad (4.26)$$

4.2.5 Open-End (Constant Pressure) Boundary Condition at the Outflow

Based on the input pressure out the outflow, the change in pressure δP_c at the J^{th} node is zero. To simulate an open-ended column, a constant pressure boundary condition was enforced at the outflow. At the outflow ($j = K$ at $x = L$), the governing equations are given as follows:

$$\boxed{1}\cdot\boxed{\delta P_{cJ}}=\boxed{0} \qquad (4.27)$$

$$\boxed{1}\cdot\boxed{\delta P_{wJ}}=\boxed{0} \qquad (4.28)$$

These are easily enforced as follows:

$$\begin{bmatrix} d_1 & e_1 & f_1 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_1 & d_1 & e_1 & f_1 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{2j-1} & b_{2j-1} & c_{2j-2} & d_{2j-1} & e_{2j-1} & f_{2j-1} & g_{2j-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{2j} & b_{2j} & c_{2j} & d_{2j} & e_{2j} & f_{2j} & g_{2j} & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} \end{bmatrix} \begin{pmatrix} \delta P_{c1} \\ \delta P_{w1} \\ \vdots \\ \delta P_{c\,j-1} \\ \delta P_{w\,j-1} \\ \delta P_{c\,j} \\ \delta P_{w\,j} \\ \delta P_{c\,j+1} \\ \delta P_{w\,j+1} \\ \vdots \\ \boxed{\delta P_{cJ}} \\ \boxed{\delta P_{wJ}} \end{pmatrix} = - \begin{pmatrix} R_1 \\ R \\ \vdots \\ \vdots \\ \vdots \\ R_{2j-1} \\ R_{2j} \\ \vdots \\ \vdots \\ \vdots \\ \boxed{0} \\ \boxed{0} \end{pmatrix} \qquad (4.29)$$

4.3 Model Validation

Unfortunately, researchers were unable to find any experimental results that could be used to validate this model. Additionally, no analytical solutions exist as Richard's equation assumes constant gas pressure (P_{nw} = *constant*), and McWhorter's solution is only valid for a specific soil water retention curve (McWhorter & Sunada, 1990). Given the lack of experimental data and analytical solutions, the model was validated by looking at the shapes of the curves for the variables of interest and making sure that the shapes look as anticipated given the nature of the system. The simulator was run with the inputs presented in Table 4.1 to perform the validation process.

Table 4.1 Input parameters selected to validate the simulation of the two-phase horizontal flow system of incompressible fluids subjected to constant pressure boundary conditions.

Parameter	Symbol	Value	Unit
SWRC N-Value	N	2.6	-
Air Entry Value	p_0	1200	
Intrinsic Permeability	K_{sat} or κ	1.00E-09	m^2
Wetting Fluid Viscocity	μ_w	8.90E-03	kg/m/s
Nonwetting Fluid Viscocity	μ_{nw}	1.03E-03	kg/m/s
Porosity	φ	35	-
Length	x	50	cm
Residual Saturation	S_r	0.1	-

Table 4.2 Constant pressure boundary conditions used to validate the coupled numerical simulator.

Parameter	Symbol	Value	Unit
Wetting Pressure at the Inflow	$P_{w,left}$	10000	Pa
Nonwetting Pressure at the Inflow	$P_{nw,left}$	10135	Pa
Wetting Pressure at the Outflow	$P_{w,right}$	3000	Pa
Nonwetting Pressure at the outflow	$P_{nw,right}$	10135	Pa

The plots for the wetting front, the pressure of the wetting and nonwetting fluids, the capillary pressure, the flux of the wetting and nonwetting fluids and the total flux are presented in the figures below.

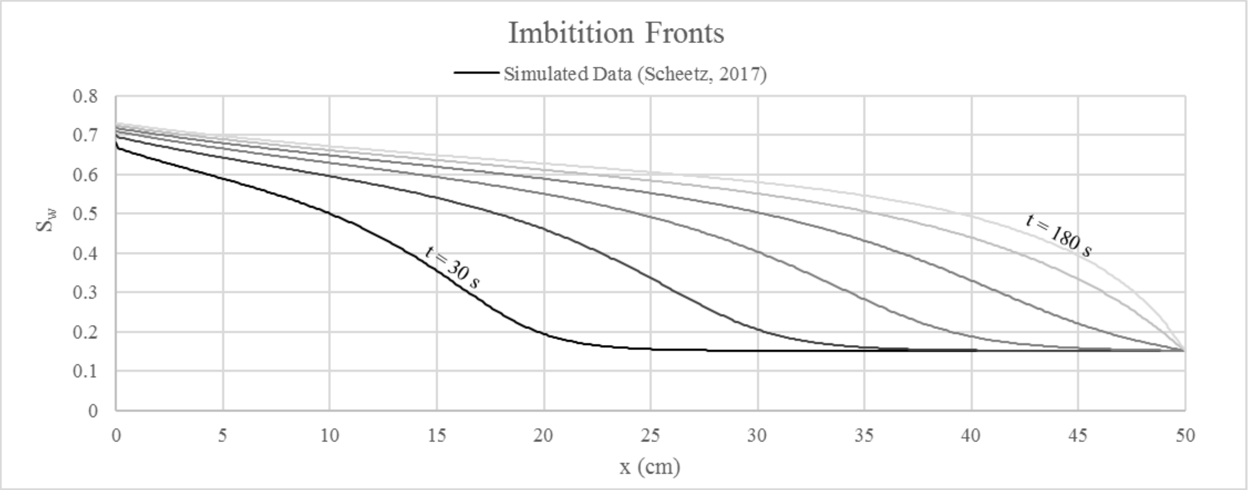


Figure 4.1 Imbibition fronts for a two-phase compressible flow system subject to constant pressure boundary conditions at both ends.

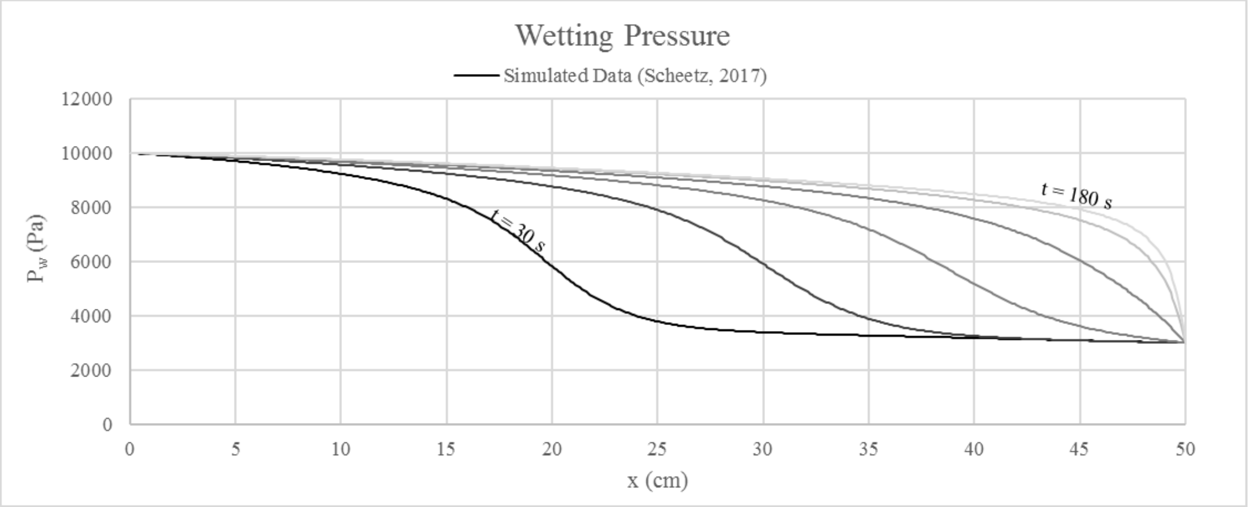


Figure 4.2 Wetting pressure profiles for a two-phase compressible flow system subject to constant pressure boundary conditions at both ends.

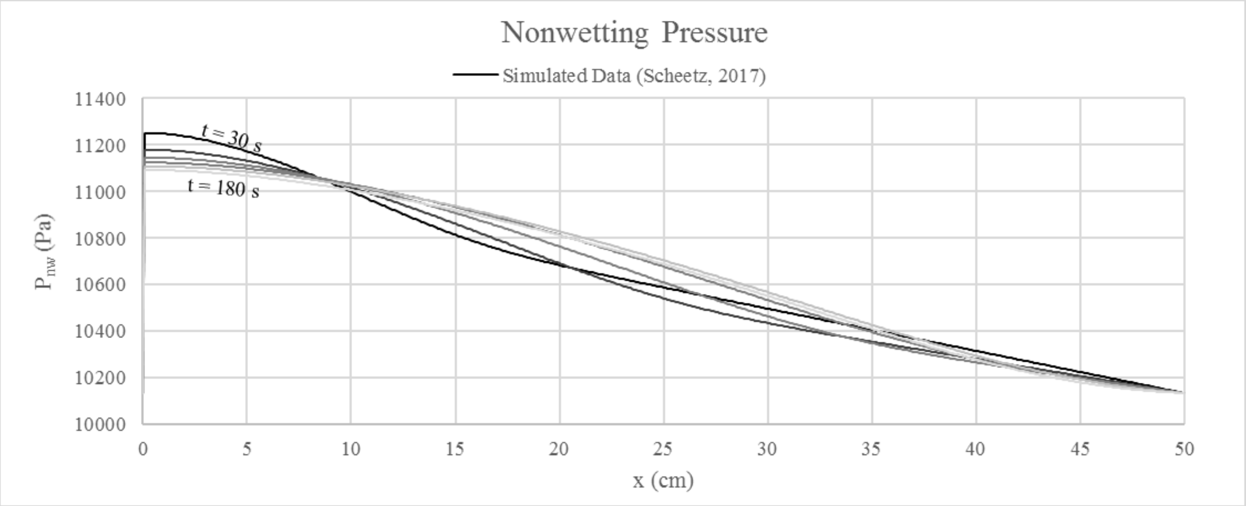


Figure 4.3 Nonwetting pressure profiles for a two-phase compressible flow system subject to constant pressure boundary conditions at both ends.

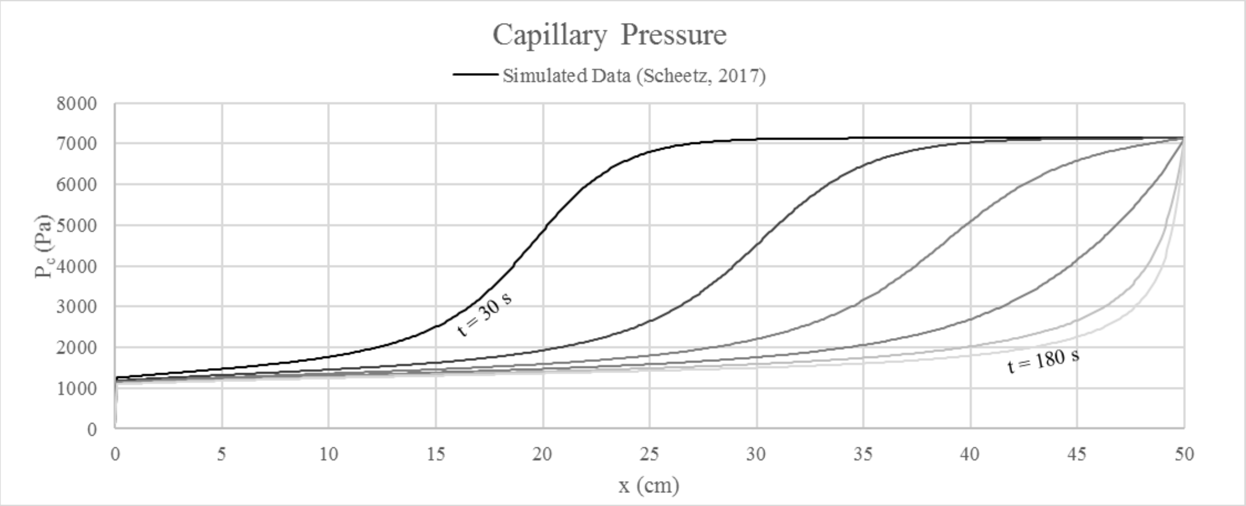


Figure 4.4 Suction profiles for a two-phase compressible flow system subject to constant pressure boundary conditions at both ends.

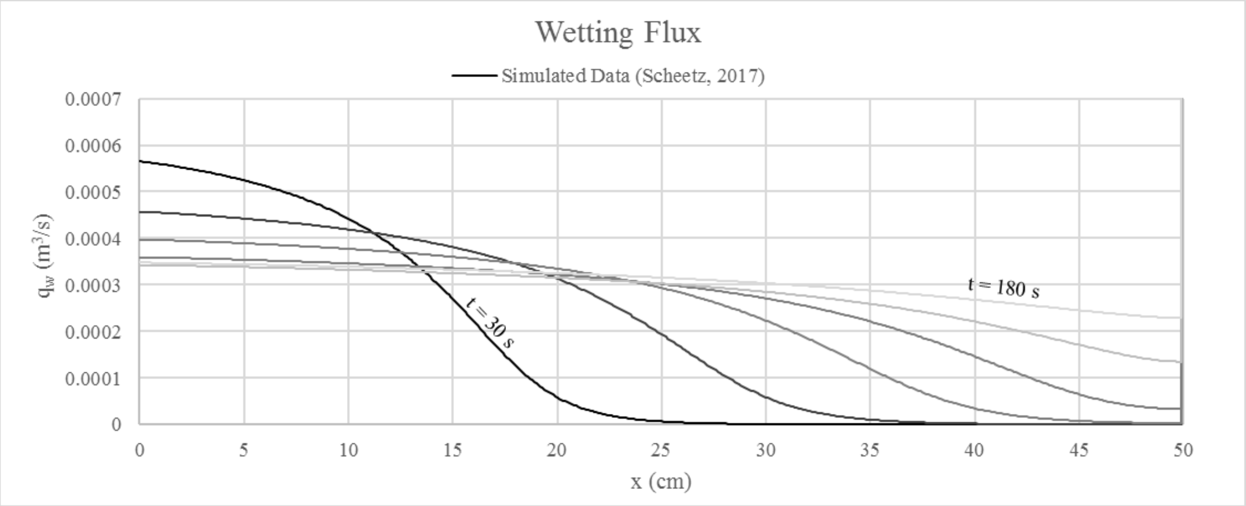


Figure 4.5 Wetting flux profiles for a two-phase compressible flow system subject to constant pressure boundary conditions at both ends.

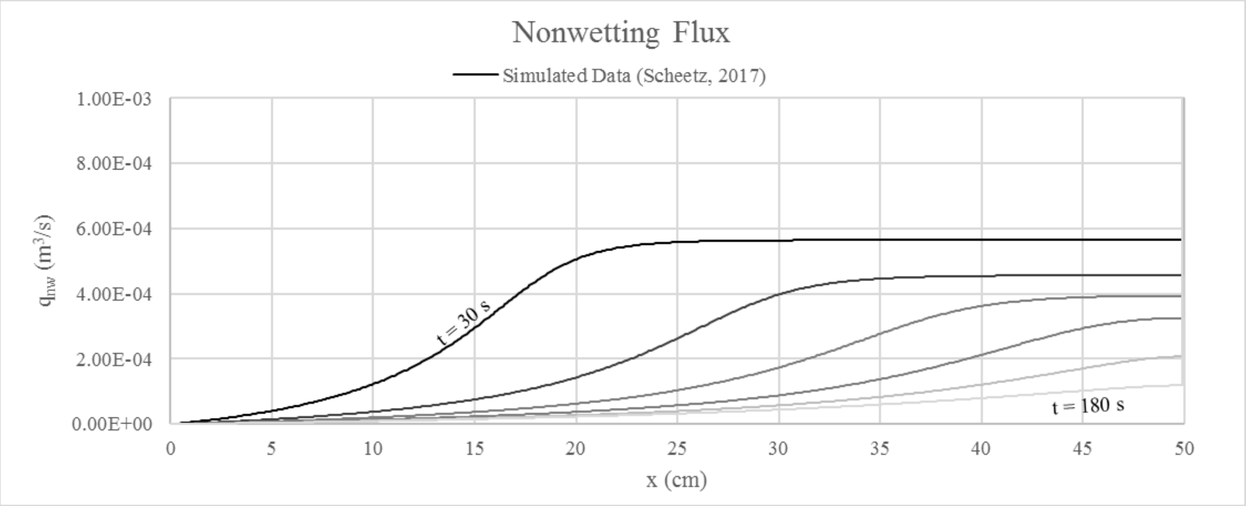


Figure 4.6 Nonwetting flux profiles for a two-phase compressible flow system subject to constant pressure boundary conditions at both ends.

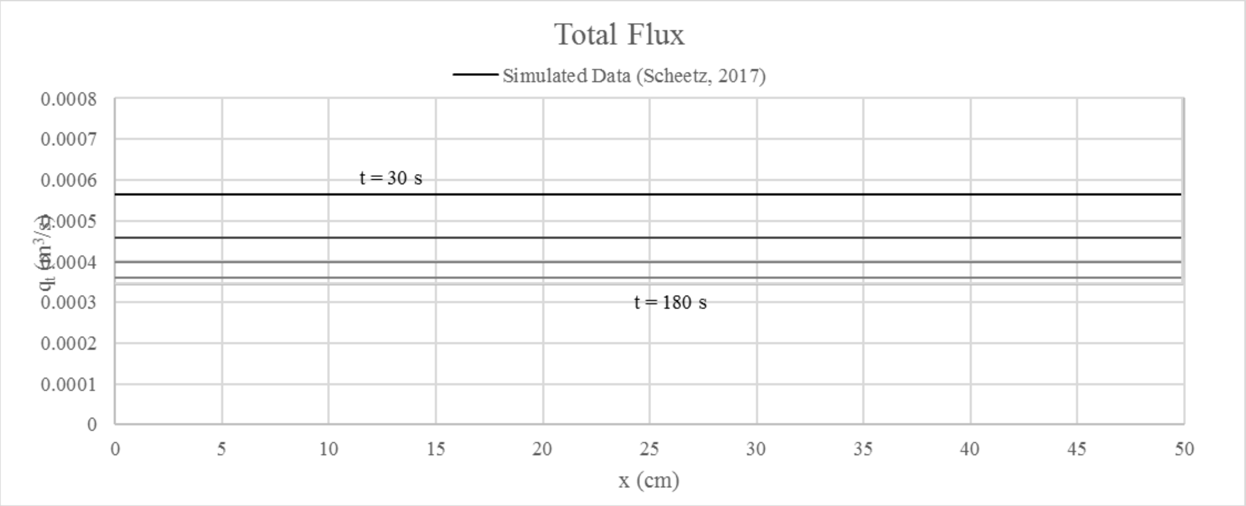


Figure 4.7 Total flux profiles for a two-phase compressible flow system subject to constant pressure boundary conditions at both ends.

In general, the shapes of the curves on all these figures look correct. The input constant pressure boundary conditions enforced cause the invasion of a wetting fluid, and the wetting fronts in Figure 4.2 and the suction profiles in Figure 4.4 indicate that the system is indeed experiencing wetting invasion. In addition to analyzing the saturation profiles, perhaps the model can be at least partially validated by analyzing the total flux plot in Figure 4.7. This plot indicates that the total flux through the medium is a constant value throughout the domain, which is expected for a system of incompressible fluids subjected to constant pressure boundary conditions.

One way that this model could be validated, however, is to change the viscosity of the nonwetting fluid to a very small value and compare the solution with existing solutions to Richard’s equation, or by using the solver developed for Chapter 3. Unfortunately, this step would require significant time, and due to time constraints, researchers deemed it important to move forward with the compressible solver discussed in Chapter 5.

4.4 Discussion

4.4.1 Model Improvement

This model could be improved by implementing the governing equations that model the invasion of a nonwetting fluid, and adding a variable *flag* that allows researchers to switch the boundary conditions at both ends of the sample easily. This would allow researchers to investigate how two incompressible fluids, like oil and water, interact and behave during both wetting and nonwetting invasions with varying boundary conditions. With these alterations, this model could be a useful tool to scientists and engineers in the petroleum industry concerned with enhanced oil recovery, hydraulic fracking or similar problems.

4.4.2 Convergence

It is exciting to note that this code does not experience convergence issues to the same extent as the code developed for Chapter 3. This may be because of the way the coupling equations were formulated and solved, or possibly because realistic values were used for all input parameters. In the next chapter, researchers use the same strategy to solve for compressible two-phase flow problems.

This page intentionally left blank.

CHAPTER 5

TWO-PHASE VERTICAL FLOW SYSTEM OF COMPRESSIBLE FLUIDS SUBJECT TO VARYING BOUNDARY CONDITIONS

5.1 Overview

5.1.1 Objective

Research has shown that the effects of the compressibility of air during water-infiltration of unsaturated soil columns are not negligible (Morel-Seytoux, 1973). The final numerical simulators developed for this project model the two-phase vertical flow of compressible fluids subjected to varying boundary conditions. Comparisons with Touma and Vauclin's 1986 experimental data show that the approximations used in the methods of the solution yield accurate and practical estimations of the infiltration variables of interest.

5.1.2 Governing Equations

Like in the previous derivations, one can start with the mass balance equation written in its most general form as in equation (2.17).

Identically to previous derivations, the fluids are assumed to be in a rigid porous medium so the porosity is considered constant with respect to time and can be taken out from the derivative. The mass balance for the wetting and nonwetting fluids can be defined as follows:

$$\phi \frac{\partial \rho_w S_w}{\partial t} + \frac{\partial \rho_w q_w}{\partial x} = 0 \tag{4.30}$$

$$\phi_0 \frac{\partial \rho_{nw} S_{nw}}{\partial t} + \frac{\partial \rho_{nw} q_{nw}}{\partial x} = 0 \tag{4.31}$$

Such treatment introduces two new unknowns, the density of wetting and nonwetting fluids, in the system, and thus requires two new constitutive equations to close it. These two equations, often referred as the equations of state, give the relationships between the density and pressure:

$$\rho_w = \frac{\rho_{w0}}{\left(1 - C_w\left(P_w - P_{w0}\right)\right)} \tag{4.32}$$

$$\rho_{nw} = \rho_{nw0}\left(\frac{P_{nw}}{P_{nw0}}\right) \tag{4.33}$$

where ρ_{w0} and ρ_{nw0} are the initial densities of the fluids, P_{w0} and P_{nw0} represent the initial pressures of the fluids, and C_w represents the compressibility of water.

Equation (4.32) comes from a simple linear model of water as follows:

$$C_w(P_w - P_{w0}) = V_w - V_{w0} \tag{4.34}$$

where V_w represents the volume of the fluid and the subscript *0* represents the same substance at the initial time and some time $t_0 < t$. Since $V_w = mass / \rho_w$ and the mass of the fluid is constant, equation (4.34) can be rearranged to form equation (4.32).

