An Extended Buckley-Leverett Model for Non-Newtonian Foam Injection

S. F. ter Haar

TU Delft

An Extended Buckley-Leverett Model for Non-Newtonian Foam Injection

by

S. F. ter Haar

to obtain the degree of Bachelor of Science

at the Delft University of Technology,

to be defended publicly on Tuesday January 23, 2018 at 9:30 AM.

Student number:	1901206	
Project duration:	November 13, 2017 – January 23, 2018	
Thesis committee:	Prof. dr. ir. W. Rossen,	TU Delft, supervisor
	MSc. ir. R. Salazar,	TU Delft
	Dr. ir. A.-C. Dieudonné,	TU Delft

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

TU Delft

Abstract

Scope: Surfactant-alternating-gas (SAG) is the preferred method of foam injection to improve sweep efficiency in enhanced-oil-recovery (EOR). Here, for the first time, fractional-flow theory is extended to include the shock for gas injection in the high-quality regime for radial flow in a non-Newtonian SAG process for shear-thinning and shear-thickening foams.

Methodology: To represent non-Newtonian behavior in the high-quality regime, the limiting water saturation for foam stability varies as superficial velocity decreases with radial distance from the well. We look at the interactions between the shock and the characteristics. The mobility control at the shock front and injectivity are examined. The system is compared to a Newtonian foam.

Results and conclusions: For shear-thinning foam, the foam front's dimensionless velocity decreases with time, while the characteristics accelerate and collide with the shock. As the foam front propagates, the mobility ratio and mobility control becomes more favorable. The injectivity decreases until breakthrough, then improves slightly.

For shear-thickening foam, dimensionless velocity of the foam front increases with time, while the shocks slow down. Mobility control worsens and injectivity improves as the foam propagates, even before breakthrough. For extremely shear-thickening foam, the near-wellbore region exhibited shear-thinning behavior. This has three causes: a shift from the high- to the low- quality regime, the extrapolation of $fmdry$ over a too large range, and the Namdar Zanganeh correction.

Recommendations: Future models should replace the shock with the colliding characteristic, instead of eliminating the characteristic. For shear-thickening foams, new characteristics should split off from the shock. Include the shear-thinning factor for the low-quality regime to check if the foam is still in the high-quality regime.

Acknowledgments

Dr. Rossen, I've enjoyed every moment of our work together for the past one-and-a-half years, and I look forward to continue our future research. Thank you for trusting me with this exciting project. I feel fortunate that this combined so many of my interests and challenged me to grow academically. You made this process so fulfilling. I'm grateful for your feedback, inspiring mentorship, and insights.

Rodrigo Salazar, I couldn't have asked for a better adviser. Thank you for the collaboration, fruitful discussions, and helpful advice.

I would like to thank my mother and father for all their sacrifices to give me opportunities they never had, my colleagues at the Institute of Environmental Sciences for the freedom to work while studying, and Mariana Marques for all the coffee breaks, jokes, LaTeX help, and Matlab trouble-shooting.

And lastly, to my organic chemistry professor, Dr. Johnson, who six years ago steered me onto this path. I don't have the right words to express my gratitude. Without your guidance, I would never have made it here.

Contents

List of Figures

Nomenclature

λ_i	mobility of phase i $[m^2/(Pa \cdot s)]$
λ_{ri}	relative mobility of phase i $[m^2/(Pa \cdot s)]$
λ_{rt}	total relative mobility $[m^2/(Pa \cdot s)]$
μ_i	viscosity of phase i [Pa $\cdot$ s]
∇p	potential difference, also known as potential gradient $[Pa/m]$
Φ	potential
ϕ	porosity
ρ	density $[kg/m^3]$
^{f}	denotes presence of foam
$_g$	gas phase
$_w$	water phase
A	area $[m^2]$
$epdry$	controls abruptness of foam collapse
$F2$	dryout factor from the STARS^{TM} foam model
f_w	fractional water flow
FM	foam-mobility reduction factor
$fmdry$	water saturation where foam weakens
$fmmob$	reference mobility reduction factor
h	height [m]
i	phase
k_i	absolute permeability of phase i [darcy]
k_i	permeability of phase i $[m^2]$
k_{rg_e}	end-point relative permeability of gas at S_{gr}
k_{ri}	relative permeability of phase i
k_{rw_e}	end-point relative permeability of water at S_{gr}
n_g	saturation exponent for gas