Equation (4.33) comes from the ideal gas law $PV = nRT$ as follows:

$$\begin{cases} P_0 V_0 = n_0 R T_0 \\ PV = nRT \end{cases} \tag{4.35}$$

where P is the pressure, V is the volume, n is the amount of substance, R is the universal gas constant, T is the temperature, and the subscript *0* represents the same substance at the initial time and some time $t_0 < t$. If the temperature of the gas does not change, the equation (4.35) can be combined as follows:

$$\frac{P_1 V_1}{n_1} = T = \frac{P_0 V_0}{n_0} \tag{4.36}$$

Where *n/V* can be understood as the density of the substance, ρ. Following this, equation (4.36) can be arranged as the equation of state as follows:

$$\rho_1 = \rho_0\left(\frac{P_1}{P_0}\right) \tag{4.37}$$

which is identical to equation (4.33).

The flux of the wetting and nonwetting fluids can be defined as in section 2.4.1 by Darcy’s Law as follows:

$$q_w = \frac{-\kappa k_{rw}(S_w)}{\mu_w}\left(\frac{\partial P_w}{\partial x} - \rho_w G\right) \tag{4.38}$$

$$q_{nw} = \frac{-\kappa k_{rnw}(S_{nw})}{\mu_{nw}}\left(\frac{\partial P_{nw}}{\partial x} - \rho_{nw} G\right) \tag{4.39}$$

Equation (4.30) can be expanded as follows for wetting fluids:

$$\phi_0 \rho_w \frac{\partial S_w}{\partial t} + \phi_0 S_w \overset{\alpha_w(p_w)}{\frac{\partial \rho_w}{\partial P_w}} \frac{\partial) P_w}{\partial t} + \rho_w \frac{\partial q_w}{\partial x} + \overbrace{q_w \frac{\partial \rho_w}{\partial x}}^{\approx 0} = 0 \tag{4.40}$$

The neglection of the fourth term in equation (4.40) follows the assumption that water, the wetting fluid, is only slightly compressible and the spatial variation of its density can be neglected (Celia & Binning, 1992). Substituting the flux in equation (4.38) into equation (4.40) gives the following:

$$\phi_0 \rho_w \frac{\partial S_w}{\partial t} + \phi_0 S_w \alpha_w \frac{\partial P_w}{\partial t} - \rho_w \frac{\partial}{\partial x}\left[\frac{\kappa k_{rw}}{\mu_w}\left(\frac{\partial) P_w}{\partial x} - \rho_w G\right)\right] = 0 \tag{4.41}$$

For the nonwetting phase equation (4.31), it is convenient to take advantage of the following equation:

$$S_w + S_{nw} = 1 \tag{4.42}$$

Following this, the saturation of the nonwetting phase S_{nw} can be eliminated from equation (4.31):

$$\phi_0 \frac{\partial \rho_{nw}\left(1 - S_w\right)}{\partial t} + \frac{\partial \rho_{nw} q_{nw}}{\partial x} = 0 \tag{4.43}$$

which can be further expanded as follows:

$$-\phi_0\rho_{nw}\frac{\partial S_w}{\partial t}+\phi_0\left(1-S_w\right)\overbrace{\frac{\partial\rho_{nw}}{\partial P_{nw}}}^{\alpha_{nw}\left(p_{nw}\right)}\frac{\partial P_{nw}}{\partial t}+\frac{\partial\rho_{nw}q_{nw}}{\partial x}=0 \tag{4.44}$$

Note that the fourth term in equation (4.44) cannot be eliminated as in equation (4.40) since air is significantly more compressible than water. Supplying Darcy's Law for the flux of a nonwetting fluid in equation (4.39) into equation (4.44) gives the following:

$$-\phi_0\rho_{nw}\frac{\partial S_w}{\partial t}+\phi_0\left(1-S_w\right)\alpha_{nw}\frac{\partial P_{nw}}{\partial t}-\frac{\partial}{\partial x}\left[\rho_{nw}\frac{\kappa k_{rnw}}{\mu_{nw}}\left(\frac{\partial P_{nw}}{\partial x}-\rho_{nw}G\right)\right]=0 \tag{4.45}$$

Equations (4.41) and (4.45) are the two equations governing the numerical solver simulating the two-phase vertical flow of two compressible fluids subjected to varying boundary conditions at the inflow. The following substitution were made to simplify model development:

$$B_w=\frac{\kappa k_{rw}}{\mu_w} \tag{4.46}$$

$$B_{nw}=\frac{\kappa k_{rnw}}{\mu_{nw}} \tag{4.47}$$

Then the governing equations can be written as follows:

$$\phi_0\rho_w\frac{\partial S_w}{\partial t}+\phi_0 S_w\alpha_w\frac{\partial P_w}{\partial t}-\rho_w\frac{\partial}{\partial x}\left[B_w\left(\frac{\partial P_w}{\partial x}-\rho_w G\right)\right]=0 \tag{4.48}$$

$$-\phi_0\rho_{nw}\frac{\partial S_w}{\partial t}+\phi_0\left(1-S_w\right)\alpha_{nw}\frac{\partial P_{nw}}{\partial t}-\frac{\partial}{\partial x}\left[\rho_{nw}B_{nw}\left(\frac{\partial P_{nw}}{\partial x}-\rho_{nw}G\right)\right]=0 \tag{4.49}$$

Equations (4.48) and (4.49) are the two mass balance equations that were discretized and approximated using a modified Picard approximation as well as an fourth-order Runge-Kutta method.

5.1.3 Enriching the Soil Water Retention Curve

The boundary conditions at the outflow require a new variable representing the residual saturation of the nonwetting fluid, S_{rnw}. This variable was added to the van Genuchten soil water relationship as follows:

$$S_w = \left(\frac{1}{(1 + P_c / p_0)^N} \right)^M (1 - S_r - S_{rnw}) + S_r \tag{4.50}$$

The implication of adding this variable is that the soil water retention curve does not start from $S_w = 1$ at $P_c = 0$ but rather $S_w = 1 - S_{rnw}$ at $P_c = 0$. This allows the predicted wetting profile to be contained below a certain S_w limit, as observed in experiments. The residual degree of saturation of the nonwetting fluid is discussed further in section 5.5.1.

5.2 Model Development using the Modified Picard Method

5.2.1 Discretization

A finite difference approximation with the implicit Euler backward algorithm was applied to discretize equations (4.48) and (4.49) as follows:

$$\begin{aligned} &\phi_0 \rho_w \frac{S_{wj}^{n+1} - S_{wj}^{n}}{\Delta t} + \phi_0 S_w \alpha_w \frac{P_{wj}^{n+1} - P_{wj}^{n}}{\Delta t} \\ &- \rho_w \frac{1}{\Delta x} \left[\begin{matrix} B_{wj+1/2}^{n} \left(\dfrac{P_{wj+1}^{n+1} - P_{wj}^{n+1}}{\Delta x} - \rho_{wj+1/2}^{n} G \right) \\ - B_{wj-1/2}^{n} \left(\dfrac{P_{wj}^{n+1} - P_{wj-1}^{n+1}}{\Delta x} - \rho_{wj-1/2}^{n} G \right) \end{matrix} \right] = 0 \end{aligned} \tag{4.51}$$

$$-\phi_0\rho_{nw}\frac{S_{wj}^{n+1}-S_{wj}^{n}}{\Delta t}+\phi_0\left(1-S_w\right)\alpha_{nw}\frac{p_{nwj}^{n+1}-p_{nwj}^{n}}{\Delta t}$$
$$-\frac{1}{\Delta x}\begin{bmatrix}\rho_{nwj+1/2}^{n}B_{nwj+1/2}^{n}\left(\dfrac{p_{nwj+1}^{n+1}-p_{nwj}^{n+1}}{\Delta x}-\rho_{nwj+1/2}^{n}G\right)\\ -\rho_{nwj-1/2}^{n}B_{nwj-1/2}^{n}\left(\dfrac{p_{nwj}^{n+1}-p_{nwj-1}^{n+1}}{\Delta x}-\rho_{nwj-1/2}^{n}G\right)\end{bmatrix}=0 \qquad (4.52)$$

where the superscript *n* represents the current time step, subscript *j* represents the current node, and subscripts *j+1/2* and *j-1/2* represent half nodes adjacent to the current node *j*.

5.2.2 *Modified Picard Method*

The modified Picard iteration for equations (4.51) and (4.52) is performed as follows:

$$\phi_0\rho_w\frac{S_{wj}^{n+1,m+1}-S_{wj}^{n}}{\Delta t}+\phi_0 S_w\alpha_w\frac{P_{wj}^{n+1,m+1}-P_{wj}^{n}}{\Delta t}$$
$$-\rho_w\frac{1}{\Delta x}\begin{bmatrix}B_{wj+1/2}^{n+1,m}\left(\dfrac{P_{wj+1}^{n+1,m+1}-P_{wj}^{n+1,m+1}}{\Delta x}-\rho_{wj+1/2}^{n+1,m}G\right)\\ -B_{wj-1/2}^{n+1,m}\left(\dfrac{P_{wj}^{n+1,m+1}-P_{wj-1}^{n+1,m+1}}{\Delta x}-\rho_{wj-1/2}^{n+1,m}G\right)\end{bmatrix}=0 \qquad (4.53)$$

$$-\phi_0\rho_{nw}\frac{S_{wj}^{n+1,m+1}-S_{wj}^{n}}{\Delta t}+\phi_0\left(1-S_w\right)\alpha_{nw}\frac{p_{nwj}^{n+1,m+1}-p_{nwj}^{n}}{\Delta t}$$
$$-\frac{1}{\Delta x}\begin{bmatrix}\rho_{nwj+1/2}^{n+1,m}B_{nwj+1/2}^{n+1,m}\left(\dfrac{p_{nwj+1}^{n+1,m+1}-p_{nwj}^{n+1,m+1}}{\Delta x}-\rho_{nwj+1/2}^{n+1,m}G\right)\\ -\rho_{nwj-1/2}^{n+1,m}B_{nwj-1/2}^{n+1,m}\left(\dfrac{p_{nwj}^{n+1,m+1}-p_{nwj-1}^{n+1,m+1}}{\Delta x}-\rho_{nwj-1/2}^{n+1,m}G\right)\end{bmatrix}=0 \qquad (4.54)$$

where the superscript *m* represents the current iteration. At the *m*th iteration, all values are known, and the values at the *m+1* iteration represent the unknown values being solved for. The pressure variables at the *m+1* iteration are approximated by adding an incremental value as follows:

$$P_{j-1}^{n+1,m+1} = P_{j-1}^{n+1,m} + \delta P_{j-1}^{n+1,m+1} \tag{4.55}$$

$$P_{j}^{n+1,m+1} = P_{j}^{n+1,m} + \delta P_{j}^{n+1,m+1} \tag{4.56}$$

$$P_{j+1}^{n+1,m+1} = P_{j+1}^{n+1,m} + \delta P_{j+1}^{n+1,m+1} \tag{4.57}$$

The saturation variable at the *m+1* iteration is approximated as follows:

$$S_{w\,j}^{\ n+1,m+1} = S_{w\,j}^{\ n+1,m} + \left.\frac{\partial S_w}{\partial P_c}\right|^{n+1,m} \delta P_{c\,j}^{\ n+1,m+1} \tag{4.58}$$

where the variable C is defined based on the van Genuchten model of the soil water retention curve as follows:

$$C_j^{n+1,m} = \frac{\partial S_w}{\partial P_c} = (-N+1)\left(\left(\frac{P_c}{p_0}\right)^N + 1\right)^{-2+1/N} \left(\frac{P_c}{p_0}\right)^{N-1} \frac{1}{p_0}\left(1 - S_r - S_{rnw}\right) \tag{4.59}$$

By substituting equations (4.55), (4.56), (4.57), (4.58) and (4.59) into equations (4.53) and (4.54), the equations to be solved by Picard method can be expressed as follows:

$$\begin{aligned}&\phi_0\rho_w \frac{S_{w\,j}^{\ n+1,m} + C_j^{n+1,m}(\delta P_{nw\,j}^{\ n+1,m+1} - \delta P_{w\,j}^{\ n+1,m+1}) - S_{w\,j}^{\ n}}{\Delta t} + \phi_0 S_w \alpha_w \frac{P_{w\,j}^{\ n+1,m} + \delta P_{w\,j}^{\ n+1,m+1} - P_{w\,j}^{\ n}}{\Delta t} \\ &-\rho_w \frac{1}{\Delta x}\left[\begin{array}{l} B_{w\,j+1/2}^{\ n+1,m}\left(\dfrac{P_{w\,j+1}^{\ n+1,m} + \delta P_{w\,j+1}^{\ n+1,m+1} - P_{w\,j}^{\ n+1,m} - \delta P_{w\,j}^{\ n+1,m+1}}{\Delta x} - \rho_{w\,j+1/2}^{\ n+1,m} G\right) \\ -B_{w\,j-1/2}^{\ n+1,m}\left(\dfrac{P_{w\,j}^{\ n+1,m} + \delta P_{w\,j}^{\ n+1,m+1} - P_{w\,j-1}^{\ n+1,m} - \delta P_{w\,j-1}^{\ n+1,m+1}}{\Delta x} - \rho_{w\,j-1/2}^{\ n+1,m} G\right)\end{array}\right] = 0\end{aligned} \tag{4.60}$$

$$\begin{aligned}&-\phi_0\rho_{nw} \frac{S_{w\,j}^{\ n+1,m} + C_j^{n+1,m}(\delta P_{nw\,j}^{\ n+1,m+1} - \delta P_{w\,j}^{\ n+1,m+1}) - S_{w\,j}^{\ n}}{\Delta t} + \phi_0\left(1 - S_w\right)\alpha_{nw} \frac{P_{nw\,j}^{\ n+1,m} + \delta P_{nw\,j}^{\ n+1,m+1} - P_{nw\,j}^{\ n}}{\Delta t} \\ &-\frac{1}{\Delta x}\left[\begin{array}{l} \rho_{nw\,j+1/2}^{\ n+1,m} B_{nw\,j+1/2}^{\ n+1,m}\left(\dfrac{P_{nw\,j+1}^{\ n+1,m} + \delta P_{nw\,j+1}^{\ n+1,m+1} - P_{nw\,j}^{\ n+1,m} - \delta P_{nw\,j}^{\ n+1,m+1}}{\Delta x} - \rho_{nw\,j+1/2}^{\ n+1,m} G\right) \\ -\rho_{nw\,j-1/2}^{\ n+1,m} B_{nw\,j-1/2}^{\ n+1,m}\left(\dfrac{P_{nw\,j}^{\ n+1,m} + \delta P_{nw\,j}^{\ n+1,m+1} - P_{nw\,j-1}^{\ n+1,m} - + \delta P_{nw\,j-1}^{\ n+1,m+1}}{\Delta x} - \rho_{nw\,j-1/2}^{\ n+1,m} G\right)\end{array}\right] = 0\end{aligned} \tag{4.61}$$

with the unknowns being δP_w and δP_{nw}.

5.2.3 Solving for the variables of interest

Equations (4.60) and (4.61) are then solved for the unknown increments δP_w and δP_{nw} as follows:

$$
\begin{aligned}
&\boxed{\overbrace{\phi\rho_w \frac{S_{w\,j}^{\,n+1,m} - S_{w\,j}^{\,n}}{\Delta t} + \phi_0 S_w \alpha_w \frac{P_{w\,j}^{\,n+1,m} - P_{w\,j}^{\,n}}{\Delta t} - \rho_w \frac{1}{\Delta x}\left[\begin{array}{l} B_{w\,j+1/2}^{\,n+1,m}\left(\dfrac{P_{w\,j+1}^{\,n+1,m} - P_{w\,j}^{\,n+1,m}}{\Delta x} - \rho_{w\,j+1/2}^{\,n+1,m} G\right) \\ -B_{w\,j-1/2}^{\,n+1,m}\left(\dfrac{P_{w\,j}^{\,n+1,m} - P_{w\,j-1}^{\,n+1,m}}{\Delta x} - \rho_{w\,j-1/2}^{\,n+1,m} G\right) \end{array}\right]}^{-R_{2\,j-1}}} \\
&\boxed{\overbrace{+\left(\frac{1}{\Delta t}\phi\rho_w C_j^{n+1,m}\right)}^{c_{2\,j-1}}}\boxed{\delta P_{nw\,j}^{\,n+1,m+1}}\boxed{\overbrace{-\left(\frac{1}{\Delta x^2}\rho_w B_{w\,j-1/2}^{\,n+1,m}\right)}^{b_{2\,j-1}}}\boxed{\delta P_{w\,j-1}^{\,n+1,m+1}}\boxed{\overbrace{+\left(\begin{array}{l} -\phi\rho_w \dfrac{C_j^{n+1,m}}{\Delta t} + \dfrac{1}{\Delta t}\phi_0 S_w \alpha_w \\ +\dfrac{1}{\Delta x^2}\rho_w\left(B_{w\,j+1/2}^{\,n,m} + B_{w\,j-1/2}^{\,n,m}\right) \end{array}\right)}^{d_{2\,j-1}}}\boxed{\delta P_{w\,j}^{\,n+1,m+1}} \\
&\boxed{\overbrace{-\left(\frac{1}{\Delta x^2}\rho_w B_{w\,j+1/2}^{\,n+1,m}\right)}^{f_{2\,j-1}}}\boxed{\delta P_{w\,j+1}^{\,n+1,m+1}} = 0
\end{aligned}
\tag{4.62}
$$

$$
\begin{aligned}
&\boxed{\overbrace{\begin{array}{l} -\phi\rho_{nw} \dfrac{S_{w\,j}^{\,n+1,m} - S_{w\,j}^{\,n}}{\Delta t} + \phi_0\left(1 - S_w\right)\alpha_{nw} \dfrac{P_{nw\,j}^{\,n+1,m} - P_{nw\,j}^{\,n}}{\Delta t} \\ -\dfrac{1}{\Delta x}\left[\begin{array}{l} \rho_{nw\,j+1/2}^{\,n+1,m} B_{nw\,j+1/2}^{\,n+1,m}\left(\dfrac{P_{nw\,j+1}^{\,n+1,m} - P_{nw\,j}^{\,n+1,m}}{\Delta x} - \rho_{nw\,j+1/2}^{\,n+1,m} G\right) \\ -\rho_{nw\,j-1/2}^{\,n+1,m} B_{nw\,j-1/2}^{\,n+1,m}\left(\dfrac{P_{nw\,j}^{\,n+1,m} - P_{nw\,j-1}^{\,n+1,m}}{\Delta x} - \rho_{nw\,j-1/2}^{\,n+1,m} G\right) \end{array}\right] \end{array}}^{-R_{2\,j}}} \\
&\boxed{\overbrace{+\left(\frac{1}{\Delta t}\phi\rho_{nw} C_j^{n+1,m}\right)}^{c_{2\,j}}}\boxed{\delta P_{w\,j}^{\,n+1,m+1}}\boxed{\overbrace{-\left(\frac{1}{\Delta x^2}\rho_{nw\,j-1/2}^{\,n+1,m} B_{nw\,j-1/2}^{\,n+1,m}\right)}^{b_{2\,j}}}\boxed{\delta P_{nw\,j-1}^{\,n+1,m+1}} \\
&\boxed{\overbrace{+\left(\begin{array}{l} -\dfrac{1}{\Delta t}\phi\rho_{nw} C_j^{n+1,m} + \dfrac{1}{\Delta t}\phi\left(1 - S_w\right)\alpha_{nw} + \\ \dfrac{1}{\Delta x^2}\left(\rho_{nw\,j+1/2}^{\,n+1,m} B_{nw\,j+1/2}^{\,n+1,m} + \rho_{nw\,j-1/2}^{\,n+1,m} B_{nw\,j-1/2}^{\,n+1,m}\right) \end{array}\right)}^{d_{2\,j}}}\boxed{\delta P_{nw\,j}^{\,n+1,m+1}} \\
&\boxed{\overbrace{-\left(\frac{1}{\Delta x^2}\rho_{nw\,j+1/2}^{\,n+1,m} B_{nw\,j+1/2}^{\,n+1,m}\right)}^{f_{2\,j}}}\boxed{\delta P_{nw\,j+1}^{\,n+1,m+1}} = 0
\end{aligned}
\tag{4.63}
$$

These two equations are then formulated into a matrix to solve for the increments δP_w and δP_{nw} at the j^{th} node simultaneously as follows:

$$\begin{bmatrix} d_1 & e_1 & f_1 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_2 & d_2 & e_2 & f_2 & g_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{2j-1} & \boxed{b_{2j-1}} & \boxed{c_{2j-2}} & \boxed{d_{2j-1}} & 0 & \boxed{f_{2j-1}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{2j} & \boxed{b_{2j}} & \boxed{c_{2j}} & \boxed{d_{2j}} & 0 & \boxed{f_{2j}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{K-1} & b_{K-1} & c_{K-1} & d_{K-1} & e_{K-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_K & b_K & c_K & d_K \end{bmatrix} \begin{pmatrix} \delta P_{w1} \\ \delta P_{nw1} \\ \vdots \\ \boxed{\delta P_{w\,j-1}} \\ \boxed{\delta P_{nw\,j-1}} \\ \boxed{\delta P_{w\,j}} \\ \boxed{\delta P_{nw\,j}} \\ \boxed{\delta P_{w\,j+1}} \\ \boxed{\delta P_{nw\,j+1}} \\ \vdots \\ \delta P_{wJ} \\ \delta P_{nwJ} \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ \vdots \\ \vdots \\ \boxed{R_{2j-1}} \\ \boxed{R_{2j}} \\ \vdots \\ \vdots \\ \vdots \\ R_{K-1} \\ R_K \end{pmatrix} \tag{4.64}$$

where R represents all known values in the equations, and the vectors a, b, c, d, e, f, and g represent the known values in front of the variables being solved for, which are the δP_w and δP_{nw} terms boxed in red. Note that the first and last nodes ($j = 1$ and $j = K$) are formulated slightly differently than the rest of the matrix, as they are the nodes that enforce constant flux boundary condition

5.2.4 *Constant Pressure Boundary Condition at the Inflow*

This solver is capable of switching between boundary conditions by switching the flag. The first boundary condition implemented in the solver was the constant pressure boundary conditions (open ends) at both ends of the porous media.