n_w	saturation exponent for water
P	pressure [Pa]
p_c	capillary pressure [Pa]
p_c^*	limiting capillary pressure [Pa]
P_D	dimensionless pressure
Q_i	volumetric flow rate of phase i $[m^3/s]$
r	radius [m]
r_e	external reservoir radius [m]
r_w	wellbore radius [m]
S_i	saturation of phase i
S_w^*	critical water saturation
S_{gr}	residual gas saturation
S_{wr}	residual water saturation
t_D	dimensionless time
u	volumetric flux [m/s]
u_i	superficial velocity of phase i $[m/s]$
u_t	total superficial velocity $[m/s]$
v_{S_w}	specific velocity at S_w
x_D	dimensionless position
EOR	enhanced oil recovery
I	outlet condition
J	injection condition
MOC	Method of Characteristics
SAG	surfactant-alternating-gas

1

Introduction

Oil and gas recovery leaves two-thirds of oil initially in place in the ground (Lake, 1989). This is due to poor sweep efficiency as a result of reservoir heterogeneity and interfacial forces. Enhanced oil recovery (EOR) can save time and energy while reducing the amount of necessary exploration and drilling. Gas injection improves sweep efficiency, but geological heterogeneity can cause fingering and gravity override. Foam, unlike gas, overcomes both challenges (Alvarez et al., 2001; Shan and Rossen, 2004; Rossen, 1996). Mobility determines the best injection process because it effects injectivity. Surfactant-alternating-gas (SAG) injection is the most preferable method for operational and sweep efficiency reasons (Shan and Rossen, 2004).

Foams are gas bubbles separated by surfactant-stabilized liquid films. Distinguishable into two regimes based on the gas fractional-flow, foam is either at high- or low- quality (Osterloh and Jante, 1992). Foam can exhibit Newtonian, shear-thinning, or shear-thickening behavior. The low-quality regime is characterized by gas mobility reduction and is either shear-thinning or Newtonian. The high-quality regime is characterized by the water saturation at which foam collapses, S_{wr}, and has been shown to be shear-thinning, shear-thickening, and Newtonian (Alvarez et al., 2001).

Fractional-flow theory is limited in its application, but is useful for analyzing foam displacements (Rossen et al., 2011). The Method of Characteristics, when applied to oil recovery, is called the Buckley-Leverett Theory (Buckley and Leverett, 1942). For a SAG process, this theory is better at describing injectivity than numerical methods (Leeftink et al., 2015). The Buckley-Leverett theory predicts that a spreading wave will follow a shock. The shock front provides mobility control at the leading edge of the foam bank, while the spreading wave near the injection well creates the conditions needed for favorable injectivity (Rossen et al., 2011).

Modeling non-Newtonian foam is difficult because foam reacts to the changing water saturation in the high-quality regime, and can collapse suddenly. The non-Newtonian nature of foam means that the mobility can change drastically (Leeftink et al., 2015).

Current models allow for non-Newtonian behavior in the low-quality regime, but not in the high-quality regime because the model can't include the changing S_{wr} with superficial velocity. Previous extensions of this theory include non-Newtonian behavior in the high-quality regime, but didn't solve for injectivity and looked at the near-wellbore region only after the shock left the region (Rossen et al., 2011). Given the importance of the shock in mobility control and injectivity, a model was proposed to solve for both the shock and the characteristics in the shear-thinning regime. It shows that injectivity is more advantageous than what the Newtonian models suggest (ter Haar, 2017). This paper builds on and continues the research of SAG injection for non-Newtonian foams, improving the model and extending it to include shear-thickening foams as well.

The paper begins in Chapter 2 by outlining the theory, starting with Darcy's Law, and ending with foam in a non-Newtonian SAG injection. Next in Chapter 3, our extended model is explained followed by an explanation of the methodology used. A Bentheimer sandstone is analyzed in Chapter 4 under the assumptions of a non-Newtonian SAG process (n=0.33, 1.34, 1.67) with a focus on overall injectivity and mobility control at the leading shock front. The paper ends in Chapter 5 with a conclusion and recommendations for further improvements to the model.