At the inflow ($j = 1$ at $x = 0$), the governing equations are given as follows:

$$\boxed{1} \cdot \boxed{\delta P_{w1}} = \boxed{0} \tag{4.65}$$

$$\boxed{1} \cdot \boxed{\delta P_{nw1}} = \boxed{0} \tag{4.66}$$

These are easily enforced in the solver as follows:

$$\begin{bmatrix} \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{2j-1} & b_{2j-1} & c_{2j-2} & d_{2j-1} & e_{2j-1} & f_{2j-1} & g_{2j-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{2j} & b_{2j} & c_{2j} & d_{2j} & e_{2j} & f_{2j} & g_{2j} & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{K-1} & b_{K-1} & c_{K-1} & d_{K-1} & e_{K-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_K & b_K & c_K & d_K \end{bmatrix} \begin{pmatrix} \boxed{\delta P_{w1}} \\ \boxed{\delta P_{nw1}} \\ \vdots \\ \delta P_{w\,j-1} \\ \delta P_{nw\,j-1} \\ \delta P_{w\,j} \\ \delta P_{nw\,j} \\ \delta P_{w\,j+1} \\ \delta P_{nw\,j+1} \\ \vdots \\ \delta P_{wJ} \\ \delta P_{nwJ} \end{pmatrix} = - \begin{pmatrix} \boxed{0} \\ \boxed{0} \\ \vdots \\ \vdots \\ \vdots \\ R_{2j-1} \\ R_{2j} \\ \vdots \\ \vdots \\ \vdots \\ R_{K-1} \\ R_K \end{pmatrix} \qquad (4.67)$$

5.2.5 *Constant Flux Boundary Condition at the Inflow*

The second type of boundary condition enforced is a constant flux boundary for the wetting and nonwetting fluids at the inflow (*j = 1* at *x = 0*). The governing equations are given as follows:

$$q_{w3/2}^{n+1,m} = -B_{w3/2}^{n+1,m}\left(\frac{P_{w2}^{n+1,m} + \delta P_{w2}^{n+1,m+1} - P_{w1}^{n+1,m} - \delta P_{w1}^{n+1,m+1}}{\Delta x} - \rho_{w3/2}^{n+1,m} G\right) \qquad (4.68)$$

$$q_{nw3/2}^{n+1,m} = -B_{nw3/2}^{n+1,m}\left(\frac{P_{nw2}^{n+1,m} + \delta P_{nw2}^{n+1,m+1} - P_{nw1}^{n+1,m} - \delta P_{nw1}^{n+1,m+1}}{\Delta x} - \rho_{nw3/2}^{n+1,m} G\right) \qquad (4.69)$$

Equation (4.68) and (4.69) can be rearranged to solve for the unknowns as follows:

$$\boxed{\overbrace{q_{w3/2}^{n+1,m} + B_{w3/2}^{n+1,m}\left(\frac{P_{w2}^{n+1,m} - P_{w1}^{n+1,m}}{\Delta x} - \rho_{w3/2}^{n+1,m} G\right)}^{-R_1}} \boxed{\overbrace{-B_{w3/2}^{n+1,m}\frac{1}{\Delta x}}^{d_1}} \boxed{\delta P_{w1}^{n+1,m+1}} \boxed{\overbrace{+B_{w3/2}^{n+1,m}\frac{1}{\Delta x}}^{f_1}} \boxed{\delta P_{w2}^{n+1,m+1}} = 0 \qquad (4.70)$$

$$\boxed{\overbrace{q_{nw3/2}^{n+1,m} + B_{nw3/2}^{n+1,m}\left(\frac{P_{nw2}^{n+1,m} - P_{nw1}^{n+1,m}}{\Delta x} - \rho_{nw3/2}^{n+1,m} G\right)}^{-R_2}} \boxed{\overbrace{-B_{nw3/2}^{n+1,m}\frac{1}{\Delta x}}^{d_2}} \boxed{\delta P_{nw1}^{n+1,m+1}} \boxed{\overbrace{+B_{nw3/2}^{n+1,m}\frac{1}{\Delta x}}^{f_2}} \boxed{\delta P_{nw2}^{n+1,m+1}} = 0 \qquad (4.71)$$

Then, the boundary conditions at the inflow are enforced as follows:

$$\begin{bmatrix} \boxed{d_1} & 0 & \boxed{f_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \boxed{d_2} & 0 & \boxed{f_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{2j-1} & b_{2j-1} & c_{2j-2} & d_{2j-1} & e_{2j-1} & f_{2j-1} & g_{2j-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{2j} & b_{2j} & c_{2j} & d_{2j} & e_{2j} & f_{2j} & g_{2j} & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{K-1} & b_{K-1} & c_{K-1} & d_{K-1} & e_{K-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_K & b_K & c_K & d_K \end{bmatrix} \begin{pmatrix} \boxed{\delta P_{w1}} \\ \boxed{\delta P_{nw1}} \\ \boxed{\delta P_{w2}} \\ \boxed{\delta P_{nw2}} \\ \vdots \\ \delta P_{w\,j-1} \\ \delta P_{nw\,j-1} \\ \delta P_{w\,j} \\ \delta P_{nw\,j} \\ \delta P_{w\,j+1} \\ \delta P_{nw\,j+1} \\ \vdots \\ \delta P_{wJ} \\ \delta P_{nwJ} \end{pmatrix} = - \begin{pmatrix} \boxed{R_1} \\ \boxed{R_2} \\ \vdots \\ \vdots \\ \vdots \\ R_{2j-1} \\ R_{2j} \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ 0 \end{pmatrix} \qquad (4.72)$$

5.2.6 *Open-End Boundary Condition at the Outflow*

To simulate an open-ended column, a constant pressure boundary condition was used. At the outflow ($j = K$ at $x = L$), the governing equations are given as follows:

$$\boxed{1}\cdot\boxed{\delta P_{wJ}} = \boxed{0} \qquad (4.73)$$

$$\boxed{1}\cdot\boxed{\delta P_{nwJ}} = \boxed{0} \qquad (4.74)$$

These are easily enforced as follows:

$$\begin{bmatrix} d_1 & e_1 & f_1 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_2 & d_1 & e_2 & f_2 & g_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{2j-1} & b_{2j-1} & c_{2j-2} & d_{2j-1} & e_{2j-1} & f_{2j-1} & g_{2j-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{2j} & b_{2j} & c_{2j} & d_{2j} & e_{2j} & f_{2j} & g_{2j} & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} \end{bmatrix} \begin{pmatrix} \delta P_{w1} \\ \delta P_{nw1} \\ \vdots \\ \delta P_{w\,j-1} \\ \delta P_{nw\,j-1} \\ \delta P_{w\,j} \\ \delta P_{nw\,j} \\ \delta P_{w\,j+1} \\ \delta P_{nw\,j+1} \\ \vdots \\ \boxed{\delta P_{wJ}} \\ \boxed{\delta P_{nwJ}} \end{pmatrix} = - \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ \vdots \\ \vdots \\ R_{2j-1} \\ R_{2j} \\ \vdots \\ \vdots \\ \vdots \\ \boxed{0} \\ \boxed{0} \end{pmatrix} \qquad (4.75)$$

5.2.7 Closed-End Boundary Condition at the Outflow

To simulate a closed-end boundary condition at the outflow, constant flux conditions were enforced for both the wetting and nonwetting fluids, $q_{nwJ} = 0$ and $q_{nwJ} = 0$, at the outflow. The governing equations for the wetting and nonwetting fluids are given as follows:

$$q_{wJ-1/2}^{n+1,m} = -B_{wJ-1/2}^{n+1,m}\left(\frac{P_{wJ}^{n+1,m} + \delta P_{wJ}^{n+1,m+1} - P_{wJ-1}^{n+1,m} - \delta P_{wJ-1}^{n+1,m+1}}{\Delta x} - \rho_{wJ-1/2}^{n+1,m} G\right) \tag{4.76}$$

$$q_{nwJ-1/2}^{n+1,m} = -B_{nwJ-1/2}^{n+1,m}\left(\frac{P_{nwJ}^{n+1,m} + \delta P_{nwJ}^{n+1,m+1} - P_{nwJ-1}^{n+1,m} - \delta P_{nwJ-1}^{n+1,m+1}}{\Delta x} - \rho_{nwJ-1/2}^{n+1,m} G\right) \tag{4.77}$$

Equations (4.76) and (4.77) can be rearranged as follows:

$$\boxed{\overbrace{q_{wJ-1/2}^{n+1,m} + B_{wJ-1/2}^{n+1,m}\left(\frac{P_{wJ}^{n+1,m} - P_{wJ-1}^{n+1,m}}{\Delta x} - \rho_{wJ-1/2}^{n+1,m} G\right)}^{-R_{K-1}}}\boxed{\overbrace{-B_{wJ-1/2}^{n+1,m}\frac{1}{\Delta x}}^{d_{K-1}}}\boxed{\delta P_{wJ-1}^{n+1,m+1}}\boxed{\overbrace{+B_{wJ-1/2}^{n+1,m}\frac{1}{\Delta x}}^{f_{K-1}}}\boxed{\delta P_{wJ}^{n+1,m+1}} = 0 \tag{4.78}$$

$$\boxed{\overbrace{q_{nwJ-1/2}^{n+1,m} + B_{nwJ-1/2}^{n+1,m}\left(\frac{P_{nwJ}^{n+1,m} - P_{nwJ-1}^{n+1,m}}{\Delta x} - \rho_{nwJ-1/2}^{n+1,m} G\right)}^{-R_{K}}}\boxed{\overbrace{-B_{nwJ-1/2}^{n+1,m}\frac{1}{\Delta x}}^{d_{K}}}\boxed{\delta P_{nwJ-1}^{n+1,m+1}}\boxed{\overbrace{+B_{nwJ-1/2}^{n+1,m}\frac{1}{\Delta x}}^{f_{K}}}\boxed{\delta P_{nwJ}^{n+1,m+1}} = 0 \tag{4.79}$$

These two boundary conditions are implemented in the model as follows:

$$\begin{bmatrix} d_1 & e_1 & f_1 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_2 & d_1 & e_2 & f_2 & g_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{2j-1} & b_{2j-1} & c_{2j-2} & d_{2j-1} & e_{2j-1} & f_{2j-1} & g_{2j-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{2j} & b_{2j} & c_{2j} & d_{2j} & e_{2j} & f_{2j} & g_{2j} & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{b_{K-1}} & 0 & \boxed{d_{K-1}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{b_K} & 0 & \boxed{d_K} \end{bmatrix} \begin{pmatrix} \delta P_{w1} \\ \delta P_{nw1} \\ \vdots \\ \delta P_{wj-1} \\ \delta P_{nwj-1} \\ \delta P_{wj} \\ \delta P_{nwj} \\ \delta P_{wj+1} \\ \delta P_{nwj+1} \\ \vdots \\ \boxed{\delta P_{wJ-1}} \\ \boxed{\delta P_{nwJ-1}} \\ \boxed{\delta P_{wJ}} \\ \boxed{\delta P_{nwJ}} \end{pmatrix} = - \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ \vdots \\ \vdots \\ R_{2j-1} \\ R_{2j} \\ \vdots \\ \vdots \\ \vdots \\ \boxed{R_{K-1}} \\ \boxed{R_K} \end{pmatrix} \tag{4.80}$$

5.3 Model Development using the Fourth-Order Runge-Kutta Method

In an attempt to validate the model built using the modified Picard method, another solver was implemented using the error controlled fourth-order Runge-Kutta method (RK4). For this solver, the same two governing equations were discretized and approximated.

5.3.1 Discretization

In the RK4 method, discretization was performed using the explicit Euler Forward finite difference approximation. Equations (4.48) and (4.49) are discretized as follows:

$$\begin{aligned}&\phi_0 \rho_{wj}^{\,n} C_j^n \frac{\Delta p_{cj}^{\,n+1}}{\Delta t} + \phi_0 S_{wj}^{\,n} \alpha_{wj}^{\,n} \frac{\Delta p_{wj}^{\,n+1}}{\Delta t}\\ &-\rho_{wj}^{\,n}\frac{1}{\Delta x}\left[B_{wj+1/2}^{\,n}\left(\frac{p_{wj+1}^{\,n}-p_{wj}^{\,n}}{\Delta x} - \rho_{wj+1/2}^{\,n}G\right) - B_{wj-1/2}^{\,n}\left(\frac{p_{wj}^{\,n}-p_{wj-1}^{\,n}}{\Delta x}-\rho_{wj-1/2}^{\,n}G\right)\right]=0\end{aligned} \tag{4.81}$$

$$\begin{aligned}&-\phi_0 \rho_{nw,j}^{\,n} C_j^n \frac{\Delta p_{cj}^{\,n+1}}{\Delta t} + \phi_0\left(1-S_{wj}^{\,n}\right) \alpha_{nwj}^{\,n} \frac{\Delta p_{nw}^{\,n+1}}{\Delta t}\\ &-\frac{1}{\Delta x}\left[\rho_{nwj+1/2}^{\,n}B_{nwj+1/2}^{\,n}\left(\frac{p_{nwj+1}^{\,n}-p_{nwj}^{\,n}}{\Delta x} - \rho_{nwj+1/2}^{\,n}G\right) - \rho_{nwj-1/2}^{\,n}B_{nwj-1/2}^{\,n}\left(\frac{p_{nwj}^{\,n}-p_{nwj-1}^{\,n}}{\Delta x}-\rho_{nwj-1/2}^{\,n}G\right)\right]=0\end{aligned} \tag{4.82}$$

Since

$$\frac{\Delta P_c}{\Delta t} = \frac{\Delta P_{nw}}{\Delta t} - \frac{\Delta P_w}{\Delta t} \tag{4.83}$$

Equations (4.81) and (4.82) can be rewritten in terms of $\frac{\Delta P_{nw}}{\Delta t}$ and $\frac{\Delta P_w}{\Delta t}$ as follows:

$$\begin{aligned}&\phi_0 \rho_{wj}^{\,n} C_j^n \frac{\Delta p_{nwj}^{\,n+1}}{\Delta t} + \left(\phi_0 S_{wj}^{\,n} \alpha_{wj}^{\,n} - \phi_0 \rho_{wj}^{\,n} C_j^n\right) \frac{\Delta p_{wj}^{\,n+1}}{\Delta t}\\ &-\rho_{wj}^{\,n}\frac{1}{\Delta x}\left[B_{wj+1/2}^{\,n}\left(\frac{p_{wj+1}^{\,n}-p_{wj}^{\,n}}{\Delta x} - \rho_{wj+1/2}^{\,n}G\right) - B_{wj-1/2}^{\,n}\left(\frac{p_{wj}^{\,n}-p_{wj-1}^{\,n}}{\Delta x}-\rho_{wj-1/2}^{\,n}G\right)\right]=0\end{aligned} \tag{4.84}$$

$$\begin{aligned}&\phi_0 \rho_{nwj}^{\,n} C_j^n \frac{\Delta p_{wj}^{\,n+1}}{\Delta t} + \left(\phi_0\left(1-S_{wj}^{\,n}\right) \alpha_{nwj}^{\,n} - \phi_0 \rho_{nwj}^{\,n} C_j^n\right) \frac{\Delta p_{nw}^{\,n+1}}{\Delta t}\\ &-\frac{1}{\Delta x}\left[\rho_{nwj+1/2}^{\,n}B_{nwj+1/2}^{\,n}\left(\frac{p_{nwj+1}^{\,n}-p_{nwj}^{\,n}}{\Delta x} - \rho_{nwj+1/2}^{\,n}G\right) - \rho_{nwj-1/2}^{\,n}B_{nwj-1/2}^{\,n}\left(\frac{p_{nwj}^{\,n}-p_{nwj-1}^{\,n}}{\Delta x}-\rho_{nwj-1/2}^{\,n}G\right)\right]=0\end{aligned} \tag{4.85}$$

5.3.2 Solving for the Variables of Interest

To implement the RK4 method, equations (4.84) and (4.85) are rearranged to solve for the unknowns $\frac{\Delta P_{nw}}{\Delta t}$ and $\frac{\Delta P_{w}}{\Delta t}$ as follows:

$$\boxed{\overbrace{\left(\phi_0 S_{w\,j}^{\;n}\alpha_{w\,j}^{\;n} - \phi_0\rho_{w\,j}^{\;n}C_j^n\right)}^{m_{11}}}\boxed{\frac{\Delta p_{w\,j}^{\;n+1}}{\Delta t}}\boxed{+\overbrace{\phi_0\rho_{w\,j}^{\;n}C_j^n}^{m_{12}}}\boxed{\frac{\Delta p_{nw\,j}^{\;n+1}}{\Delta t}}$$
$$\boxed{=\overbrace{\rho_{w\,j}^{\;n}\frac{1}{\Delta x}\left[B_{w\,j+1/2}^{\;n}\left(\frac{p_{w\,j+1}^{\;n}-p_{w\,j}^{\;n}}{\Delta x}-\rho_{w\,j+1/2}^{\;n}G\right)-B_{w\,j-1/2}^{\;n}\left(\frac{p_{w\,j}^{\;n}-p_{w\,j-1}^{\;n}}{\Delta x}-\rho_{w\,j-1/2}^{\;n}G\right)\right]}^{R_1}} \tag{4.86}$$

$$\boxed{\overbrace{\phi_0\rho_{nw\,j}^{\;n}C_j^n}^{m_{21}}}\boxed{\frac{\Delta p_{w\,j}^{\;n+1}}{\Delta t}}\boxed{+\overbrace{\left(\phi_0\left(1-S_{w\,j}^{\;n}\right)\alpha_{nw\,j}^{\;n}-\phi_0\rho_{nw\,j}^{\;n}C_j^n\right)}^{m_{22}}}\boxed{\frac{\Delta p_{nw}^{\;n+1}}{\Delta t}}$$
$$\boxed{=\overbrace{\frac{1}{\Delta x}\left[\begin{array}{l}\rho_{nw\,j+1/2}^{\;n}B_{nw\,j+1/2}^{\;n}\left(\frac{p_{nw\,j+1}^{\;n}-p_{nw\,j}^{\;n}}{\Delta x}-\rho_{nw\,j+1/2}^{\;n}G\right)\\ -\rho_{nw\,j-1/2}^{\;n}B_{nw\,j-1/2}^{\;n}\left(\frac{p_{nw\,j}^{\;n}-p_{nw\,j-1}^{\;n}}{\Delta x}-\rho_{nw\,j-1/2}^{\;n}G\right)\end{array}\right]}^{R_2}} \tag{4.87}$$

From here, one needs to solve a linear system:

$$\begin{bmatrix}\boxed{m_{11}} & \boxed{m_{12}}\\ \boxed{m_{21}} & \boxed{m_{22}}\end{bmatrix}\begin{pmatrix}\boxed{\Delta P_w / \Delta t}\\ \boxed{\Delta P_{nw} / \Delta t}\end{pmatrix}=\begin{pmatrix}\boxed{R_1}\\ \boxed{R_2}\end{pmatrix} \tag{4.88}$$

In this way, the solver is able to take known values of the variables at the current time step and solve the changes in pressure at the next time step.

5.3.3 Fourth-Order Runge-Kutta Method

The RK4 solver operates as follows:

$$\phi_0 \rho_w \frac{S_{wj}^{n+1,m} + C(\delta P_{nwj}^{n+1,m+1} - \delta P_{wj}^{n+1,m+1}) - S_{wj}^{n}}{\Delta t} + \phi_0 S_w \alpha_w \frac{P_{wj}^{n+1,m} + \delta P_{wj}^{n+1,m+1} - P_{wj}^{n}}{\Delta t} - \frac{1}{\Delta x}\begin{bmatrix} \rho_{nw\,j+1/2}^{n,m} B_{w\,j+1/2}^{n,m}\left(\dfrac{P_{w\,j+1}^{n+1,m} + \delta P_{w\,j+1}^{n+1,m+1} - P_{wj}^{n+1,m} - \delta P_{wj}^{n+1,m+1}}{\Delta x} - \rho_{w\,j+1/2}^{n,m} G \right) \\ -B_{w\,j-1/2}^{n,m}\left(\dfrac{P_{wj}^{n+1,m} + \delta P_{wj}^{n+1,m+1} - P_{w\,j-1}^{n+1,m} - \delta P_{w\,j-1}^{n+1,m+1}}{\Delta x} - \rho_{w\,j-1/2}^{n,m} G \right) \end{bmatrix} = 0 \tag{4.89}$$

Or, more conveniently:

$$\left[\Delta p_{wj}^{n+1}, \Delta p_{nwj}^{n+1}\right] = f\left(p_{w\,j-1}^{n}, p_{wj}^{n}, p_{w\,j+1}^{n}, p_{nw\,j-1}^{n}, p_{nwj}^{n}, p_{nw\,j+1}^{n}\right)\Delta t \tag{4.90}$$

Using RK4, the four orders of changes in pressure are calculated as follows:

$$\left[\Delta p_{wj}^{n+1,1}, \Delta p_{nwj}^{n+1,1}\right] = f\left(p_{w\,j-1}^{n}, p_{wj}^{n}, p_{w\,j+1}^{n}, p_{nw\,j-1}^{n}, p_{nwj}^{n}, p_{nw\,j+1}^{n}\right)\Delta t \tag{4.91}$$

$$\left[\Delta p_{wj}^{n+1,2}, \Delta p_{nwj}^{n+1,2}\right] = f\begin{pmatrix} p_{w\,j-1}^{n} + \dfrac{\Delta p_{w\,j-1}^{n+1,1}}{2}, p_{wj}^{n} + \dfrac{\Delta p_{wj}^{n+1,1}}{2}, p_{w\,j+1}^{n} + \dfrac{\Delta p_{w\,j+1}^{n+1,1}}{2}, \\ p_{nw\,j-1}^{n} + \dfrac{\Delta p_{nw\,j-1}^{n+1,1}}{2}, p_{nwj}^{n} + \dfrac{\Delta p_{nwj}^{n+1,1}}{2}, p_{nw\,j+1}^{n} + \dfrac{\Delta p_{nw\,j+1}^{n+1,1}}{2} \end{pmatrix}\Delta t \tag{4.92}$$

$$\left[\Delta p_{wj}^{n+1,3}, \Delta p_{nwj}^{n+1,3}\right] = f\begin{pmatrix} p_{w\,j-1}^{n} + \dfrac{\Delta p_{w\,j-1}^{n+1,2}}{2}, p_{wj}^{n} + \dfrac{\Delta p_{wj}^{n+1,2}}{2}, p_{w\,j+1}^{n} + \dfrac{\Delta p_{w\,j+1}^{n+1,2}}{2}, \\ p_{nw\,j-1}^{n} + \dfrac{\Delta p_{nw\,j-1}^{n+1,2}}{2}, p_{nwj}^{n} + \dfrac{\Delta p_{nwj}^{n+1,2}}{2}, p_{nw\,j+1}^{n} + \dfrac{\Delta p_{nw\,j+1}^{n+1,2}}{2} \end{pmatrix}\Delta t \tag{4.93}$$

$$\left[\Delta p_{wj}^{n+1,4}, \Delta p_{nwj}^{n+1,4}\right] = f\begin{pmatrix} p_{w\,j-1}^{n} + \Delta p_{w\,j-1}^{n+1,3}, p_{wj}^{n} + \Delta p_{wj}^{n+1,3}, p_{w\,j+1}^{n} + \Delta p_{w\,j+1}^{n+1,3}, \\ p_{nw\,j-1}^{n} + \Delta p_{nw\,j-1}^{n+1,3}, p_{nwj}^{n} + \Delta p_{nwj}^{n+1,3}, p_{nw\,j+1}^{n} + \Delta p_{nw\,j+1}^{n+1,3} \end{pmatrix}\Delta t \tag{4.94}$$

The solver updates the pressure as follows:

$$p_{wj}^{n+1} = p_{wj}^{n} + \frac{1}{6}\left(\Delta p_{wj}^{n+1,1} + 2\Delta p_{wj}^{n+1,2} + 2\Delta p_{wj}^{n+1,3} + \Delta p_{wj}^{n+1,4}\right) \tag{4.95}$$

$$p_{nwj}^{n+1} = p_{nwj}^{n} + \frac{1}{6}\left(\Delta p_{nwj}^{n+1,1} + 2\Delta p_{nwj}^{n+1,2} + 2\Delta p_{nwj}^{n+1,3} + \Delta p_{nwj}^{n+1,4}\right) \tag{4.96}$$

The error can be defined as follows:

$$ERROR = MAX\{abs(p_{wj}^{n+1,*} - p_{wj}^{n+1}), abs(p_{nwj}^{n+1,*} - p_{nwj}^{n+1})\} \tag{4.97}$$

The RK4 solver has the ability to update the time step Δt as needed based on the error. If the error is larger than some set tolerance, the above calculation is abandoned and run again with a reduced time step, which is calculated as follows:

$$\Delta t_{new} = \Delta t \left| \frac{TOL}{ERROR} \right|^{0.2} \tag{4.98}$$

If the error is smaller than the set tolerance, the update is accepted and the time step is increased for the next iteration as follows:

$$\Delta t_{next} = \Delta t \left| \frac{TOL}{ERROR} \right|^{0.2} \tag{4.99}$$

5.3.4 Constant Pressure Boundary Condition at the Inflow

Since the RK4 solver defines the change in pressure at the first node directly, it is extremely easy to enforce constant pressure boundaries at the inflow by assigning the variables as follows:

$$\delta P_{w1} = 0 \tag{4.100}$$

$$\delta P_{nw1} = 0 \tag{4.101}$$

5.3.5 Constant Flux Boundary Condition at the Inflow

A constant flux boundary at the inflow is governed by Darcy's Law for the wetting and nonwetting fluids at the first node:

$$q_{w3/2}^{n+1} = -B_{w3/2}^{\;n} \left(\frac{P_{w2}^{\;n} - P_{w1}^{\;n+1}}{\Delta x} - \rho_{w3/2}^{\;n} G \right) \tag{4.102}$$

$$q_{nw3/2}^{\;n} = -B_{nw3/2}^{\;n} \left(\frac{P_{nw2}^{\;n} - P_{nw1}^{\;n+1}}{\Delta x} - \rho_{nw3/2}^{\;n} G \right) \tag{4.103}$$

In order to incorporate the flux of the wetting fluid at the inflow, equations (4.102) and (4.103) can be rearranged to define the pressure of the fluids at the inflow as follows:

$$P_{w1}^{n+1} = P_{w2}^{n} - \Delta x\left(\frac{q_{w3/2}^{n}}{-B_{w3/2}^{n}} + \rho_{w3/2}^{n} G \right) \tag{4.104}$$

$$P_{nw1}^{n+1} = P_{nw2}^{n} - \Delta x\left(\frac{q_{nw3/2}^{n}}{-B_{nw3/2}^{n}} + \rho_{nw3/2}^{n} G \right) \tag{4.105}$$

These variables may be directly input in the code to enforce the boundary conditions.

5.3.6 Open-End Boundary Condition at the Outflow

Since the RK4 solver defines the change in pressure at the last node directly, constant pressure boundaries at the outflow are defined as follows:

$$\delta P_{wJ} = 0 \tag{4.106}$$

$$\delta P_{nwJ} = 0 \tag{4.107}$$

Identically to the constant pressure boundary at the inflow discussed in section 5.3.4, the constant pressure boundary at the outflow is also easily enforced by assigning the variables as stated in equations (4.106) and (4.107).

5.3.7 Closed-End Boundary Condition at the Outflow

Similar to the closed end boundary conditions discussed in section 5.2.7, this boundary can be enforced according to Darcy's Law for the nonwetting fluid at the last node:

$$q_{wJ-1/2}^{n} = -B_{wJ-1/2}^{n}\left(\frac{P_{wJ}^{n+1} - P_{wJ-1}^{n}}{\Delta x} - \rho_{wJ-1/2}^{n} G \right) \tag{4.108}$$

$$q_{nwJ-1/2}^{n} = -B_{nwJ-1/2}^{n}\left(\frac{P_{nwJ}^{n+1} - P_{nwJ-1}^{n}}{\Delta x} - \rho_{nwJ-1/2}^{n} G \right) \tag{4.109}$$

Equations (4.108) and (4.109) can be rearranged to solve the pressure of the nonwetting fluid at the outflow as follows:

$$P_{wJ}^{n+1} = P_{wJ-1}^{n} + \Delta x\left(\frac{q_{wJ-1/2}^{n}}{-B_{wJ-1/2}^{n}} + \rho_{wJ-1/2}^{n} G \right) \tag{4.110}$$

$$P_{nwJ}^{n+1} = P_{nwJ-1}^{n} + \Delta x \left(\frac{q_{nwJ-1/2}^{n}}{-B_{nwJ-1/2}^{n}} + \rho_{nwJ-1/2}^{n} G \right) \tag{4.111}$$

5.3.8 Model Comparison between Picard and RK4 Solvers

A comparison study between the Picard and RK4 solvers was performed to confirm that both solvers give the same solution. The input parameters for the comparison study are listed in Table 5.1 and Table 5.2.

Table 5.1 Input parameters used to compare Picard and RK4 solvers

Parameter	Symbol	Value	Unit
SWRC N-Value	N	2.4	-
Intrinsic Permeability	K_{sat} or κ	1.00E-11	m^2
Wetting Fluid Viscocity	μ_w	8.90E-04	kg/m/s
Nonwetting Fluid Viscocity	μ_{nw}	1.81E-05	kg/m/s
Compressibility of Water	C_w	5.10E-10	Pa^{-1}
Density of Water	ρ_w	1000	kg/m^3
Density of Air	ρ_{nw}	1.225	kg/m^3
Inflow of Wetting Fluid	q	8	cm/hr
Porosity	φ	0.37	-
Length	x	93.5	cm
Residual Saturation	S_r	0.085	-
Residual Saturation of Nonwetting Fluid	S_{rnw}	0.15-0.25	-

Table 5.2 Constant pressure boundary conditions used to compare Picard and RK4 solvers

Parameter	Symbol	Value	Unit
Wetting Pressure at the Inflow	$P_{w,left}$	10000	Pa
Nonwetting Pressure at the Inflow	$P_{nw,left}$	10134	Pa
Wetting Pressure at the Outflow	$P_{w,right}$	3000	Pa
Nonwetting Pressure at the outflow	$P_{nw,right}$	10135	Pa

For the comparison study between the Picard solver and the RK4 solver, the variables of interest include the degree of saturation of the wetting fluid, the wetting fluid pressure, the nonwetting fluid pressure, the capillary pressure, and the flow both the wetting and nonwetting fluids. The following figures present a comparison between variables of interest from the Picard and RK4 solvers at $t = 0.05$ seconds.

It is important to note that in the following figures the gravitational constant pulls along the direction of x.

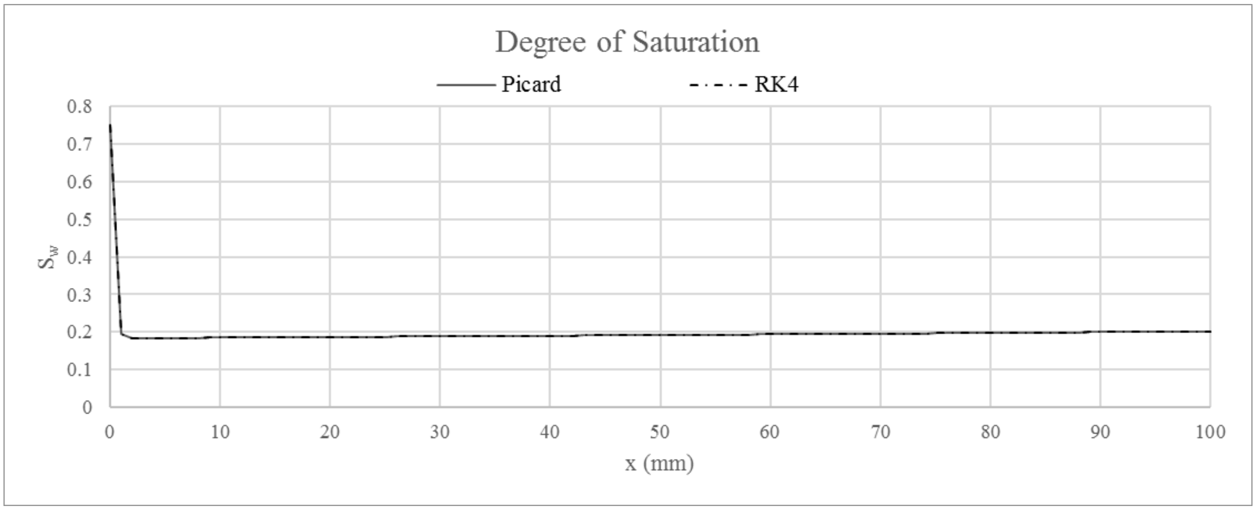


Figure 5.1 A comparison between the simulated degree of saturation from the Picard and RK4 solvers.

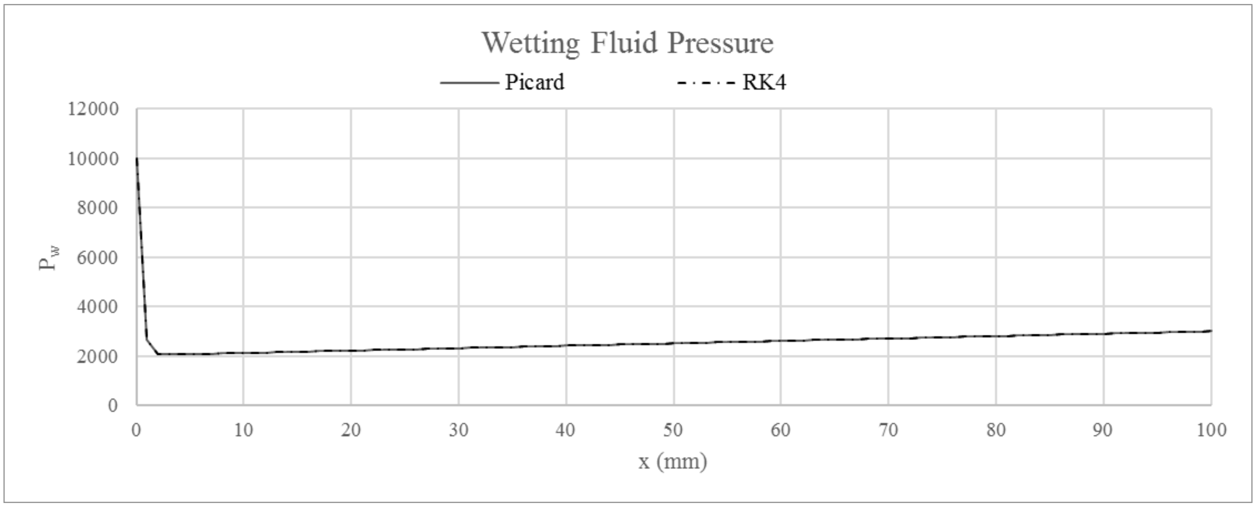


Figure 5.2 A comparison between the simulated pressure of the wetting fluid from the Picard and RK4 solvers.

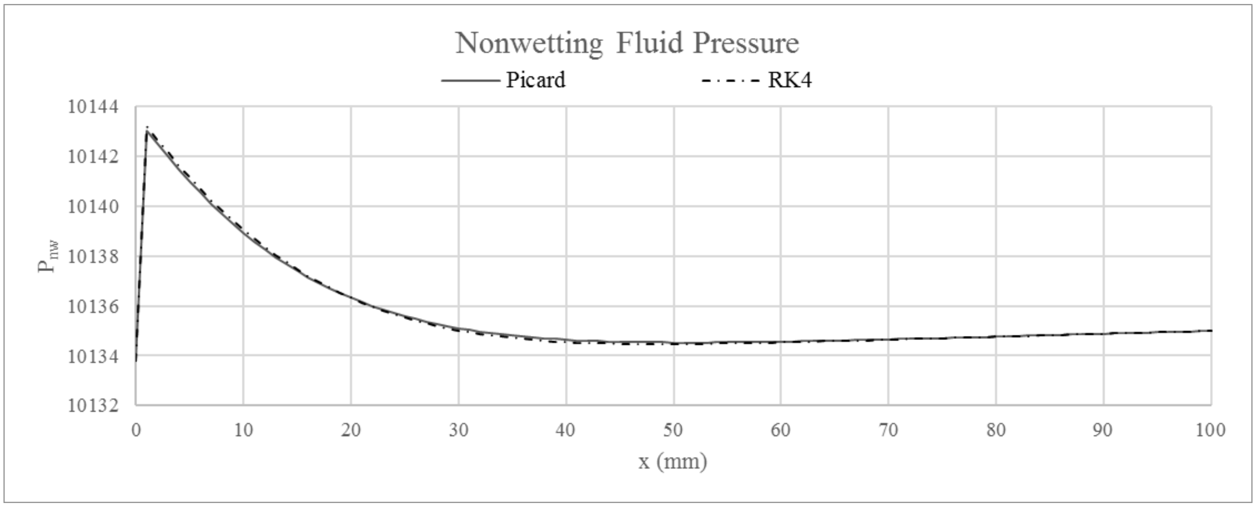


Figure 5.3 A comparison between the simulated pressure of the nonwetting fluid from the Picard and RK4 solvers.

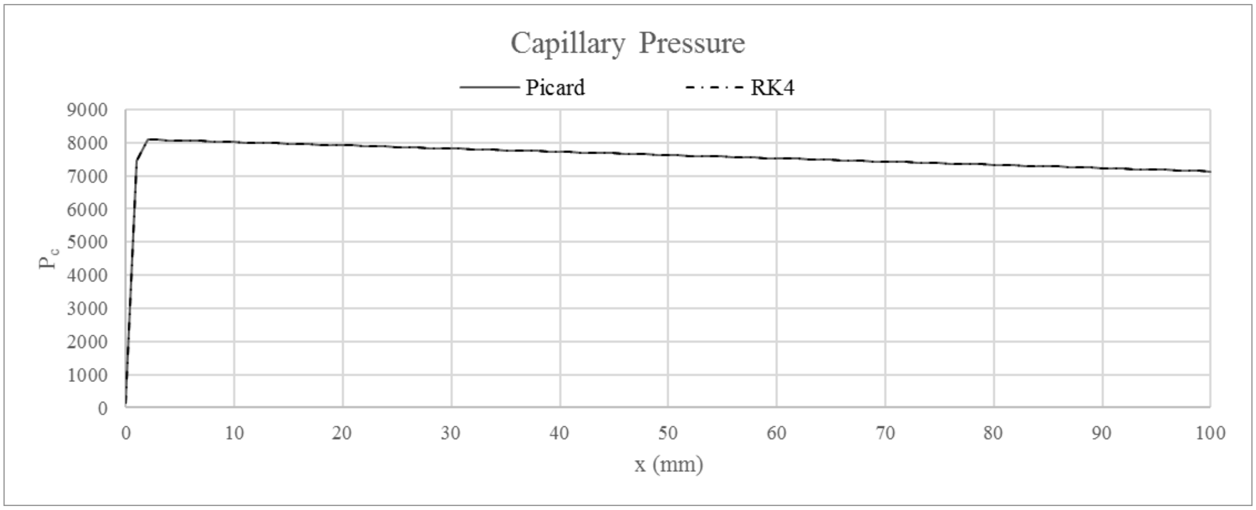


Figure 5.4 A comparison between the simulated capillary pressure from the Picard and RK4 solvers.

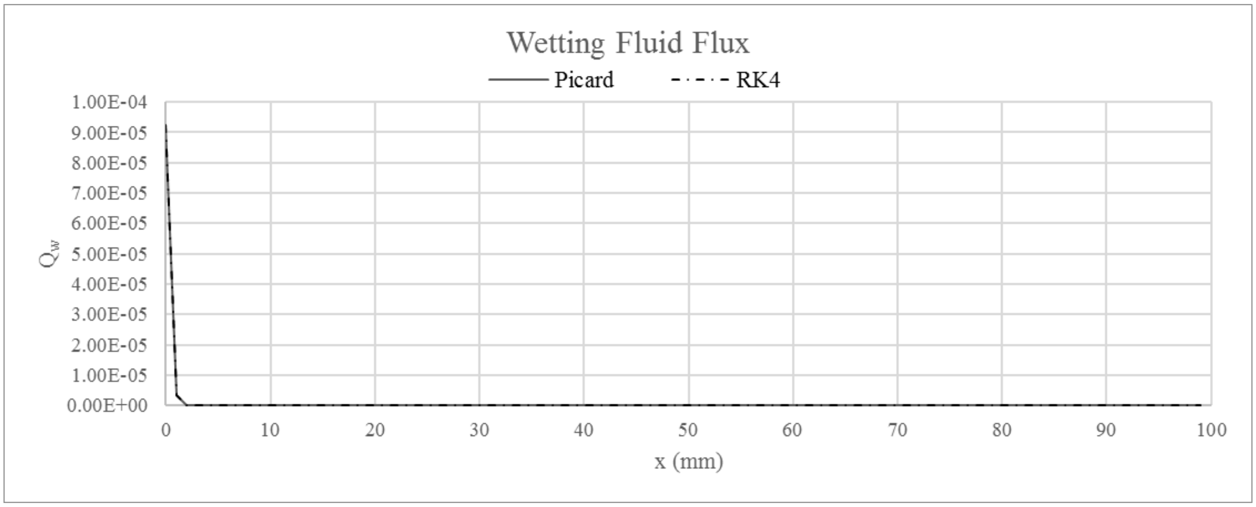


Figure 5.5 A comparison between the simulated flux of the wetting fluid from the Picard and RK4 solvers.

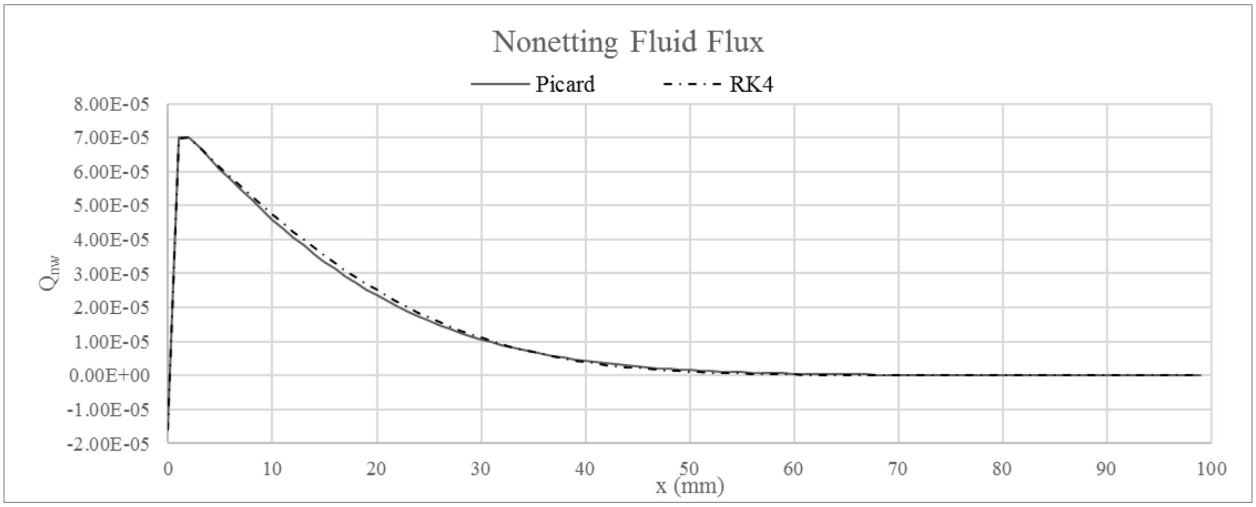


Figure 5.6 A comparison between the simulated flux of the nonwetting fluid from the Picard and RK4 solvers.

From Figure 5.1 to Figure 5.6, it is plain to see that the two solvers produce essentially identical solutions. It is important to note, however, that both solver require small step sizes to produce the essentially identical solutions.

Table 5.3 Step sizes for the Picard and RK4 solvers required to produce (essentially) identical solutions.

Step	**Unit**	**Step Size**	
		Picard	**RK4**
dt	s	0.001	0.001
dx	m	0.001	0.001

It should be noted that larger step sizes (for either solver) produce disagreement between the two solutions.

From this comparison study, researchers were able to debug problems with both solvers based on the overall results. Following much study and analysis, it was determined that these results are correct for the input parameters used. The results are further validated using Touma and Vauclin’s 1986 experimental data.

5.3.9 Computational Time Comparison between Picard and RK4 Solvers

As discussed in section 2.5, though explicit methods are usually easier to implement, they also have maximum allowable step sizes and are generally more "computationally expensive." This was confirmed by the two solvers developed for modelling compressible flow.

The following comparison for the two solvers represents constant pressure boundary conditions at the inflow and the outflow. Table 5.4 presents the initial input parameters used in the comparison, and Table 5.5 presents the run time taken for the initial parameters, and how the run time for each solver changes when the different input parameters are varied.

Table 5.4 Initial parameters used to compare Picard and RK4 Solvers

Parameter	**Symbol**	**Value**	**Unit**
Spatial Step	dx	0.01	m
Timestep	dt	0.01	s
Tolerance for Error	tol	1.00E-04	-
Length	x_{end}	1	m
Run Time	t_{end}	0.02	s

Table 5.5 Varying input parameters to observe the effects on the computational time of each solver

Parameter	**Time Elapsed (s)**	
	Picard	**RK4**
Initial Comparison	0.67	2.33
dx = 0.001 m	3.98	28.54
dt = 0.001 s	2.41	2.29
tol = 1.00E-06	0.69	19.99

From Table 5.5, one can see the RK4 method generally takes much longer to run than the Picard method, with the one exception to this observation being reducing the temporal step size. This is because the RK4 solver has the ability to adjust the temporal step size between successive iterations based on the error. In general, however, it is evident that the RK4 solver runs significantly slower than the Picard solver, with the maximum being almost 30 times slower when the tolerance for error is decreased. Though the variable tolerance has a different fundamental meaning in each solver, it is still easy to see why researchers prefer the Picard method for solving Richard's equation.

5.4 Model Validation

5.4.1 Experimental Background

The Picard model was validated by producing a best fit to the experimental data captured by Touma and Vauclin (1986). Touma and Vauclin's experiments model the vertical dispersion of imbibition fronts in a compressible water-air system. The 93.5cm deep vertical column of sandy soil was subjected to varying boundary conditions and water injection velocities. Volumetric water content at varying times was measured by attenuation of gammy rays emitted by a 241 Am, 100 mCr source (Touma & Vauclin, 1986). Touma and Vauclin's paper gives the inputs presented in Table 5.6.