2

Theory

Chapter 2 covers the theory, starting with Darcy's law in Section 2.1, followed by the Corey Parameters in Section 2.2 and the mobility equations in Section 2.3. Foams are explained in Section 2.4. Afterwards the dimensionless parameters - position, time, and pressure - are introduced in Section 2.7. The mass balance is given in Section 2.8, which leads in to the Buckley-Leverett Theory in Section 2.9 and SAG process theory in Section 2.10.

2.1. Darcy's Law

2.1.1. Single-Phase Flow

Darcy's Law is an empirical relationship for fluid flow through a porous medium. Assuming Newtonian flow in phase i, the volumetric flux, u_i, of the phase is proportional to the volumetric flow rate, Q_i, divided by the radial flow area A_r. The volumetric flow rate is equal to the permeability, k_i, divided by the fluid's viscosity, μ_i, multiplied by the pressure gradient, $\frac{d\Phi}{d_r}$. The pressure gradient is also known as the potential gradient. This gives us the following relationship

$$u_i = \frac{Q_i}{A_r} = -\frac{k}{u_i}\frac{d\Phi}{d_r} = -\frac{k_a}{\mu_a}\frac{\partial}{\partial r}(P_a + \rho_i g Z) \tag{2.1}$$

We can ignore gravity in horizontal radial flow. Representing the pressure gradient now as ∇p, Darcy's law can be simplified to

$$u_i = \frac{Q}{A_r} = -\frac{k_i}{\mu_i}\nabla p \tag{2.2}$$

(Lake, 1989; Rossen, 1996).

Darcy's law assumes laminar flow with dominant viscous forces, i.e. a low Reynold's number. The fluid is assumed to be Newtonian, with no-slip boundary conditions on

solid surfaces within the media, and no chemical interactions between the fluid and porous media.

2.1.2. Multi-Phase Flow

Darcy's law can be extended for multi-phase flow under the assumption that all phases are simultaneously present, where all phases have their own interconnected pore network and can flow simultaneously. Each phase reduces the permeability of the other. The permeability reduction is represented by the relative permeability, k_r of phase i, which is a function of the phase's saturation.

$$\vec{u_i} = -\frac{kk_{ri}(S_i)}{\mu_i}\nabla p \tag{2.3}$$

(Rossen and Zhou, 1995).

2.2. Corey Parameters

General theoretical equations for relatively permeability functions do not (yet) exist, so empirical functions are used. For two-phase flow, such as gas and water, their respective relative permeabilities are described, respectively, by

$$k_{rg}(S_w) = k_{rg_e}\left(\frac{1 - S_w - S_{wr}}{1 - S_{wr} - S_{gr}}\right)^{n_g} \text{ where } S_{wr} \leq S_w \leq 1 - S_{gr} \tag{2.4}$$

and

$$k_{rw}(S_w) = k_{rw_e}\left(\frac{S_w - S_{wr}}{1 - S_{wr} - S_{gr}}\right)^{n_w} \text{ where } S_{gr} \leq S_g \leq 1 - S_{wr} \tag{2.5}$$

The end-point relative permeabilities are given by k_{rw_e} and k_{rg_e}, which depends on the wettability of the rock. The water saturation, residual water saturation, and residual gas saturation are written as S_w, S_{wr}, and S_{gr}, respectively. The saturation exponents for water, n_w, and gas, n_g, define the shape of the relative-permeability curves, where a higher exponent means that the change in relative permeability is less linear (Eftekhari & Farajzadeh, 2017).

2.3. Mobility

Mobility, λ, is the ease of movement through a porous medium with a pressure gradient. It is the ratio of effective permeability to phase viscosity; it represents the ratio between the pressure gradient and volumetric flux.

$$\lambda_i(S_w) = \frac{k_i(S_w)}{\mu_i} = \frac{kk_{ri}(S_w)}{\mu_i} = k\lambda_{ri}(S_w) \tag{2.6}$$

This can be substituted into Equation 2.3.

$$\vec{u_i} = -k\lambda_i(S_w)\nabla p \tag{2.7}$$

The total relative mobility, λ_{rt}, is the sum of the relative mobilities of both phases at a specific water saturation.

$$\lambda_{rt}(S_w) = \lambda_{rw}(S_w) + \lambda_{rg}(S_w) = \frac{k_{rw}(S_w)}{\mu_w} + \frac{k_{rg}(S_w)}{\mu_g} \tag{2.8}$$

(Rossen et al., 2011).