Table 5.6 Experimental Parameters given by Touma and Vauclin (Touma & Vauclin, 1986)

Parameter	Symbol	Value	Unit
Intrinsic Permeability	K_{sat} or κ	1.00E-11	m^2
Wetting Fluid Viscocity	μ_w	8.90E-04	kg/m/s
Nonwetting Fluid Viscocity	μ_{nw}	1.81E-05	kg/m/s
Porosity	φ	0.37	-
Length	x	93.5	cm

Table 5.7 presents the boundary conditions extracted from Touma and Vauclin's paper.

Table 5.7 Boundary conditions for matching Touma and Vauclin's experimental data

Parameter	Symbol	Value	Unit
Wetting Flux at the Inflow	q_{w0}	varies	cm/hr
Nonwetting Pressure at the Inflow	$P_{nw,left}$	10134	Pa
Wetting Pressure at the Outflow	$P_{w,right}$	3000	Pa
Nonwetting Pressure at the outflow	$P_{nw,right}$	10135	Pa

5.4.2 *Soil Water Retention Curve Calibration*

The soil water retention curve in the model was calibrated based on Touma and Vauclin's experimentally determined curve:

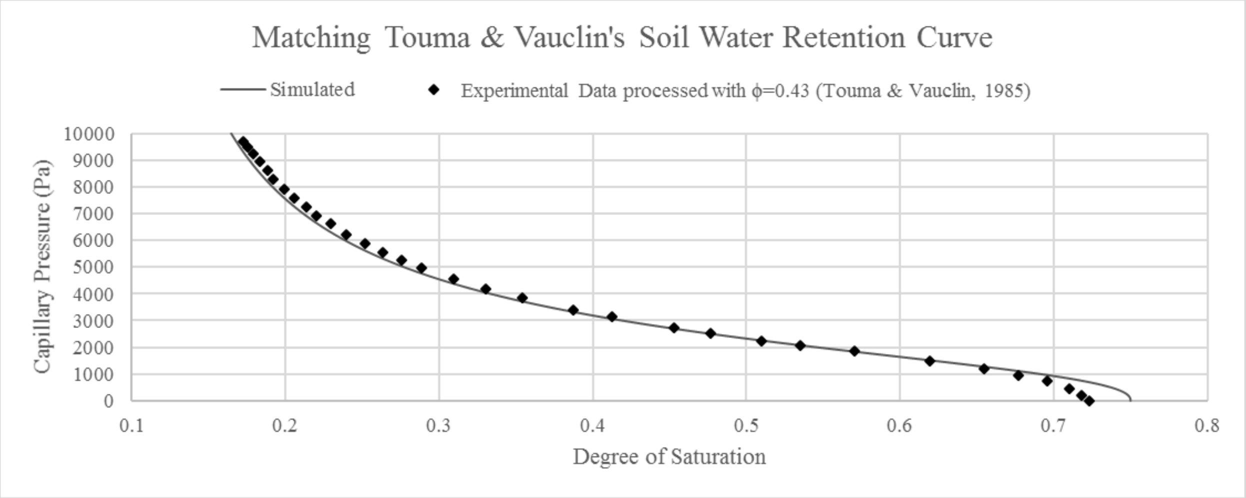


Figure 5.7 Calibration of the solver based on the experimentally determined soil water retention curve data processed with φ = 0.43 (Touma & Vauclin, 1986).

The calibrated soil water retention curve parameters are given in Table 5.8.

Table 5.8 Calibrated soil water retention curve parameters

Parameter	Symbol	Value	Unit
SWRC *N*-Value	N	2.35	-
Air Entry Value	p_0	2300	Pa
Residual Saturation	S_r	0.085	-
Residual Saturation of Nonwetting Fluid	S_{rnw}	0.15-0.25	-

The reason the residual degree of saturation varies between S_{rnw} = *0.15-0.25* is likely due to an error in the experimental set up, i.e. inconsistent sample preparation or other uncertain factors.

5.4.3 *Model Validation*

Figure 5.8 and Figure 5.9 present the best fit imbibition fronts with injection rates of 8.3cm/hr and 20cm/hr respectively at the inflow and an open-end boundary condition at the inflow. It is important to note that in the following figures the gravitational constant pulls along the direction of x.

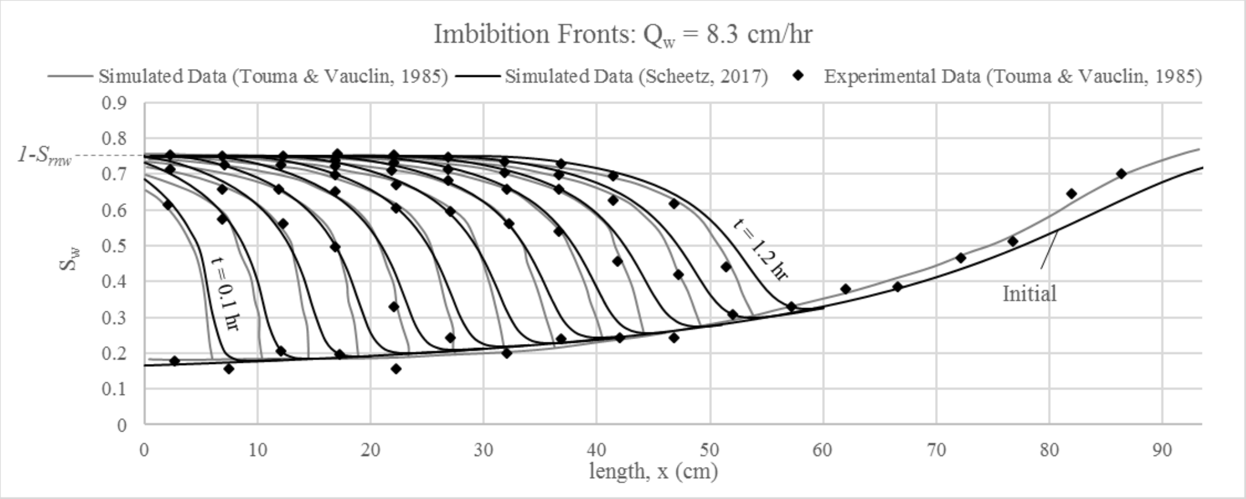


Figure 5.8 Best fit to Touma and Vauclin's experimental data for the imbibition fronts of a two-phase compressible flow system subject to a constant flux boundary condition of 8.3 cm/hr at the inflow and an open-end boundary condition at the outflow. Note that S_{rnw} = 0.25.

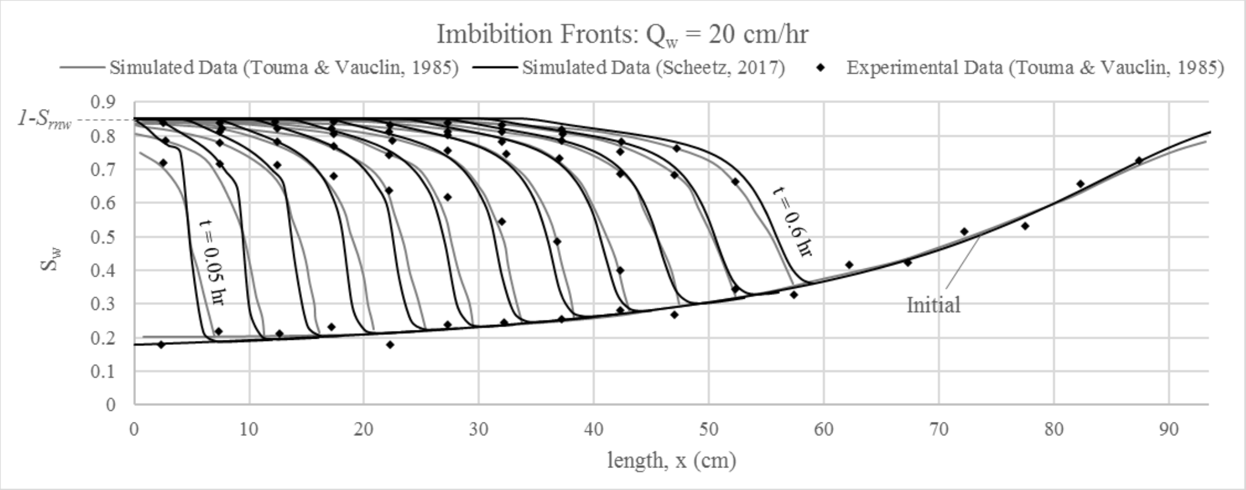


Figure 5.9 Best fit to Touma and Vauclin's experimental data for the imbibition fronts of a two-phase compressible flow system subject to a constant flux boundary condition of 20 cm/hr at the inflow and an open-end boundary condition at the outflow. Note that S_{rnw} = 0.15.

It is exciting to observe, that for the open-ended outflow boundary experiments with varying injection rates, the simulator’s solutions are very close to Touma and Vauclin’s simulated solution, and also a very good match to Touma and Vauclin’s experimental data.

5.5 Discussion

5.5.1 Introduction of Residual Saturation of the Nonwetting Fluid

The residual degree of saturation was introduced in this chapter after the shapes of the plots for the variables of interest were observed without this term. As discussed in section 5.1.3, the introduction of the residual degree of saturation of the nonwetting fluid “caps” the maximum degree of saturation observed according to the following equation

$$S_{w\max} = 1 - S_{rnw} \tag{4.112}$$

For Touma and Vauclin’s experiment with Q_w = *8.3 cm/hr*, the maximum degree of saturation observed is approximately S_w = *0.75*, so the corresponding residual degree of saturation of the nonwetting fluid is set at S_{rnw} = *0.25*. Similarly, the maximum degree of saturation observed with Q_w = *20 cm/hr*, the maximum degree of saturation observed is approximately S_w = *0.85*, so the corresponding residual degree of saturation of the nonwetting fluid is set at S_{rnw} = *0.15*.

5.5.2 Unusual Curve Shapes Observed

Researchers noted that Touma and Vauclin report only the initially observed volumetric water content profiles for all simulations. It was observed that the profiles change dramatically as the wetting fluid infiltrates through the porous medium. See Figure 5.10 through Figure 5.15 for reference.

Note that the following figures present plots for the variables of interest for simulations without the S_{rnw} term in equation (4.50). Also note that in the following figures, the gravitational constant pulls to the right.

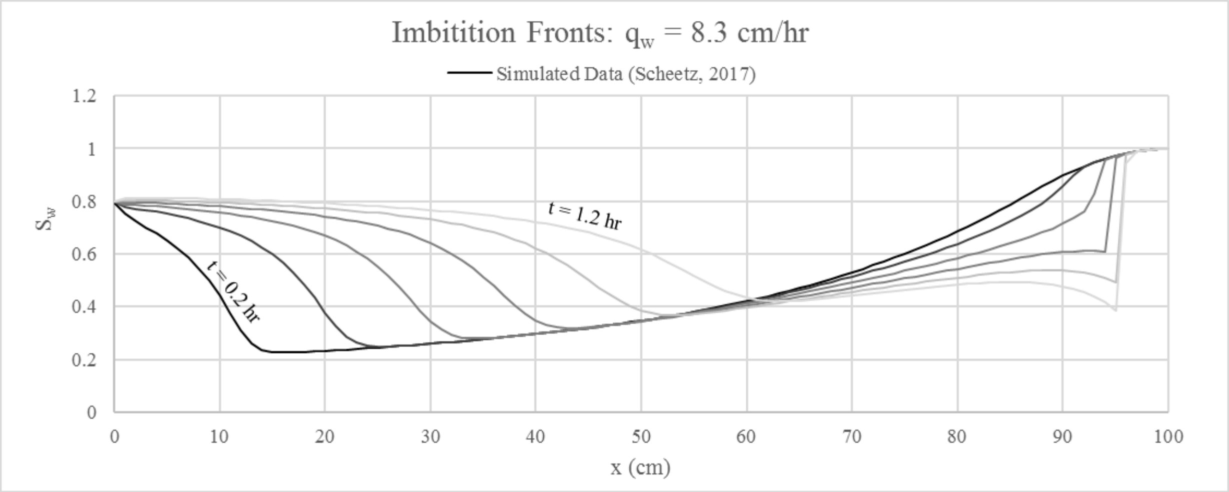


Figure 5.10 Imbibition fronts for all specified time increments.

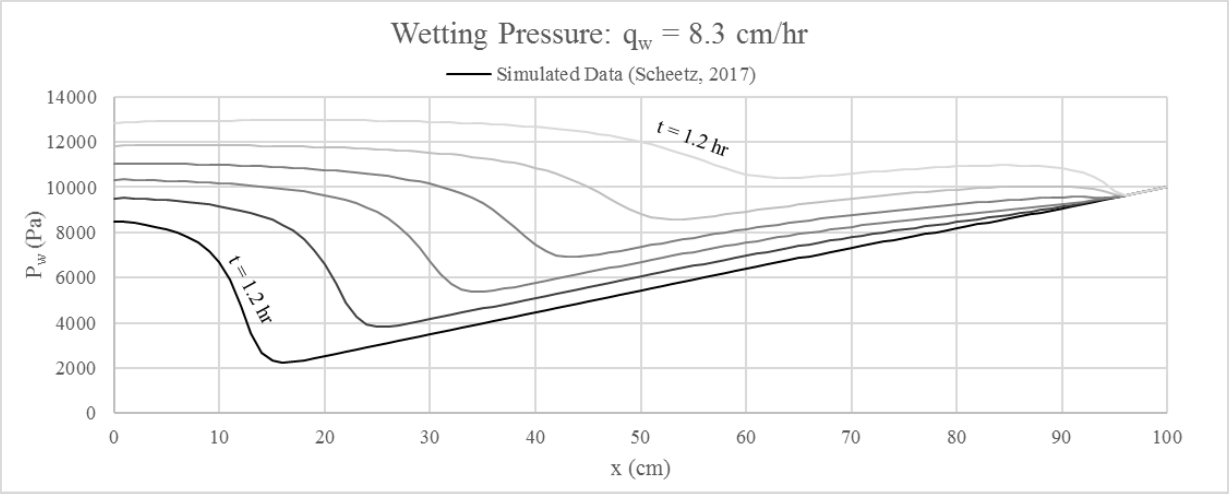


Figure 5.11 Wetting pressure profiles for all specified time increments.

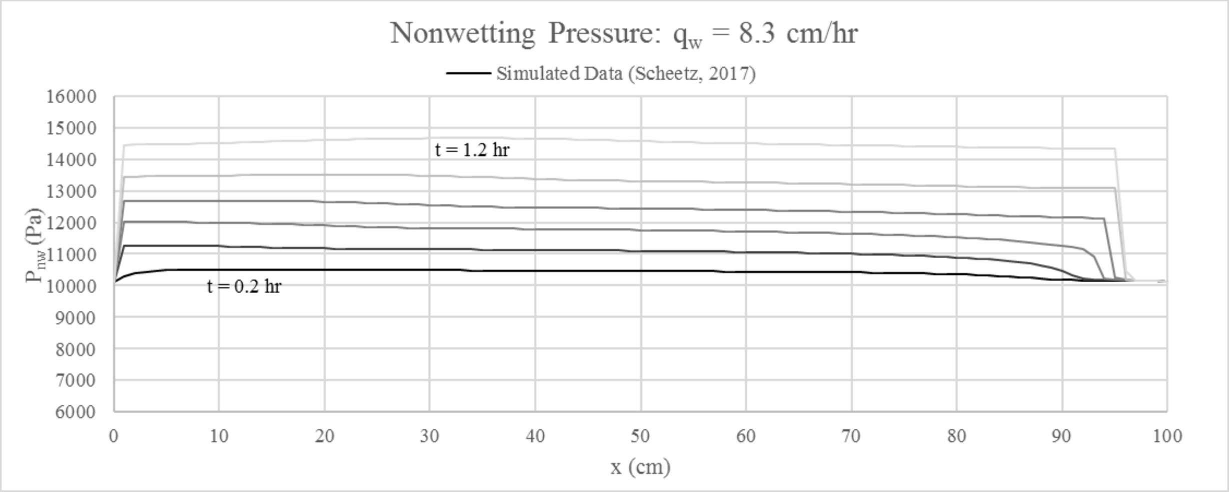


Figure 5.12 Nonwetting pressure profiles for all specified time increments.

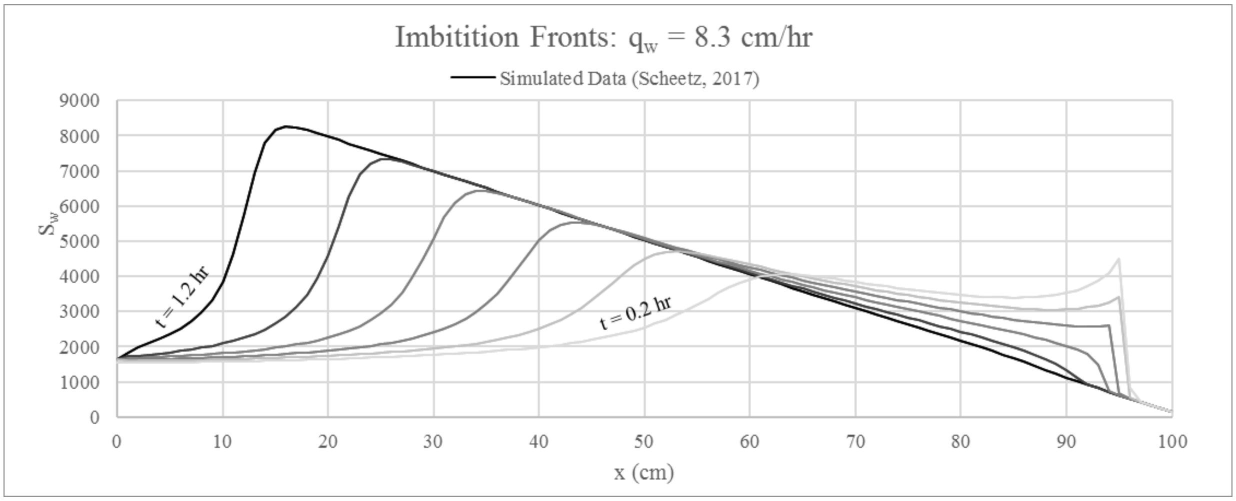


Figure 5.13 Capillary pressure profiles for all specified time increments.

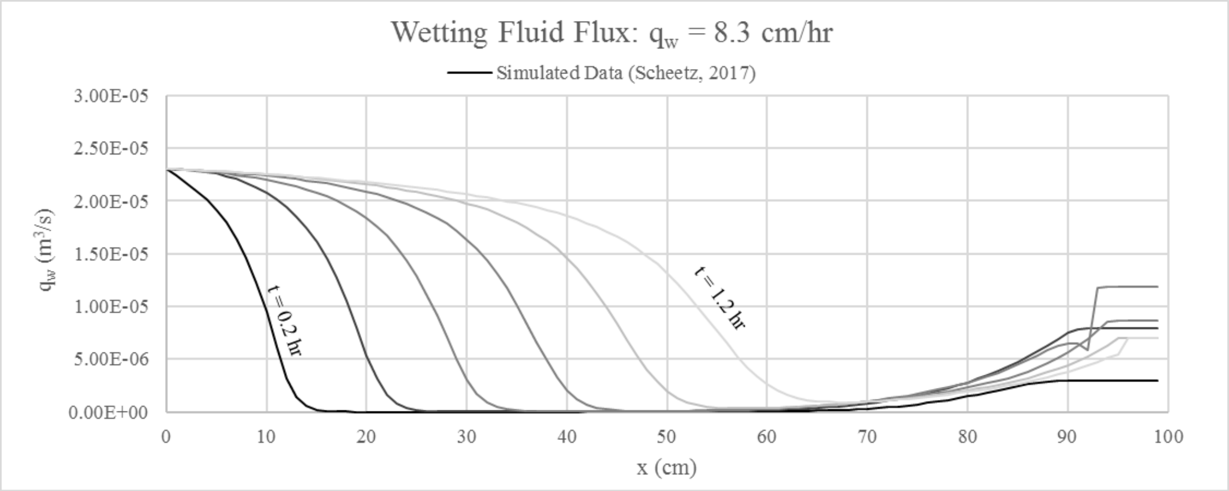


Figure 5.14 Flux of the wetting fluid for all specified time increments.

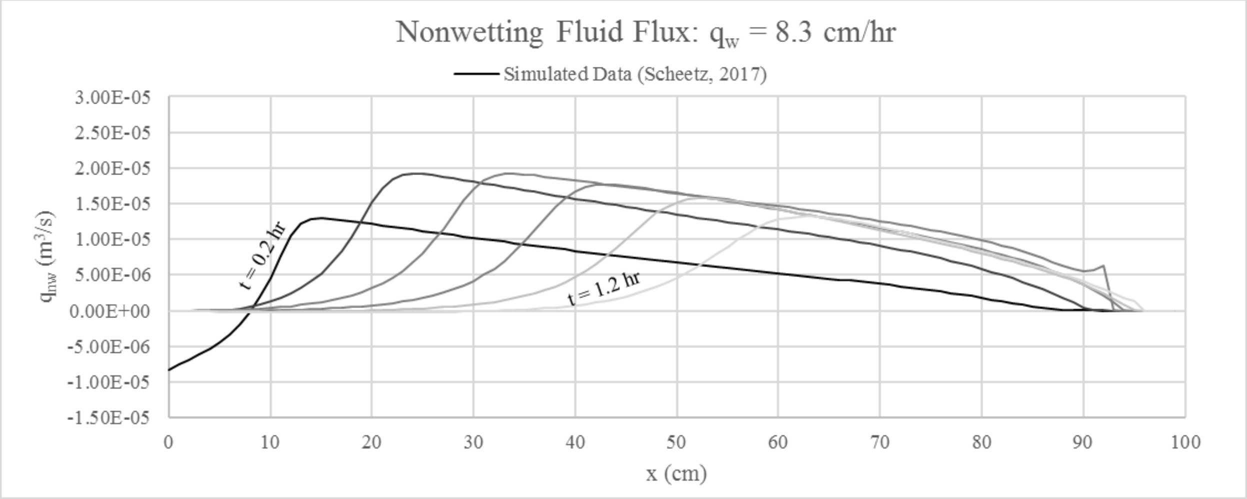


Figure 5.15 Flux of the nonwetting fluid for all time increments.

From these figures, one can see that the shapes are very unusual. Even through Touma and Vauclin only reported the initial S_w profile shortly after the onset of water injection, the dramatic drop in the S_w profile predicted by the solver was unexpected.

To remedy this, the van Genutchen model was modified by including a residual degree of saturation for the nonwetting fluid, hence the soil retention curve does not start from $S_w = 1$ at $P_c = 0$ but rather $S_w = 1 - S_{rnw}$ at $P_c = 0$, a modification that allows the nonwetting fluid to drain at the outflow. Even with this change, however, the plot shapes are still unusual. Researchers observed that closed-end simulations did not exhibit the same unusual curves.