2.4. Foam

2.4.1. Foam Theory

Foam can redirect flow patterns by blocking swept zones, giving foam the potential to improve oil recovery by controlling mobility throughout the reservoir. A strong leading edge on a foam bank can be used to displace oil to collection wells (Rossen 1996).

The lamellae of the foam can break and collapse if the capillary pressure, p_c, becomes too high. This point, p_c^*, is called the limiting capillary pressure (Khatib et al., 1988). A foam regulates itself to keep p_c around p_c^*. When p_c rises above p_c^*, gas mobility increases, water saturation rises, and the foam collapses causing the capillary pressure to decrease. Water saturation is kept contanst by the relationship between water saturation and capillary pressure, i.e. $S_w(p_c^*) \equiv S_w^*$. Water mobility, λ_w, is also a function of S_w and remains constant as well. This allows us to apply Darcy's law to the water phase without taking the lamellae into account in the gas phase (Rossen 1996).

2.4.2. Foam Modeling

Osterloh and Jante (1992) identify two foam-flow regimes based on gas-fractional flow: high- and low- quality. The high-quality (dry) regime is where the steady-state pressure gradient is independent of gas superficial velocity. The pressure gradient decreases with increasing foam quality at constant total superficial velocity (Eftekari and Farajzdeh, 2017). In the low-quality regime, the pressure gradient is independent of the liquid flow rate.

Alvarez et al. (2001) observed that in the low-quality regime, foam is typically shear-thinning. This gives the foam high mobility near the wellbore, improving injectivity. The mobility decreases as the capillary pressure approaches p_c^*. Foam quality, f_g, increases and foam enters the high-quality regime (Rossen and Zhou, 1995). The high-quality regime can be either Newtonian, shear-thickening, or shear-thinning.

Cheng et al. (2000) modeled foams in two simulators - STARS and UTCOMP - both of which fit foam behavior reasonably well for shear-thinning foams in the low-quality regime, but assume Newtonian behavior in the high-quality regime. They observed that shear-thinning in the low-quality regime increases foam injectivity in radial flow. This does not apply to all cases in the high-quality regime. We use the STARS foam model under a set of assumptions:

- Only two phases - water and gas in our case - are present
- The reservoir is initially fully saturated with water
- Foam doesn't alter the $k_{rw}(S_w)$ function
- The gas relative permeability is the only factor altered by foam in gas phase mobility
- Foam is present throughout the reservoir wherever there is surfactant and gas present

Foam reduces gas relative permeability, increasing sweep efficiency. The STARS model simulates this by adding in a foam mobility reduction factor, FM. This represents how the gas mobility is reduced at a certain water saturation relative to a system without foam. The presence of foam is denoted by superscript ^{f}.

$$k_{rg}^f(S_w) = FM(S_w) \times k_{rg}(S_w) \tag{2.9}$$

The foam mobility reduction factor is calculated as

$$FM(S_w) = \frac{1}{1 + fmmob \times F_1F_2F_3F_4F_5F_6} \tag{2.10}$$

F_1, F_2, F_3, F_4, F_5, and F_6 describe the effects of surfactant concentration, dryout, oil saturation, gas velocity, shear-thinning effect in the low-quality regime, and critical capillary number. We can ignore all of these except for the dryout factor, F_2, which shows the effect of water saturation on foam behavior. The reference mobility-reduction factor for gas mobility at maximum foam strength is called $fmmob$ (Cheng et al., 2000). The foam mobility reduction factor simplifies to

$$FM(S_w) = \frac{1}{1 + fmmob \times F_2} \tag{2.11}$$

In the STARS model, foam doesn't completely collapse at residual water saturation. Laboratory studies suggest that foam is weaker at low water saturations than what the STARS model calculates (Khatib et al., 1998; Rossen et al., 2016). The Namdar Zanganeh correction for F_2 is used to adjust for this, as it assumes that foam collapses completely at the well (Namdar Zanganeh et al., 2011). This correction is used in

our model due to the importance of accurately modeling the behavior of foam in the high-quality regime.

$$F_2(S_w) = \overbrace{\left(0.5 + \frac{\arctan(epdry(S_w - fmdry))}{\pi}\right)}^{\text{STARS model}} \underbrace{- \left(0.5 + \frac{\arctan(epdry(S_{wr} - fmdry))}{\pi}\right)}_{\text{Namdar Zanganeh et al. correction}} \tag{2.12}$$

The water saturation where foam weakens is called $fmdry$. The abruptness of foam collapse as a function of water saturation is controlled by $epdry$. It controls the transition from the high-quality regime to the low-quality regime (Kapetas et al., 2017).