5.5.3 Closed-End Boundary Condition

Touma and Vauclin's closed-end experiments show that the velocity of the wetting front slows down significantly relative to the open-end experiments. The simulator's solution does not reflect this. Figure 5.16 presents the simulator's solution and Touma and Vauclin's simulated response and experimental data. Note that in the following figure, the gravitational constant pulls along the direction of x.

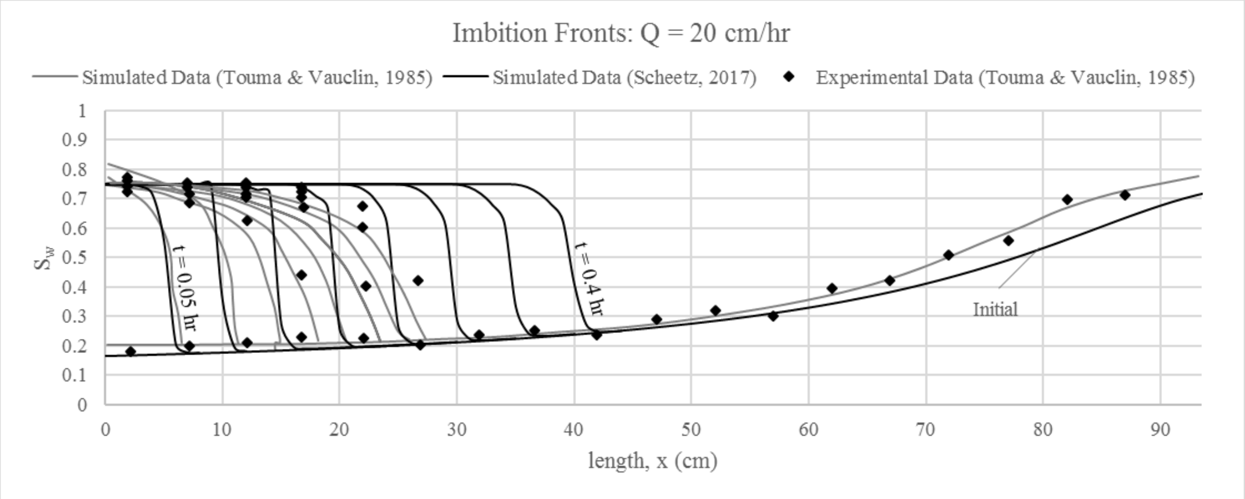


Figure 5.16 An attempt to fit Touma and Vauclin's experimental data for the imbibition fronts of a two-phase compressible flow system subject to a constant flux boundary condition of 20 cm/hr at the inflow and an closed-end boundary condition at the outflow.

The simulated solution for the closed-end system does not match Touma and Vauclin's simulated solution nor their experimental data. Researchers ran a comparison study between the open and closed-end simulations to understand the reason for this discrepancy. The following figures present a comparison between the infiltration fronts, nonwetting pressure profiles and nonwetting density profiles of open and closed-end simulations at time *t = 1200* seconds.

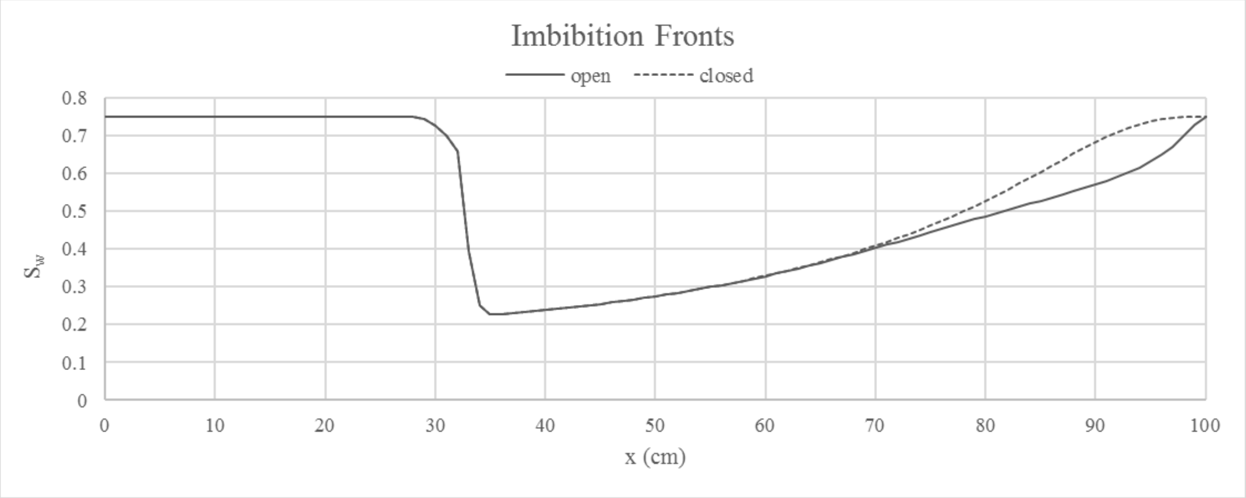


Figure 5.17 Comparison between imbibition fronts for open-end and closed-end simulations.

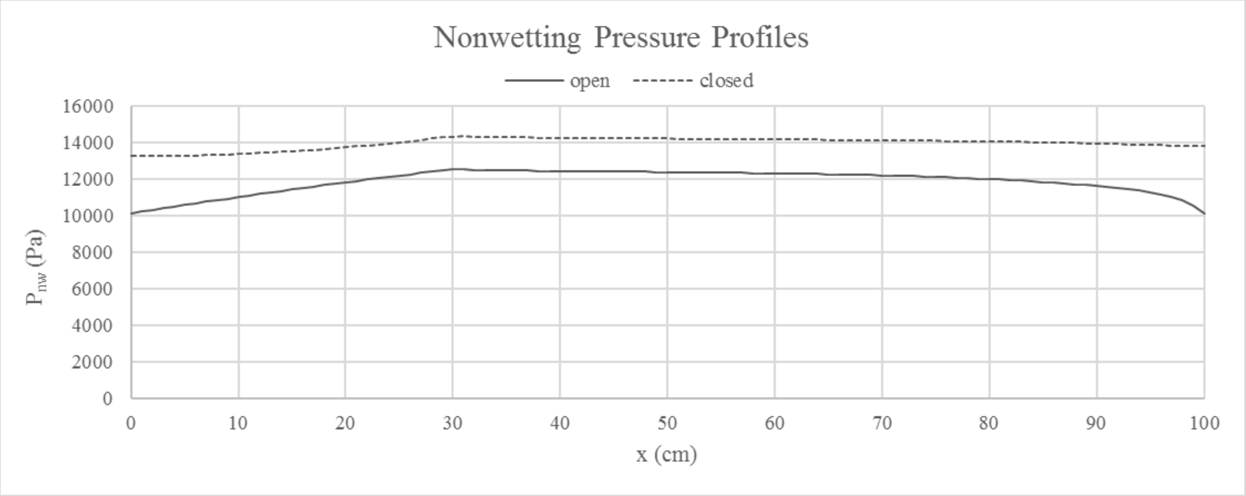


Figure 5.18 Comparison between nonwetting pressure profiles for open-end and closed-end simulations.

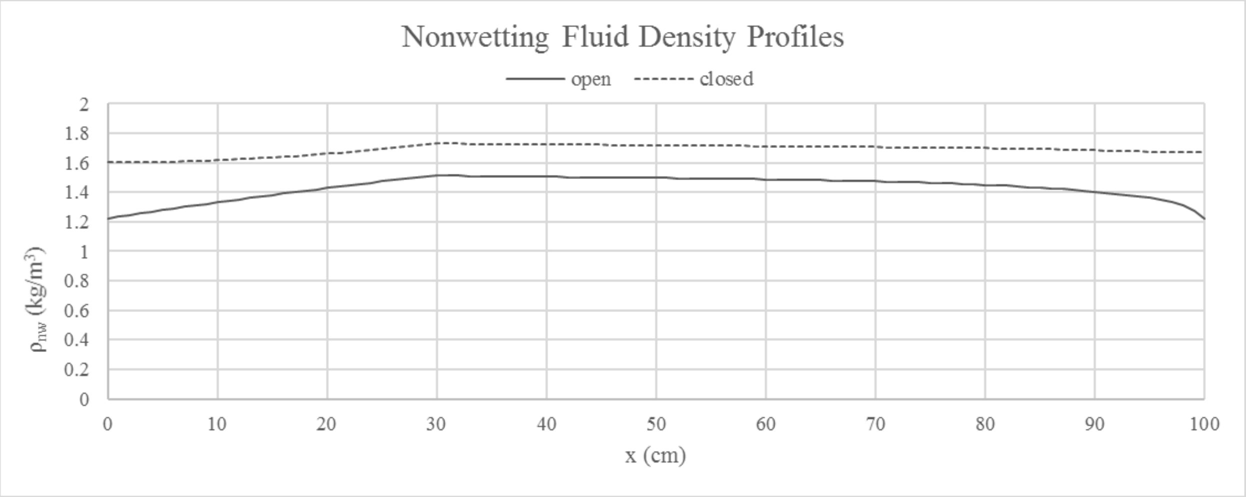


Figure 5.19 Comparison between nonwetting fluid density profiles for open-end and closed-end simulations.

From Figure 5.17, it is observed that the simulated wetting fronts travel at approximately the same velocity for both open and closed end-boundary conditions at the outflow. This is starkly different from the experimental results observed by Touma and Vauclin.

Fundamentally, it makes sense that a closed-end simulation should produce a higher nonwetting fluid (air) pressure and subsequently a higher air density throughout the sample relative to an open-end simulation. Figure 5.18 and Figure 5.19 reflect this. Based on this, researchers conclude that the numerical solver produces a fundamentally accurate solution, but that, for this specific set of input parameters, the differences in nonwetting fluid pressure observed are not large enough to influence the velocity of the wetting front. It is possible that a different set of input parameters could better capture Touma and Vauclin’s experimental data.

5.5.4 Convergence Improvement

Like the model developed for Chapter 4, this model does not exhibit convergence issues similar to those experienced with the constant flux solver discussed in Chapter 3. This Picard model for compressible systems is quite robust, even with increasing SWRC *N*-values, higher injection rates and higher air entry values. This may be because of the way the coupled governing equations were formulated and solver, or possibly because realistic values were used for all input parameters.

CHAPTER 6

CONCLUDING REMARKS AND FUTURE WORK

6.1 Summary of Conclusions

In this thesis, researchers have solved the governing equations for two phase flow in a rigid porous skeleton using modified Picard and RK4 methods. Some major conclusions are summarized as follows:

1. With a generic treatment of the advection diffusion equation discussed in Chapter 3, the cases of wetting fluid invasion, nonwetting fluid invasion, and, though not comprehensively discussed in this thesis, the classical Richard's equation can be recovered in the same code framework.
2. During the injection of wetting fluid, the effects of varying the SWRC *N*-value, the initial suction P_c, air entry pressure p_0, and the residual saturation S_r were observed. It was observed that higher N-values create a faster-travelling, more gradual wetting front through the porous medium, higher initial suction values slow down the speed of the wetting front in the porous medium, and higher residual saturation and higher air entry values increase the travelling speed of the wetting front.
3. The effects of viscous fingering may be non-negligible when a less viscous fluid is displacing a more viscous fluid, depending on the set of input parameters selected.
4. It was observed that the convergence the solver developed for Chapter 3 is fairly sensitive to varying model parameters, especially with increasing wetting fluid injection rates q_w. A higher injection rate requires finer spatial and temporal discretization to enhance convergence in the modified Picard solver. This effect is not observed to the same extent for the solvers discussed in Chapters 4 and 5.

5. In the cases that explored in this thesis, the numerical performance of the modified Picard solver far exceeds the performance of the RK4 solver.
6. Though not observed with the set of input parameters selected, the effect of air trapping during wetting may be more significant with a different set of input parameters, as confirmed by the experimental data by Vauclin and the simulation results in Chapter 5.

6.2 Future Work

Future tasks that should immediately follow this thesis include the following:

1. To model blast in a 1-D context, the constant pressure boundary solver should be modified to reflect the pressure-volume relation of the encapsulated gas at the location of detonation.
2. For higher flow rates and blasting scenarios, the flow equation Darcy's Law may need to be replaced by the Forchheimer equation to reflect the nonlinear effects due to inertia at microscale. Calculations of Reynold's numbers during blasting scenarios will be used to confirm this.
3. Most importantly, a third governing equation should be implemented with porosity (or equivalently, displacement *u*) as an unknown. Advanced constitutive models for sand will be implemented to properly coupled the fluid flow with mechanical deformation of soil during blasting.

The above extensions will allow a more realistic simulation of the explosion studies to be performed in the 2017-2018 academic year.

BIBLIOGRAPHY

Buckley, S. E., & Leverett, M. C. (1942). Mechanism of Fluid Displacement in Sands. *Transactions of the AIME, Vol, 146*, 107-116.

Celia, M. A., & Binning, P. (1992). A Mass Conservative Numerical Solution for Two-Phase Flow in Porous Media With Application to Unsaturated Flow . *Water Resources Research, Vol, 28, NO. 10*, 2819-2828.

Celia, M. A., Ahuja, L. R., & Pinder, G. E. (1987). Orthogonal collocation and alternating-direction procedures for unsaturated flow problems . *Advances in Water Resources, Vol. 10, Issue 4*, 178-187.

Celia, M. A., Bouloutas, E. T., & Zarba, R. L. (1990). A General Mass-Conservative Numerical Solution for the Unsaturated Flow Equation. *Water Resources Research, Vol. 26, No. 7*, 1483-1496.

Fokas, A. S., & Yortsos, Y. C. (1982). On the Exactly Solvable Equation S,=[(13S+t)-Sl+(13S+)-S Occuring in Two-Phase Flow in Porous Media. *Siam Journal of Applied Mathematics*, 318-332.

FracFocus. (2017). *FracFocus Chemical Disclosure Registry*. Retrieved from Fluid Flow in the Subsurface (Darcy's Law): https://fracfocus.org/groundwater-protection/fluid-flow-subsurface-darcys-law

Hillel, D. (1980). *Fundamentals of Soil Physics.* Academic Press.

McWhorter, D. B., & Sunada, D. K. (1990). Exact Integral Solutions for Two-Phase Flow. *Water Resources Research Vol. 26, No. 3*, 399-413.

Melean, Y., Broseta, D., Hasmy, A., & Blossey, R. (2003). Dispersion of imbibition fronts. *Europhysics Letters, 62*, 505-511.

Morel-Seytoux, H. J. (1973). *Systematic Treatment of Infiltration with Applications.* Fort Collins: Department of Civil Engineering at Colorado State University.

Mualem, Y. (1976). A New Model for Predicting the Hydraulic Conductivity of Unsaturated Porous Media. *Water Resources Research, Vol. 13, No. 3,* 513-522.

Rahimi, A., Rahardjo, H., & Leong, E.-C. (1997). Effects of soil-water characteristic curve and relative permeability equations on estimation of unsaturated permeability function. *Journal of Geotechnical and Geoenvironmental Engineering* , 1106-1117.

Saffman, P. G., & Taylor, G. (1958). The Penetration of a Fluid into a Porous Medium of Hele-Shaw Cell Containing a More Viscous Fluid . *Proceedings of the Royal Society*, 312-329.

Touma, J., & Vauclin, M. (1986). Experimental and Numerical Analysis of Two-Phase Infiltration in a Partically Saturated Soil. *Transport in Porous Media 1*, 27-55.

youtube.com. (2013). Retrieved from Hele-Shaw cell HD (900CC): https://www.youtube.com/watch?v=dJcomGs2YyA

CODE APPENDIX

Incompressible Flow subjected to a Constant Flux Boundary at the Inflow

Main Code

```
clear
close all
clc

%units: s, m, kg, N, Pa
%define global variables and soil properties (van Genuchten)

global Ksat N M p0 mu_w mu_nw q phi Sr flag

flag = 1;                                % 1-wetting invasion;
                                         % 2-nonwetting invasion
Ksat = 1e-09;                            % m^2
N = 2.0;
M = 1-1/N;
p0 = 300;                                % Pa
mu_w = 0.89e-03;                         % kg/m/s (water)
mu_nw = 1.03e-03;                        % kg/m/s (oil)
q = 100/(3600*1000*1000*pi*(.015)^2);    % flow (m/s)
phi = 0.35;
Sr = 0.05;

% Temporal Discretization
dt =.1; % s
tend = 3620; % s
t = dt:dt:tend;

% Spatial Discretization
dx = 0.001;                              % m
xend = 0.4;                              % m
x = 0:dx:xend;
J = length(x);

% Initial and Boundary Conditions
initpc = 2000;                           % Pa
pcleft = initpc;
pc0 = initpc*ones(length(x),1);
Sw0(1:J,1) = Swfuns_vec(pc0(1:J,1));

% Initialize
pcold = pc0;
Swold = Sw0;

% Time Loop
nt = length(t);
reverseStr = '';
for it = 1:nt
    time = (it-1)*dt;

    % Initial Guesses
    pc = pcold;
```

```
Sw = Swold;

% Constant Pressure BC at the outflow
pc(J,1) = initpc;

% Initialization for Picard Iteration
iter = 0;
error = 999;

% Solve for dPc via Picard iteration
while (error > 1.e-4) && (iter < 100)

    G(1:J,1) = Gfuns_vec(pc(1:J,1));
    H(1:J,1) = Hfuns_vec(pc(1:J,1));
    C(1:J,1) = cpcfuns_vec(pc(1:J,1));

    % Take Harmonic Average for 1/2 nodes
    Gmean(1:J-1,1) = 2./(1./G(1:J-1,1)+1./G(2:J,1));
    Gminus = [0;Gmean];
    Gplus = [Gmean;0];
    Hmean(1:J-1,1) = 2./(1./H(1:J-1,1)+1./H(2:J,1));
    Hminus = [0;Hmean];
    Hplus = [Hmean;0];

    % Define a,b,c and r vectors
    switch flag
        case 1
            a = [0; Gminus(2:J-1)/dx^2; 0];
            b = [Gplus(1,1)/dx; phi*C(2:J-1)/dt - (Gplus(2:J-1)+...
                Gminus(2:J-1))/dx^2;1];
            c = [-Gplus(1,1)/dx; Gplus(2:J-1)/dx^2; 0];
            r = [Hplus(1,1)-q+Gplus(1,1)/dx*(pc(2,1)-pc(1,1)); ...
                -phi*(Sw(2:J-1)-Swold(2:J-1))/dt-(Hplus(2:J-1)-...
                Hminus(2:J-1))/dx-1/dx^2*(Gplus(2:J-1).*(pc(3:J)-...
                pc(2:J-1))-Gminus(2:J-1).*(pc(2:J-1)-pc(1:J-2))); 0];

        case 2
            a = [0; Gminus(2:J-1)/dx^2; 0];
            b = [Gplus(1,1)/dx; phi*C(2:J-1)/dt - (Gplus(2:J-1)+...
                Gminus(2:J-1))/dx^2; 1];
            c = [-Gplus(1,1)/dx; Gplus(2:J-1)/dx^2; 0];
            r = [Hplus(1,1) + q + Gplus(1,1)/dx*(pc(2,1)-pc(1,1)); ...
                -phi*(Sw(2:J-1)-Swold(2:J-1))/dt - (Hplus(2:J-1)-...
                Hminus(2:J-1))/dx - 1/dx^2*(Gplus(2:J-1).*(pc(3:J)-...
                pc(2:J-1))-Gminus(2:J-1).*(pc(2:J-1)-pc(1:J-2))); 0];
    end

    %solve for dpc using Thomas algorithm
    dpc = mythomas(a,b,c,r);
    error = max(abs(dpc));

    %update
    pc = pc + dpc;
    pc(pc<0) = 0;
    Sw(1:J,1) = Swfuns_vec(pc(1:J,1));
    switch flag
        case 1
```

```
            qvar(1:J-1,1) = Hmean + Gmean/dx.*(pc(2:J)-pc(1:J-1));
        case 2
            qvar(1:J-1,1) = -Hmean - Gmean/dx.*(pc(2:J)-pc(1:J-1));
    end

    iter = iter + 1;
end

% update "old" values for next time-step
pcold = pc;
Swold = Sw;

% save Sw and Pc variables
     if time == 200
         Sw900 = Sw;
         Pc900 = pc;
     end

     if time == 400
         Sw1800 = Sw;
         Pc1800 = pc;
     end

     if time == 600
         Sw2700 = Sw;
         Pc2700 = pc;
     end

     if time == 3600
         Sw3600 = Sw;
         Pc3600 = pc;
     end

% Plotting for Animation
% Plotting Sw
subplot(3,1,1)
plot(x,Sw)
ylim([0 1])
xlim([0 xend])
title('Sw vs. x')
ylabel('Degree of saturation, S_w')
xlabel('x (m)')

% Plotting Pc
subplot(3,1,2)
plot(x,pc)
ylim([0 inf])
xlim([0 xend])
title('suction vs. x')
ylabel('Suction, p_c')
xlabel('x (m)')

% Plotting Qw
subplot(3,1,3)
plot(x(1:J-1)+1/2*dx,qvar)
xlim([0 xend])
ylim([0 1.2*q])
```

```
    title('q vs. x')
    ylabel('Flux, q (m/s)')
    xlabel('x (m)')

    % Print
    msg = sprintf('Current time: %d,     Pcleft:%d,     Number of iteration:
%d \n',time,pc(1),iter);
    fprintf([reverseStr, msg]);
    reverseStr = repmat(sprintf('\b'), 1, length(msg));

    if (iter >= 100)
        disp 'no convergence'
        return
    end

    % Refresh
    pause(0.001)
end
```

Sw_funs

```
function Swval = Swfuns_vec(pcval)
%theta(psi) function - water content - suction relationship

global N M p0 Sr
Swval = 1./(1+(pcval/p0).^N).^M.*(1-Sr)+Sr;

end
```

Kfuns

```
function [K] = Kfuns_vec(pcval)
%unsaturated hydraulic conductivity function

global Ksat M
Sw = Swfuns_vec(pcval);
K = Ksat.*sqrt(Sw).*(1-(1-Sw.^(1./M)).^M).^2;
K(pcval<0) = Ksat;

end
```

Gfuns

```
function [G] = Gfuns_vec(pcval)
%unsaturated hydraulic conductivity function

global Ksat mu_w mu_nw
Sw = Swfuns_vec(pcval);
Krw = Krwfuns_vec(Sw);
Krnw = Krnwfuns_vec(Sw);

G = Ksat*Krw.*Krnw./(mu_w*Krnw+mu_nw*Krw);

end
```

Hfuns

```
function [H] = Hfuns_vec(pcval)
%unsaturated hydraulic conductivity function

global mu_w mu_nw q flag
Sw = Swfuns_vec(pcval);
Krw = Krwfuns_vec(Sw);
Krnw = Krnwfuns_vec(Sw);

switch flag
    case 1
        h = mu_nw*Krw./(mu_w*Krnw+mu_nw*Krw);
        H = q*h;
    case 2
        h = mu_w*Krnw./(mu_w*Krnw+mu_nw*Krw);
        H = -q*h;
end

end
```