2.5. Fractional-Flow

We limit ourselves to local-steady-state processes and ignore capillary-pressure gradients. This implies that the total superficial velocity is constant, so we can define the fractional-flow of water as

$$f_w(S_w) \equiv \frac{Q}{A} = \frac{u_w}{u_t} = \frac{u_w}{u_w + u_g} \tag{2.13}$$

We can rewrite this equation in terms of mobility

$$f_w(S_w) = \frac{\lambda_{rw}(S_w)}{\lambda_{rw}(S_w) + \lambda^f_{rg}(S_w)} \tag{2.14}$$

which simplifies to

$$f_w(S_w) = \left(1 + \frac{\lambda^f_{rg}(S_w)}{\lambda_{rw}(S_w)}\right)^{-1} \tag{2.15}$$

(Zhou and Rossen, 1995).

2.6. Pressure Differences

2.6.1. Single-Phase Flow

For incompressible, steady-state, 1D radial flow, the pressure difference between the wellbore radius and outer radius is found by integrating the simplified Darcy's law, Equation 2.2, from the wellbore to the outer reservoir radius.

$$\int_{P_{r_e}}^{P_{r_w}} dP = -\int_{r_e}^{r_w} \frac{Q_i \mu}{A(r) k k_r} dr = \int_{r_w}^{r_e} \frac{Q}{A(r) k \lambda_r} dr \tag{2.16}$$

In single-phase Newtonian flow, the mobility is constant. $A(r) = 2\pi rh$, so the pressure difference is

$$P_{r_w} - P_{r_e} = \frac{Q}{2\pi rh\lambda_r} ln\left(\frac{r_e}{r_w}\right) \tag{2.17}$$

2.6.2. Multi-Phase Flow

Expand the formula for total superficial velocity, Equation 2.7, with the formula for mobility, Equation 2.13, to get

$$u_t = u_w + u_g^f = -\left(k\lambda_{rw}(S_w)\frac{\partial P_w}{\partial r} + k\lambda_{rg}^f(S_w)\frac{\partial P_g}{\partial r}\right) \tag{2.18}$$

By neglecting the capillary pressures, the pressure gradient in both phases are equal. The total relative mobility is the sum of the phase mobilities, so now the equation can be simplified to

$$u_t = -k\lambda_{rt}(S_w)\frac{dP}{dr} \tag{2.19}$$

Integrating this between the wellbore radius and outer radius yields

$$P_{r_w} - P_{r_e} = \frac{Q_t}{2\pi hk}\int_{r_w}^{r_e} \frac{1}{r\lambda_{rt}(S_w(r))}dr \tag{2.20}$$

2.7. Dimensionless Parameters

2.7.1. Position

Dimensionless position, x_D, is defined as the fraction of the pore volume at a certain radius, r, back to the injection point.

$$x_D(r) \equiv \frac{r^2 - r_w^2}{r_e^2 - r_w^2} \quad \text{where } x_D \in [0,1] \tag{2.21}$$

The outer radius is denoted by r_e, and the well radius is given by r_w.

2.7.2. Time

Dimensionless time, t_D, is defined as pore volumes injected divided by the pore volume of the region of interest.

$$t_D \equiv \int \frac{Q_t}{\pi(r_e^2 - r_w^2)h\phi} \quad \text{where } t_d > 0 \tag{2.22}$$

The height is denoted by h and the porosity is symbolized by ϕ.