Krwfuns

```
function [Krw] = Krwfuns_vec(Swval)

% given the degree of saturation of the wetting fluid determine the
% relative conductivity of the wetting fluid

global M
Krw = sqrt(Swval).*(1-(1-Swval.^(1/M)).^M).^2;

end
```

Krnwfuns

```
function [Krnw] = Krnwfuns_vec(Swval)
% given the degree of saturation of the wetting fluid determine the
% relative conductivity of the wetting fluid

global M
Krnw = sqrt(1-Swval).*(1-(1-(1-Swval).^(1./M)).^M).^2;

end
```

cpcfuns

```
function cpcval = cpcfuns_vec(pcval)
%specific moisture capacity function

global N p0 Sr
cpcval = (-N+1)*((pcval/p0).^N+1).^(-2+1/N).*(pcval/p0).^(N-1)/p0.*(1-Sr);
cpcval(pcval<0) = 0;

end
```

mythomas

```
function x = mythomas(a,b,c,r)

% L-U Decomposition
J = length(b);
f(1,1) = b(1,1);
g(1,1) = c(1,1)/b(1,1);
e(1,1) = 0;
for j = 2:J
    e(j,1) = a(j,1);
    f(j,1) = b(j,1)-a(j,1)*g(j-1,1);
    g(j,1) = c(j,1)/f(j,1);
end
% solving [L]y = r for y
y(1,1) = r(1,1)/f(1,1);
for j = 2:J
    y(j,1) = (r(j,1)-e(j,1)*y(j-1,1))/f(j,1);
end
%solving [U]x=y for x
x(J,1) = y(J,1);
for j = J-1:-1:1
    x(j,1) = y(j,1)-g(j,1)*x(j+1,1);
end
```

Incompressible Flow subjected to Constant Pressure Boundaries

Main Code

```
clear
close all
clc

% units: s, m, N, Pa
% define globals

global Ksat N M mu_w mu_nw p0 phi Sr flag

Ksat = 1e-09;                                % m^2
N = 1.6;
M = 1-1/N;
mu_w = 0.89e-03;                             % kg/m/s (water)
mu_nw = 1.03e-03;                            % kg/m/s (oil)
p0 = 1200;                                   % reference pressure (Pa)
qw0 = 3/(3600*1000*1000*pi*(.015)^2);        % flow (m/s)
qnw0 = 0;
phi = .35;
Sr = .1;

flag = 2;                                    % flag 1: constant Pressure BC
                                             % flag 2: constant Flow BC

% Temporal Discretization
dt = .001;                                   % seconds
```

```
tend = 3*3600 + 100;                        % seconds
t = dt:dt:tend;

% Spatial Discretization
dx = 0.0001;                                % m
xend = 0.4;                                 % 0.4m long domain
x = 0:dx:xend;
J = length(x)*2;                            % total number of unknowns
K = length(x);                              % total number of nodes

% Define initial pressure at the inflow
leftPc = 10135-10000;                       % Pa
leftPw = 10000;                             % Pa

% Define initial pressure throughout the sample
initPc = 10135-3000;                        % Pa
initPw = 3000;                              % Pa
Pc0 = initPc*ones(K,1);                          % length K
Pw0 = initPw*ones(K,1);                          % length K

% calculate Sw
Sw0(1:K,1) = Swfuns_vec(Pc0(1:K,1));            % length K


% assign old time step values for the first timestep
Pcold = Pc0;                                     % length K
Pwold = Pw0;                                     % length K
Swold = Sw0;                                     % length K


% Time Loop
nt = length(t);
for it = 1:nt
    time = (it-1)*dt;

    % Initial Guess
    Pc = Pcold;
    Pw = Pwold;
    Sw = Swold;

    % Set Pc and Pw at the both ends of the sample based on B.C.
    switch flag
        case 1
            Pc(1,1) = leftPc;
            Pw(1,1) = leftPw;
            Pc(K,1) = initPc;
            Pw(K,1) = initPw;
        case 2
            Pw(K,1) = initPw;
            Pc(K,1) = initPc;
    end

    % Initialization for Picard iteration
    iter = 0;
    error = 999;

    % Picard Iteration
```

```
while (error > 1.e-4) && (iter < 100)
    % Evaluate nonlinear functions Krnw(Sw), Krw(Sw) at every node
    Krw(1:K,1) = Krwfuns_vec(Sw(1:K,1));        % length K
    Krnw(1:K,1) = Krnwfuns_vec(Sw(1:K,1));      % length K

    % Evaluate cfuns (dSw/dPc)
    C(1:K,1) = cfuns_vec(Pc(1:K,1));            % length K


    % Form Aw and Anw at sets of 2 nodes
    Aw(1:K,1) = Ksat*Krw(1:K,1)/mu_w;           % length K
    Anw(1:K,1) = Ksat*Krnw(1:K,1)/mu_nw;        % length K

    % Take harmonic average of Aw and Anw at every node
    Awmean(1:K-1,1) = 2./(1./Aw(1:K-1,1)+1./Aw(2:K,1));
    Awminus = [0; Awmean];
    Awplus = [Awmean; 0];
    Anwmean(1:K-1,1) = 2./(1./Anw(1:K-1,1)+1./Anw(2:K,1));
    Anwminus = [0; Anwmean];
    Anwplus = [Anwmean; 0];

    % Form mat
    mat(1:2:J-1,1) = Pc;
    mat(2:2:J,1) = Pw;

    % Indexing
        % odd nodes: solve for dPc
        % even nodes: solve for dPw

    a(1:3) = 0;                                 % outside of matrix
    a(5:2:J-3) = 0;                             % Pc eqn
    a(4:2:J-2) = 1/dx^2*Anwminus(2:1:K-1);      % Pw eqn
    a(J-1:J) = 0;                               % B.C.

    b(1:2) = 0;                                 % outside of matrix
    b(3:2:J-3) = 0;                             % Pc eqn
    b(4:2:J-2) = 1/dx^2*Anwminus(2:1:K-1);      % Pw eqn
    b(J-1:J) = 0;                               % B.C.

    c(1) = 0;                                   % outside of matrix
    c(2) = 0;                                   % B.C.
    c(3:2:J-3) =  1/dx^2*Awminus(2:1:K-1);      % Pc eqn
    c(4:2:J-2) = phi/dt*C(2:1:K-1) - 1/dx^2*(Anwplus(2:1:K-1) ...
        + Anwminus(2:1:K-1));                   % Pw eqn
    c(J-1:J) = 0;                               % B.C.


    switch flag
        case 1
            d(1:2) = -1;                        % B.C.
        case 2
            d(1) = 0;
            d(2) = Anwplus(1,1)*1/dx;
    end
    d(3:2:J-3) = -phi/dt*C(2:1:K-1);
    d(4:2:J-2) =  -1/dx^2*(Anwplus(2:1:K-1) + Anwminus(2:1:K-1));
    d(J-1:J) = -1;                              % B.C.
```

```
switch flag
    case 1
        e(1:2) = 0;                         % B.C.
    case 2
        e(1) = Awplus(1,1)*1/dx;
        e(2) = Anwplus(1,1)*1/dx;
end
e(3:2:J-3) = -1/dx^2*(Awplus(2:1:K-1) + Awminus(2:1:K-1));
e(4:2:J-2) = 1/dx^2*Anwplus(2:1:K-1);
e(J-1) = 0;                                 % B.C.
e(J) = 0;                                   % outside of matrix


switch flag
    case 1
        f(1:2) = 0;                         % B.C.
    case 2
        f(1) = 0;
        f(2) =  -Anwplus(1,1)*1/dx;
end
f(3:2:J-3) = 0;                             % Pc eqn
f(4:2:J-2) = 1/dx^2*Anwplus(2:1:K-1);       % Pw eqn
f(J-1:J) = 0;                               % outside of matrix

switch flag
    case 1
        g(1:2) = 0;                         % B.C.
    case 2
        g(1) = -Awplus(1,1)*1/dx;
        g(2) = -Anwplus(1,1)*1/dx;
end
g(3:2:J-3) = 1/dx^2*Awplus(2:1:K-1);        % Pc eqn
g(4:2:J-4) = 0;                             % Pw eqn
g(J-2:J) = 0;                               % outside of matrix


switch flag
    case 1
        r(1) = 0;                           % left B.C. (Pc eqn)
        r(2) = 0;                           % left B.C. (Pw eqn)
    case 2
        r(1) = qw0 +  1/dx*Awplus(1,1).*(Pw(2)-Pw(1));
        r(2) = qnw0 + 1/dx*Anwplus(1,1).*(Pc(2)+Pw(2)-Pc(1)-Pw(1));
end
r(3:2:J-3) = phi*(Sw(2:1:K-1)-Swold(2:1:K-1))/dt - ...
    1/dx^2*Awplus(2:1:K-1).*(Pw(3:1:K)-Pw(2:1:K-1)) + ...
    1/dx^2*Awminus(2:1:K-1).*(Pw(2:1:K-1)-Pw(1:1:K-2)); % Pc eqn
r(4:2:J-2) = -phi*(Sw(2:1:K-1)-Swold(2:1:K-1))/dt - ...
    1/dx^2*Anwplus(2:1:K-1).*(Pw(3:1:K)+Pc(3:1:K)-...
    Pw(2:1:K-1)-Pc(2:1:K-1)) + 1/dx^2*Anwminus(2:1:K-1)...
    .*(Pw(2:1:K-1)+Pc(2:1:K-1)-Pw(1:1:K-2)-Pc(1:1:K-2)); % Pw eqn
r(J-1) = 0;                                 % right B.C. (Pc eqn)
r(J) = 0;                                   % right B.C. (Pw eqn)

R = r';                             % transpose to solve for del
```

```
    m = diag(a(4:J),-3) + diag(b(3:J),-2) + diag(c(2:J),-1) + ...
        diag(d) + diag(e(1:J-1),1) + diag(f(1:J-2),2) + ...
        diag(g(1:J-3),3);

    del = m\R;
    error=max(abs(del));
    mat =  mat + del;

    mat(mat<0)=0;

    Pc = mat(1:2:J-1,1);                        % length K
    Pw = mat(2:2:J,1);                          % length K
    Sw=Swfuns_vec(Pc);                          % length K

    iter=iter+1;

end

qw = -Awmean.*(Pw(2:K)-Pw(1:K-1))/dx;
qnw = -Anwmean.*((Pw(2:K)-Pw(1:K-1))+(Pc(2:K)-Pc(1:K-1)))/dx;
qt = qw + qnw;

% Update
Pcold = Pc;
Pwold = Pw;
Swold = Sw;
Pnw = Pc + Pw;

    % save variables
     if time == 3600
         Sw3600 = Sw1;
         Pc3600 = Pc;
         Pw3600 = Pw;
         Pnw3600 = Pnw;
         qw3600 = qw;
         qnw3600 = qnw;
     end

     if time == 2*3600
         Sw7200 = Sw1;
         Pc7200 = Pc;
         Pw7200 = Pw;
         Pnw7200 = Pnw;
         qw7200 = qw;
         qnw7200 = qnw;
     end

     if time == 3*3600
         Sw10800 = Sw1;
         Pc10800 = Pc;
         Pw10800 = Pw;
         Pnw10800 = Pnw;
         qw10800 = qw;
         qnw10800 = qnw;
     end

% Plotting for Animation
```

```
 % Plotting Sw
 subplot(7,1,1)
 plot(x, Sw)
 xlim([0 xend])
 ylim([0 1.1])
 title('Coupled Solver Results')
 ylabel('S_w')
 xlabel('x (m)')

 % Plotting Pc
 subplot(7,1,2)
 plot(x, Pc)
 xlim([0 xend])
ylabel('P_c')
 xlabel('x (m)')

 % plotting Pc
 subplot(7,1,3)
 plot(x,Pw)
 xlim([0 xend])
 ylabel('P_w')
 xlabel('x (m)')

 % Plotting Pnw
 subplot(7,1,4)
 plot(x, Pnw)
 ylabel('P_n_w')
 xlabel('x (m)')

 % Plotting Qw
 subplot(7,1,5)
 plot(x(1:K-1),qw)
 ylabel('Q_w')
 xlabel('x (m)')

 % Plotting Qnw
 subplot(7,1,6)
 plot(x(1:K-1),qnw)
 ylabel('Q_n_w')
 xlabel('x (m)')

 % Plotting Qtot
 subplot(7,1,7)
 plot(x(1:K-1),qnw+qw)
 ylabel('Q_t_o_t')
 xlabel('x (m)')

 % Print
 [t(it), iter]

 if (iter >= 100)
     disp 'no convergence'
     return
 end

 % Refresh
 pause(0.001)%to refresh
```

```
end
```

Swfuns

```
function Swval = Swfuns_vec(Pcval)
% given the capillary pressure determine the degree of saturation of the
% wetting fluid

global N M p0 Sr
Swval = 1./(1+(Pcval/p0).^N).^M.*(1-Sr)+Sr;

end
```

Krwfuns

```
function [Krw] = Krwfuns_vec(Swval)
% given the degree of saturation of the wetting fluid determine the
% relative conductivity of the wetting fluid

global M

Krw = sqrt(Swval).*(1-(1-Swval.^(1/M)).^M).^2;

end
```

Krnwfuns

```
function [Krnw] = Krnwfuns_vec(Swval)
% given the degree of saturation of the wetting fluid determine the
% relative conductivity of the wetting fluid

global M
Krnw = sqrt(1-Swval).*(1-(1-(1-Swval).^(1./M)).^M).^2;

end
```

cfuns

```
function cval = cfuns_vec(pcval)
% change in saturation over change in pressure

global N p0 Sr
cval = (-N+1)*((pcval/p0).^N+1).^(-2+1/N).*(pcval/p0).^(N-1)/p0.*(1-Sr);
cval(pcval<0) = 0;

end
```

Compressible Flow Subjected to Varying Boundary Conditions (Picard)

Main Code

```
clear all;
close all;
```

```
clc;

% define globals
global flag Ksat N M mu_w mu_nw p0 phi Sr Srnw Cw rhow0 rhonw0 Pw0 Pnw0 grav

Ksat = 1e-11;                                   % m^2
N = 2.35;
M = 1-1/N;
mu_w = 0.89e-03;                                % kg/m/s (water)
mu_nw = 1.81e-05;                               % kg/m/s (air)
p0 = 2100;                                      % reference pressure (Pa)
phi = .37;                                      % calibrated porosity
Sr = .085;
Srnw = .25;
Cw = 5.1e-10;                                   % 1/Pa
rhow0 = 1000;                                   % kg/m3 (water)
rhonw0 = 1.225;                                 % kg/m3 (air)
grav = 9.81;                                    % m/s2

flag = 8;                      % flag 1: Pw0 and Pnw0 constant at left B.C.
                               % flag 2: qw0 and Pnw0 constant at left B.C.
                               % flag 7: qw0 and qnw0 constant at left B.C.
                               % flag 8: qw0 and qnw0 constant at left and
right
qw0 = 20/(3600*100);                            % flow (m/s) % given
qwJ = 0;
qnw0 = 0;


% Temporal Discretizaion
dt = 0.1;                                       % seconds
tend = 1441;                                    % seconds
t = dt:dt:tend;

% Spatial Discretizaion
dx = 0.01; % m
xend = 93.5;                                    % 0.935 long domain
x = 0:dx:xend;
K = length(x);                                  % total number of nodes
J = length(x)*2;                                % total number of unknowns

% define initial pressures at the left boundary and throughout the sample
leftPw = 10000;
leftPnw = 10135-rhonw0*grav*xend;
leftPc = leftPnw-leftPw;
initPw = 10000; % adjusted this to fit initial curve of Sw
initPnw = 10135;
initPc = initPnw-initPw;

Pw0 = linspace(initPw-rhow0*grav*xend,initPw,K)';
Pnw0 = linspace(leftPnw,initPnw,K)';
Pc0 = Pnw0-Pw0;


% calculate Sw based on Pc0
Sw0(1:K,1) = Swfuns_vec(Pc0(1:K,1));
```

```
% assign old time step values for the first timestep
Pwold = Pw0;                                                % length K
Pnwold = Pnw0;                                              % length K
Pcold = Pc0;                                                % length K
Swold = Sw0;                                                % length K

% Time Loop
nt=length(t);
for it=1:nt
    time=(it-1)*dt;

    % initial guess
    Pw = Pwold;
    Pnw = Pnwold;
    Pc = Pcold;
    Sw = Swold;                                             % length K

    % Left B.C.
    switch flag
        case 1
            Pw(1,1) = leftPw;
            Pnw(1,1) = leftPnw;
            Pc(1,1)=leftPc;
            Pw(K,1)=initPw;
            Pnw(K,1) = initPnw;
            Pc(K,1) = initPc;
        case 2
            Pnw(1,1) = leftPnw;
            Pc(1,1) = leftPc;
            Pw(K,1)=initPw;
            Pnw(K,1) = initPnw;
            Pc(K,1) = initPc;

        case 7
            Pnw(1,1) = leftPnw;
            Pc(1,1) = leftPc;
            Pc(K,1) = initPc;
        case 8
            Pc(1,1) = leftPc;
            Pc(K,1) = initPc;
    end


    % Initialization for Picard iteration
    iter = 0;
    error = 999;

    % Picard Iteration
    while (error > 1.e-4) && (iter < 100)

        % evaluate nonlinear functions Krnw(Sw), Krw(Sw) at every node
        Krw(1:K,1) = Krwfuns_vec(Sw(1:K,1));                % length K
        Krnw(1:K,1) = Krnwfuns_vec(Sw(1:K,1));              % length K

        % evaluate nonlinear functions rhow(Pw), rhonw(Pnw) at every node
        rhow(1:K,1) = rhowfuns(Pw(1:K,1));
        rhonw(1:K,1) = rhonwfuns(Pnw(1:K,1));
        % take harmonic average of rhow and rhonw
        rhowmean(1:K-1,1) = 2./(1./rhow(1:K-1,1)+1./rhow(2:K,1));
        rhowminus = [0;rhowmean];
        rhowplus = [rhowmean;0];
        rhonwmean(1:K-1,1) = 2./(1./rhonw(1:K-1,1)+1./rhonw(2:K,1));
        rhonwminus = [0;rhonwmean];
        rhonwplus = [rhonwmean;0];

        % Evaluate cfuns (dSw/dPc)
        C(1:K,1) = cfuns_vec(Pnw(1:K,1)-Pw(1:K,1));           % length K

        % Form Aw and Anw (dP/drho)                           % length K
        Aw(1:K,1) = rhow0*Cw./(1-Cw.*(Pw-Pw0)).^2;
        Anw(1:K,1) =  rhonw0./Pnw0;

        % Form Bw and Bnw                                     % length K
        Bw(1:K,1) = Ksat*Krw(1:K,1)/mu_w;
        Bnw(1:K,1) = Ksat.*Krnw(1:K,1)/mu_nw;
        % take harmonic average of Bw and Bnw at every node
        Bwmean(1:K-1,1) = 2./(1./Bw(1:K-1,1)+1./Bw(2:K,1));
        Bwminus = [0;Bwmean];
        Bwplus = [Bwmean;0];
        Bnwmean(1:K-1,1) = 2./(1./Bnw(1:K-1,1)+1./Bnw(2:K,1));
        Bnwminus = [0;Bnwmean];
        Bnwplus = [Bnwmean;0];

        % Form mat
        mat(1:2:J-1,1) = Pw;
        mat(2:2:J,1) = Pnw;



        % Form vectors a,b,c,d,e,f,g,R
        a(5:2:J-3) = 0;                 % nature of matrix
        a(4:2:J-2) = 0;                 % no dPwj-1 term in eqn 2

        b(3:2:J-3) = -1/dx^2*rhow(2:1:K-1).*Bwminus(2:1:K-1); ...
            % dPwj-1 in eqn 1
        b(4:2:J-2) = -1/dx^2*rhonwminus(2:1:K-1).*Bnwminus(2:1:K-1); ...
            % dPnwj-1 in eqn 2

        c(3:2:J-3) = 0;                                   % no dPnwj-1 in eqn
1
        c(4:2:J-2) = 1/dt*phi*rhonw(2:1:K-1).*C(2:1:K-1); % dPwj in eqn 2


        switch flag  % B.C.
            case 1                      % Pw0 and Pnw0 constant
                a(1:3) = 0;             % outside of matrix
                a(J-1:J) = 0;

                b(1:2) = 0;             % outside of matrix
                b(J-1:J) = 0;

                c(1) = 0;               % outside of matrix
                c(2) = 0;
                c(J-1:J) = 0;
                d(1:2) = 1;
                d(J-1:J) = 1;

                e(1:2) = 0;
                e(J-1) = 0;
                e(J) = 0;               % outside of matrix

                f(1:2) = 0;
                f(J-1:J) = 0;           % outside of matrix

                g(1:2) = 0;
                g(J-2:J) = 0;           % outside of matrix

                r(1:2) = 0;
                r(J-1:J) = 0;
            case 2
                a(1:3) = 0;             % outside of matrix
                a(J-1:J) = 0;

                b(1:2) = 0;             % outside of matrix
                b(J-1:J) = 0;

                c(1) = 0;               % outside of matrix
                c(2) = 0;
                c(J-1:J) = 0;

                d(1) = -1/dx*Bwplus(1,1);
                d(2) = 1;
                d(J-1:J) = 1;

                e(1:2) = 0;
                e(J-1) = 0;
                e(J) = 0;               % outside of matrix

                f(1) = 1/dx*Bwplus(1,1);
                f(2) = 0;
                f(J-1:J) = 0;           % outside of matrix

                g(1:2) = 0;
                g(J-2:J) = 0;           % outside of matrix

                r(1) = qw0 + Bwplus(1,1)*((Pw(2)-Pw(1))/dx-
                        rhowplus(1,1)*grav);
                r(2) = 0;
                r(J-1:J) = 0;
            case 4
                a(1:3) = 0;             % outside of matrix
                a(J-1:J) = 0;

                b(1:2) = 0;             % outside of matrix
                b(J-1:J) = 0;

                c(1) = 0;               % outside of matrix
                c(2) = 0;
                c(J-1:J) = 0;
                d(1) = -1/dx*Bwplus(1,1);
                d(2) = -1/dx*Bnwplus(1,1);
                d(J-1:J) = 1;

                e(1:2) = 0;
                e(J-1) = 0;
                e(J) = 0;               % outside of matrix

                f(1) = 1/dx*Bwplus(1,1);
                f(2) = 1/dx*Bnwplus(1,1);
                f(J-1:J) = 0;           % outside of matrix

                g(1:2) = 0;
                g(J-2:J) = 0;           % outside of matrix

                r(1) = qwJ + Bwplus(1,1)*((Pw(2)-Pw(1))/dx-...
                    rhowplus(1,1)*grav);
                r(2) = qnw0 + Bnwplus(1,1)*((Pnw(2)-Pnw(1))/dx-...
                    rhonwplus(1,1)*grav);
                r(J-1:J) = 0;
            case 7 % ct qw0 qnw0 at inflow, ct pw0 and qnw0 at outflow
                a(1:3) = 0;             % outside of matrix
                a(J-1:J) = 0;

                b(1:2) = 0;             % outside of matrix
                b(J-1) = 0;
                b(J) = -1/dx*Bnwminus(K);

                c(1) = 0;               % outside of matrix
                c(2) = 0;
                c(J-1:J) = 0;

                d(1) = -1/dx*Bwplus(1,1);
                d(2) = -1/dx*Bnwplus(1,1);
                d(J-1) = 1;
                d(J) = 1/dx*Bnwminus(K,1);

                e(1:2) = 0;
                e(J-1) = 0;
                e(J) = 0;               % outside of matrix

                f(1) = 1/dx*Bwplus(1,1);
                f(2) = 1/dx*Bnwplus(1,1);
                f(J-1:J) = 0;           % outside of matrix

                g(1:2) = 0;
                g(J-2:J) = 0;           % outside of matrix

                r(1) = qw0 + Bwplus(1,1)*((Pw(2)-Pw(1))/dx-...
                    rhowplus(1,1)*grav);
                r(2) = qnw0 + Bnwplus(1,1)*((Pnw(2)-Pnw(1))/dx-...
                    rhonwplus(1,1)*grav);
                r(J-1) = 0; % B.C.
                r(J) = qnw0 + Bnwminus(K,1)*((Pnw(K)-Pnw(K-1))/dx-...
                    rhonwminus(K,1)*grav);
            case 8 % ct qw0 qnw0 at inflow, ct pw0 and qnw0 at outflow
                a(1:3) = 0;             % outside of matrix
```

```
                a(J-1:J) = 0;

                b(1:2) = 0;             % outside of matrix
                b(J-1) = -1/dx*Bwminus(K);
                b(J) = -1/dx*Bnwminus(K);

                c(1) = 0;               % outside of matrix
                c(2) = 0;
                c(J-1:J) = 0;

                d(1) = -1/dx*Bwplus(1,1);
                d(2) = -1/dx*Bnwplus(1,1);
                d(J-1) = 1/dx*Bwminus(K,1);
                d(J) = 1/dx*Bnwminus(K,1);

                e(1:2) = 0;
                e(J-1) = 0;
                e(J) = 0;                   % outside of matrix

                f(1) = 1/dx*Bwplus(1,1);
                f(2) = 1/dx*Bnwplus(1,1);
                f(J-1:J) = 0;               % outside of matrix

                g(1:2) = 0;
                g(J-2:J) = 0;               % outside of matrix

                r(1) = qw0 + Bwplus(1,1)*((Pw(2)-Pw(1))/dx-...
                    rhowplus(1,1)*grav);
                r(2) = qnw0 + Bnwplus(1,1)*((Pnw(2)-Pnw(1))/dx-...
                    rhonwplus(1,1)*grav);
                r(J-1) = qwJ + Bwminus(K,1)*((Pw(K)-Pw(K-1))/dx-...
                    rhowminus(K,1)*grav);
                r(J) = qnw0 + Bnwminus(K,1)*((Pnw(K)-Pnw(K-1))/dx-...
                    rhonwminus(K,1)*grav);
        end

        d(3:2:J-3) = -1/dt*phi*rhow(2:1:K-1).*C(2:1:K-1) + 1/dt*phi*...
            Sw(2:1:K-1).*Aw(2:1:K-1)+ 1/dx^2*rhow(2:1:K-1).*...
            (Bwplus(2:1:K-1)+Bwminus(2:1:K-1));     % dPwj in eqn 1
        d(4:2:J-2) =  -1/dt*phi*rhonw(2:1:K-1).*C(2:1:K-1) +...
            1/dt*phi*(1-Sw(2:1:K-1)).*Anw(2:1:K-1) + 1/dx^2*...
            (rhonwplus(2:1:K-1).*Bnwplus(2:1:K-1)+rhonwminus(2:1:K-1).*...
            Bnwminus(2:1:K-1));                     % dPnwj in eqn 2


        e(3:2:J-3) = 1/dt*phi*rhow(2:1:K-1).*C(2:1:K-1);  % dPnwj in eqn 1
        e(4:2:J-2) = 0;                                   % no dPwj+1 in eqn
2

        f(3:2:J-3) = -1/dx^2*rhow(2:1:K-1).*Bwplus(2:1:K-1);
            % dPwj+1 in eqn 1
        f(4:2:J-2) = -1/dx^2*rhonwplus(2:1:K-1).*Bnwplus(2:1:K-1);
            % dPnwj+1 in eqn 2

        g(3:2:J-3) = 0; % no dPnwj+1 in eqn 1
        g(4:2:J-4) = 0; % nature of matrix
        r(3:2:J-3) = 1/dt*phi*rhow(2:1:K-1).*(Sw(2:1:K-1)-Swold(2:1:K-1))...
            + 1/dt*phi*Sw(2:1:K-1).*Aw(2:1:K-1).*(Pw(2:1:K-1)-...
            Pwold(2:1:K-1)) - 1/dx*rhow(2:1:K-1).*(Bwplus(2:1:K-1).*...
            ((mat(5:2:J-1)-mat(3:2:J-3))/dx-rhowplus(2:1:K-1)*grav)-...
            Bwminus(2:1:K-1).*((mat(3:2:J-3)-mat(1:2:J-5))/dx-...
            rhowminus(2:1:K-1)*grav));                  % eqn 1
        r(4:2:J-2) = -1/dt*phi*rhonw(2:1:K-1).*(Sw(2:1:K-1)-Swold(2:1:K-
1))...
            + 1/dt*phi*(1-Sw(2:1:K-1)).*Anw(2:1:K-1).*(Pnw(2:1:K-1)-...
            Pnwold(2:1:K-1)) - 1/dx*(rhonwplus(2:1:K-1).*Bnwplus(2:1:K-1)...
            .*((mat(6:2:J)-mat(4:2:J-2))/dx-rhonwplus(2:1:K-1)*grav)-...
            rhonwminus(2:1:K-1).*Bnwminus(2:1:K-1).*((mat(4:2:J-2)-...
            mat(2:2:J-4))/dx-rhonwminus(2:1:K-1)*grav)); % eqn 2

        R = r'; % transpose to solve for del

        m = diag(a(4:J),-3) + diag(b(3:J),-2) + diag(c(2:J),-1) + diag(d)...
            + diag(e(1:J-1),1) + diag(f(1:J-2),2) + diag(g(1:J-3),3);

        del = m\-R;
        error = max(abs(del));
        mat =  mat+del;

        Pw = mat(1:2:J-1,1);                                % length K
        Pnw = mat(2:2:J,1);                                 % length K
        Pc = mat(2:2:J,1)-mat(1:2:J-1,1);                   % length K
        Pc(Pc<0) = 0;
        Sw = Swfuns_vec(Pc);                                  % length K

        iter = iter + 1;
    end

    % Update
    Pwold = Pw;
    Pnwold = Pnw;
    Pcold = Pc;
    Swold = Sw;

    % Save Variables
    if time == 0.05*3600
        Sw05 = Sw;
    end

    if time == 0.10*3600
        Sw10 = Sw;
    end

    if time == 0.15*3600
       Sw15 = Sw;
    end

    if time == 0.20*3600
        Sw20 = Sw;
    end

    if time ==0.25*3600
        Sw25 = Sw;
```

```
end

if time == 0.3*3600
    Sw30 = Sw;
end

if time == 0.35*3600
    Sw35 = Sw;
end
if time == 0.4*3600
    Sw40 = Sw;
end

% Calculate Flux
qw = -Bwmean.*((Pw(2:K)-Pw(1:K-1))/dx - rhowmean*grav);
qnw = -Bnwmean.*((Pnw(2:K)-Pnw(1:K-1))/dx - rhonwmean*grav);
qt = qw + qnw;

% Animated Plots
% Plotting Sw
subplot(6,1,1)
plot(x,Sw)
xlim([0 xend])
ylim([0 1.1])
title('Coupled Solver Results (Compressibility)')
ylabel('S_w')
xlabel('x (m)')

% Plotting Pw
subplot(6,1,2)
plot(x, Pw)
xlim([0 xend])
ylabel('P_w')
xlabel('x (m)')

% Plotting Pnw
subplot(6,1,3)
plot(x, Pnw)
xlim([0 xend])
ylabel('P_n_w')
xlabel('x (m)')

% Plotting Pc
subplot(6,1,4)
plot(x, Pc)
xlim([0 xend])
ylabel('P_c')
xlabel('x (m)')

% Plotting Qw
subplot(6,1,5)
plot(x(1:K-1),qw)
ylabel('Q_w')
xlabel('x (m)')

% Plotting Qnw
subplot(6,1,6)
```

```
    plot(x(1:K-1),qnw)
    ylabel('Q_n_w')
    xlabel('x (m)')

    % Print
    [t(it), iter]

    if (iter >= 100)
        disp 'no convergence'
        return
    end

    % Refresh
    pause(0.001)
end
```