2.7.3. Pressure

Changing injectivity is the main feature of interest for injection of non-Newtonian foams. Injectivity profiles are the easiest way to examine and compare foam performance. Dimensionless pressure, P_D, is the inverse of injectivity. It is defined as the pressure it takes to inject the foam, normalized by the pressure it would take to inject water in a fully water-saturated formation (Al Ayesh et al., 2016).

$$P_D = \frac{P_{r_w} - P_{r_e}}{(P_{r_w} - P_{r_e})_{S_w=1}} = \frac{\partial P_g}{\partial P_w} = \frac{\frac{Q_g}{2\pi rkh}\int_{r_w}^{r_e}\frac{1}{r\lambda_{rt}(S_w)}dr}{\frac{Q_w}{2\pi rkh\lambda_{rw}}ln\left(\frac{r_e}{r_w}\right)} \tag{2.23}$$

The equation can be simplified by canceling out like terms, and assuming a water viscosity of 1 cP (10^{-3} Pa·s).

$$P_D = \frac{\int_{r_w}^{r_e}\frac{1}{r\lambda_{rt}(S_w)}dr}{1000 \times ln\left(\frac{r_e}{r_w}\right)} \tag{2.24}$$

Since we can not solve the function with an integral analytically, we take the sum instead. The dimensionless pressure is a summation of the pressure differences across the water bank, P_w (Equation 2.25), and between successive characteristics within the spreading wave, $P_{foam\,bank}$ (Equation 2.26), at a specific time interval.

Calculate the water bank pressure, P_w, ahead of the shock with

$$P_w = \mu_{water} \times \ln\frac{r_e}{r_{shock}} \tag{2.25}$$

Calculate the pressure difference between characteristics behind the shock using

$$P_{foam\,bank} = \sum_{i=1}^{n-1}\left[\left(\frac{1}{\lambda_{rt}}\right)_i + \left(\frac{1}{\lambda_{rt}}\right)_{i+1}\right] \times \frac{1}{2} \times \ln\frac{r_{i+1}}{r_i} \tag{2.26}$$

where n is the number of characteristics including the shock if it's still within the region of interest, i.e. $r_{shock} \leq r_e$. If the foam has experienced breakthrough, the first characteristic pair is calculated using a total relative mobility value at r_{step}, which is interpolated between the nearest exited characteristic, a, and itself, b. Taking their respective positions, r_a and r_b, with their known mobility

$$\lambda_{interpolated} = \lambda_a + (\lambda_b - \lambda_a)\left(\frac{r_{step} - r_a}{r_b - r_a}\right) \tag{2.27}$$

The total dimensionless pressure is the summation of all the characteristic pressures at the time interval divided by the reference injection pressure of water

$$P_{total} = \frac{P_w + \sum P_{foam\,bank}}{\mu_{water} \times \ln\frac{r_e}{r_w}} \tag{2.28}$$

2.8. Mass Balance

For an incompressible system, the conversation of mass states that the net flow equals the accumulation, i..e.

$$\text{mass in} - \text{mass out} + \text{source} = \text{accumulation} \tag{2.29}$$

In terms of radial flow, using ρ for density, ϕ for porosity, and t for time, the mass balance is

$$\underbrace{2\pi h(\rho u r)_r \Delta t}_{\text{flow in}} - \underbrace{2\pi h(\rho u r)_{r+\Delta r} \Delta t}_{\text{flow out}} = \underbrace{2\pi r \Delta r h(\rho\phi)_{t+\Delta t} - 2\pi r h(\phi\rho)_t}_{\text{accumulation}} \tag{2.30}$$

Divide both sides by $2\pi r h \Delta t \Delta r$

$$\frac{1}{r}\frac{(\rho u r)_r - (\rho u r)_{r+\Delta r}}{\Delta r} = \frac{(\rho\phi)_{t+\Delta t} - (\rho\phi)}{\Delta t} \tag{2.31}$$

Take the limit as both Δx and Δr approach 0.

$$\lim_{\Delta r \to 0} \lim_{\Delta x \to 0} \left(\frac{1}{r}\frac{(\rho u r)_r - (\rho u r)_{r+\Delta r}}{\Delta r} = \frac{(\rho\phi)_{t+\Delta t} - (\rho\phi)}{\Delta t} \right) \tag{2.32}$$

This gives the radial continuity equation,

$$-\frac{1}{r}\frac{\partial(\rho u r)}{\partial r} = \frac{\partial(\rho\phi)}{\partial t} \tag{2.33}$$