Swfuns

```
function Swval = Swfuns_vec(Pcval)
% given the capillary pressure determine the degree of saturation of the
% wetting fluid

global N M p0 Sr Srnw
Swval = 1./(1+(Pcval/p0).^N).^M.*(1-Sr-Srnw)+Sr;

end
```

cfuns

```
function cval = cfuns_vec(pcval)
% change in saturation over change in pressure

global N p0 Sr Srnw
cval = (-N+1)*((pcval/p0).^N+1).^(-2+1/N).*(pcval/p0).^(N-1)/p0.*(1-Sr-Srnw);
cval(pcval<0) = 0;

end
```

Krwfuns

```
function [Krw] = Krwfuns_vec(Swval)
% given the degree of saturation of the wetting fluid determine the
% relative conductivity of the wetting fluid

global Ksat N M mu_w mu_nw p0 q phi Sr
Krw = sqrt(Swval).*(1-(1-Swval.^(1/M)).^M).^2;

end
```

Krnwfuns

```
function [Krnw] = Krnwfuns_vec(Swval)
% given the degree of saturation of the wetting fluid determine the
% relative conductivity of the wetting fluid
```

```
global Ksat N M mu_w mu_nw p0 q phi Sr
Krnw = sqrt(1-Swval).*(1-(1-(1-Swval).^(1./M)).^M).^2;

end
```

rhowfuns

```
function [rhow] = rhowfuns(Pwval)
% given the pressure of the wetting fluid, determine the density of the
% wetting fluid

global Ksat N M mu_w mw_nw p0 phi Sr Cw rhow0 rhonw0 Pw0 Pnw0
rhow = rhow0./(1-Cw.*(Pwval-Pw0));

end
```

rhonwfuns

```
function [rhonw] = rhonwfuns(Pnwval)
% given the pressure of the wetting fluid, determine the density of the
% wetting fluid

global rhonw0 Pnw0
rhonw = rhonw0.*(Pnwval./Pnw0);

end
```

Compressible Flow Subjected to Varying Boundary Conditions (RK4)

Main Code

```
clear all;
close all;
clc;

% define globals
global flag Ksat N M mu_w mu_nw p0 phi Sr Cw rhow0 rhonw0 Pw0 Pnw0 grav dx K
qw0 Srnw

Ksat= 1e-11;                                    % m^2
N = 2.35;
M = 1-1/N;
mu_w = 0.89e-03;                                % kg/m/s (water)
mu_nw = 1.81e-05;                               % kg/m/s (air)
p0 = 2100;                                      % reference pressure (Pa)
phi = .311;                                     % calibrated porosity
Sr = .05;
Srnw = 0.25;
Cw = 5.1e-10;                                   % 1/Pa
rhow0 = 1000;                                   % kg/m3 (water)
rhonw0 = 1.225;                                 % kg/m3 (air)
grav = 9.81;                                    % m/s2
```

```
flag = 1;                   % flag 1: Pw0 and Pnw0 constant at left B.C.
                            % flag 2: qw0 and Pnw0 constant at left B.C.


% Temporal Discretizaion
dt = 0.0001;                                    % seconds
tend = 0.05;                                    % seconds
t = 0:dt:tend;

% Spatial Discretizaion
dx = 0.01;                                      % m
xend = 0.935;                                   % 0.935m long domain
x = 0:dx:xend;
K = length(x);

% Initial Conditions
leftPw = 10000;
leftPnw = 10135-rhonw0*grav*xend;
leftPc = leftPnw-leftPw;
rightPw = 10000;
rightPnw = 10135;
rightPc = rightPnw-rightPw;
qw0 = 8/(3600*100); % flow (m/s)

Pw0 = linspace(rightPw-rhow0*grav*xend,rightPw,K)';
Pnw0 = linspace(leftPnw,rightPnw,K)';
Pc0 = Pnw0-Pw0;

% calculate Sw based on Pc0
Sw0(1:K,1) = Swfuns_vec(Pc0(1:K,1));

% assign old time step values for the first time step
Pw = Pw0;                                                   % length K
Pnw = Pnw0;                                                 % length K
Pc = Pc0;                                                   % length K
Sw = Sw0;                                                   % length K


% Substepping Control
TOL = 1e-04;
growth = 0.2;
dt_upper = 1;

t = 0;
reverseStr = '';
while t < tend

    switch flag % boundary conditions
        case 1
            Pw(1,1) = leftPw;
            Pnw(1,1) = leftPnw;
            Pc(1,1) = leftPc;
            Pw(K,1) = rightPw;
            Pnw(K,1) = rightPnw;
            Pc(K,1) = rightPc;
        case 2
            if t==0
            Pw(1,1) = leftPw;
        else
            Pw(1,1) = Pw(2,1)+(qw0/Bwmean(1,1)-rhowmean(1,1)*grav)*dx;
        end
        Pnw(1,1) = leftPnw;
        Pc(1,1) = leftPnw -Pw(1,1);
        Pw(K,1) = rightPw;
        Pnw(K,1) = rightPnw;
        Pc(K,1) = rightPc;
end


% RK4 (call EEFD)

[dPw1_dt,dPnw1_dt] = EFFD(Pw,Pnw);

[dPw2_dt,dPnw2_dt] = EFFD(Pw+dPw1_dt*dt/2,Pnw+dPnw1_dt*dt/2);

[dPw3_dt,dPnw3_dt] = EFFD(Pw+dPw2_dt*dt/2,Pnw+dPnw2_dt*dt/2);

[dPw4_dt,dPnw4_dt] = EFFD(Pw+dPw3_dt*dt,Pnw+dPnw3_dt*dt);

% defined for error
Pw_new1 = Pw+dPw1_dt*dt;
Pnw_new1 = Pnw+dPnw1_dt*dt;

% real value
Pw_new2 = Pw+1/6*(dPw1_dt+2*dPw2_dt+2*dPw3_dt+dPw4_dt)*dt;
Pnw_new2 = Pnw+1/6*(dPnw1_dt+2*dPnw2_dt+2*dPnw3_dt+dPnw4_dt)*dt;

% error and substepping
ERROR = max(max(abs(Pw_new2-Pw_new1)),max(abs(Pnw_new2-Pnw_new1)));

if ERROR > TOL
    % abandon this update, reduce time step
    dt = dt*abs(TOL/ERROR)^growth;
else

    % accept the update and plot variables
        % everything is defined to plot q variables
    Pw = Pw_new2;
    Pnw = Pnw_new2;
    Pc = Pnw - Pw;
    Sw = Swfuns_vec(Pc);

    rhow = rhowfuns(Pw);
    rhowmean(1:K-1,1) = 2./(1./rhow(1:K-1,1)+1./rhow(2:K,1));
    Krw = Krwfuns_vec(Sw);
    Bw = Ksat*Krw/mu_w;
    Bwmean(1:K-1,1) = 2./(1./Bw(1:K-1,1)+1./Bw(2:K,1));
    qw= - Bwmean.*((Pw(2:K)-Pw(1:K-1))/dx - rhowmean*grav);

    rhonw = rhonwfuns(Pnw);
    rhonwmean(1:K-1,1) = 2./(1./rhonw(1:K-1,1)+1./rhonw(2:K,1));
    Krnw = Krnwfuns_vec(Sw);
    Bnw = Ksat.*Krnw/mu_nw;
    Bnwmean(1:K-1,1) = 2./(1./Bnw(1:K-1,1)+1./Bnw(2:K,1));
```

```
qnw = - Bnwmean.*(((Pw(2:K)-Pw(1:K-1))+(Pc(2:K)-Pc(1:K-1)))/dx - ...
    rhonwmean*grav);

% adjust time step
dt = dt*abs(TOL/ERROR)^growth;
dt = min(dt,dt_upper); % dt will never be larger than 1 second
t = t+dt;

% Animated Plots
% Plotting Sw
subplot(6,1,1)
plot(x,Sw)
xlim([0 xend])% x-axis range
ylim([0 1.1])
title('Coupled Solver Results (Compressibility)')
ylabel('S_w')
xlabel('x (m)')

% Plotting Pw
subplot(6,1,2)
plot(x, Pw)
xlim([0 xend])
ylabel('P_w')
xlabel('x (m)')

% Plotting Pnw
subplot(6,1,3)
plot(x, Pnw)
xlim([0 xend])
ylabel('P_n_w')
xlabel('x (m)')

% Plotting Pc
subplot(6,1,4)
plot(x, Pc)
xlim([0 xend])
ylabel('P_c')
xlabel('x (m)')

% Plotting Qw
subplot(6,1,5)
plot(x(1:K-1),qw)
ylabel('Q_w')
xlabel('x (m)')

% Plotting Qnw
subplot(6,1,6)
plot(x(1:K-1),qnw)
ylabel('Q_n_w')
xlabel('x (m)')

% Print
msg = sprintf('Current time: %d',t);
fprintf([reverseStr, msg]);
reverseStr = repmat(sprintf('\b'), 1, length(msg));

% Refresh
```

```
        pause(0.0001)
    end
end
```

EFFD

```
function [dPw_dt,dPnw_dt]=EFFD(Pw,Pnw)
% Euler Forward Finite Discretization
% solves for dPw/dt and dPnw/dt given Pw and Pnw vectors

global flag Ksat mu_w mu_nw  phi Cw rhow0 rhonw0 Pw0 Pnw0 grav dx K

Pc = Pnw-Pw;
Sw = Swfuns_vec(Pc);

rhow = rhowfuns(Pw);
rhowmean(1:K-1,1) = 2./(1./rhow(1:K-1,1)+1./rhow(2:K,1));
rhowminus = [0;rhowmean];
rhowplus = [rhowmean;0];
rhonw = rhonwfuns(Pnw);
rhonwmean(1:K-1,1) = 2./(1./rhonw(1:K-1,1)+1./rhonw(2:K,1));
rhonwminus = [0;rhonwmean];
rhonwplus = [rhonwmean;0];

C = cfuns_vec(Pc);

Aw = rhow0*Cw./(1-Cw.*(Pw-Pw0)).^2;
Anw =  rhonw0./Pnw0;

Krw = Krwfuns_vec(Sw);
Bw = Ksat*Krw/mu_w;
Bwmean(1:K-1,1) = 2./(1./Bw(1:K-1,1)+1./Bw(2:K,1));
Bwminus = [0;Bwmean];
Bwplus = [Bwmean;0];

Krnw = Krnwfuns_vec(Sw);
Bnw = Ksat.*Krnw/mu_nw;
Bnwmean(1:K-1,1) = 2./(1./Bnw(1:K-1,1)+1./Bnw(2:K,1));
Bnwminus = [0;Bnwmean];
Bnwplus = [Bnwmean;0];

switch flag
    case 1 % ct pressure
        dPw_dt(1,1) = 0;
        dPw_dt(K,1) = 0;
        dPnw_dt(1,1) = 0;
        dPnw_dt(K,1) = 0;
    case 2
        dPw_dt(1,1) = 0;
        dPw_dt(K,1) = 0;
        dPnw_dt(1,1) = 0;
        dPnw_dt(K,1) = 0;
end

m11 = phi*Sw(2:K-1).*Aw(2:K-1)-phi*rhow(2:K-1).*C(2:K-1);
m12 = phi*rhow(2:K-1).*C(2:K-1);
```

```
m21 = phi*rhonw(2:K-1).*C(2:K-1);
m22 = phi*(1-Sw(2:K-1)).*Anw(2:K-1)-phi*rhonw(2:K-1).*C(2:K-1);

r1 = rhow(2:K-1)./dx.*(Bwplus(2:K-1).*((Pw(3:K)-Pw(2:K-1))/dx-...
    rhowplus(2:K-1)*grav)-Bwminus(2:K-1).*((Pw(2:K-1)-Pw(1:K-2))/dx...
    -rhowminus(2:K-1)*grav));
r2 = 1/dx.*(rhonwplus(2:K-1).*Bnwplus(2:K-1).*((Pnw(3:K)-Pnw(2:K-1))/dx...
    -rhonwplus(2:K-1)*grav)-rhonwminus(2:K-1).*Bnwminus(2:K-1).*...
    ((Pnw(2:K-1)-Pnw(1:K-2))/dx-rhonwminus(2:K-1).*grav));

dPw_dt(2:K-1) = m22./(-m12.*m21+m11.*m22).*r1 - m12./(-
m12.*m21+m11.*m22).*r2;
dPnw_dt(2:K-1) = - m21./(-m12.*m21+m11.*m22).*r1 + m11./(-
m12.*m21+m11.*m22).*r2;

end
```

Swfuns

```
function Swval = Swfuns_vec(Pcval)
% given the capillary pressure determine the degree of saturation of the
% wetting fluid

global N M Sr Srnw
Swval = 1./(1+(Pcval/p0).^N).^M.*(1-Sr-Srnw)+Sr;

end
```

cfuns

```
function cval = cfuns_vec(pcval)
% change in saturation over change in pressure

global N M p0 Sr Srnw
cval = (-N+1)*((pcval/p0).^N+1).^(-2+1/N).*(pcval/p0).^(N-1)/p0.*(1-Sr-Srnw);
cval(pcval<0) = 0;

end
```

Krwfuns

```
function [Krw] = Krwfuns_vec(Swval)
% given the degree of saturation of the wetting fluid determine the
% relative conductivity of the wetting fluid

global M
Krw = sqrt(Swval).*(1-(1-Swval.^(1/M)).^M).^2;

end
```

Krnwfuns

```
function [Krnw] = Krnwfuns_vec(Swval)
% given the degree of saturation of the wetting fluid determine the
% relative conductivity of the wetting fluid

global M
Krnw = sqrt(1-Swval).*(1-(1-(1-Swval).^(1./M)).^M).^2;

end
```

rhowfuns

```
function [rhow] = rhowfuns(Pwval)
% given the pressure of the wetting fluid, determine the density of the
% wetting fluid

global Cw rhow0 Pw0
rhow = rhow0./(1-Cw.*(Pwval-Pw0));

end
```

rhonwfuns

```
function [rhonw] = rhonwfuns(Pnwval)
% given the pressure of the wetting fluid, determine the density of the
% wetting fluid

global rhonw0 Pnw0
rhonw = rhonw0.*(Pnwval./Pnw0);

end
```