The continuity equation can be used for multi-phase flow if the phase saturation is taken into account, and u is understood to be for a single-phase i. Using the same assumptions and method as for the single-phase method above, the continuity equation for multi-phase flow becomes

$$-\frac{1}{r}\frac{\partial(u_i r)}{\partial r} = \phi\frac{\partial S_i}{\partial t} \tag{2.34}$$

and when expressed in three dimensions

$$-\nabla \cdot (\rho\vec{u}) = \frac{\partial}{\partial t}(\rho\phi) \tag{2.35}$$

We can rewrite Equation 2.34 by substituting in Equation 2.13.

$$-\frac{1}{r}\frac{\partial(u f_w r)}{\partial r} = \phi\frac{\partial S_w}{\partial t} \tag{2.36}$$

The quantity $(u \cdot r)$ is conserved and constant so we take it out of the derivative. Superficial velocity, $u_t = Q/A$, can be substituted in. Rearranging gives

$$\phi\frac{\partial S_w}{\partial t} + \frac{Q_t}{2\pi r h}\frac{\partial f_w}{\partial r} = 0 \tag{2.37}$$

This equation can be expanded using the derivative of dimensionless position with respect to radius,

$$\frac{dx_D}{dr} = \frac{2r}{r_e^2 - r_w^2} \tag{2.38}$$

and the derivative of dimensionless time with respect to radius,

$$\frac{dt_D}{dr} = \frac{Q_t}{\pi h \phi (r_e^2 - r_w^2)} \tag{2.39}$$

to give

$$\frac{Q_t}{\pi h \phi (r_e^2 - r_w^2)} \phi \frac{\partial S_w}{\partial t} + \frac{2r}{r_e^2 - r_w^2} \frac{Q_t}{2\pi r h} \frac{\partial f_w}{\partial r} = 0 \tag{2.40}$$

Canceling like-terms, the dimensionless mass balance is

$$\frac{\partial f_w}{\partial x_D} + \frac{\partial S_w}{\partial t_D} = 0 \tag{2.41}$$

This is the governing differential equation. When the chain rule is used to expand the fractional water partial derivative, Equation 2.41 becomes

$$\frac{df_w}{dS_w} \frac{\partial S_w}{\partial x_D} + \frac{\partial S_w}{\partial t_D} = 0 \tag{2.42}$$

The boundary conditions are

$$S_1(x_D, 0) = S_{1I}, \quad x_D \geq 0 \tag{2.43a}$$

$$S_1(0, t_D) = S_{1J}, \quad t_D \geq 0 \tag{2.43b}$$

where I is the initial condition and J is the injection condition (Lake, 1989).

2.9. Buckley-Leverett Theory

For Newtonian flow, water fractional-flow is a function of position and time, and is only dependent on water saturation. At a constant saturation, this partial differential equation becomes an ordinary differential equation. The solution is found via integration with initial conditions in a process called the 'Method of Characteristics' (MOC). This calculates the rate at which an injected water bank moves through a porous medium. We use the normal assumptions, except we will allow for non-Newtonian phases. Fractional-flow theory is based on the following assumptions:

- flow is linear
- flow is horizontal
- the reservoir is homogeneous, with uniform thickness, with inner radius r_w and an open outer boundary at r_e, where $r_e >> r_w$

- foams are injected at r_w at fixed total volumetric rate Q
- only two phases are present
- both phases are both incompressible and immiscible
- gravity and disperssive processes (e.g. capillary pressure effects, fingering, and dispersion) are negligible
- dissipative effects near the shock are ignored
- local steady-state mobilities are dependent only on local saturations and are instantaneously attained
- no chemical or biological reactions

(Pope, 1980; Lake, 1989). For the foam model, we also assume that surfactant adsorption is zero (Rossen et al., 2011).

Saturation depends only on dimensionless position and time. With these assumptions we can write the total derivative for a constant saturation,

$$dS_w(x_D, t_D) = \frac{\partial S_w}{\partial x_D} dx_D + \frac{\partial S_w}{\partial t_D} dt_D = 0 \tag{2.44}$$

Rearrange Equation 2.44 for the specific velocity, v_{S_w}, and rewrite the relationship using $df_w/dS_w = dx_d/dt_D$.

$$v_{S_w} \equiv \left(\frac{dx_D}{dt_D}\right)_{S_w} = -\frac{\left(\frac{\partial S_w}{\partial t_D}\right)_{x_D}}{\left(\frac{\partial S_w}{\partial x_D}\right)_{t_D}} \tag{2.45}$$

The partial derivatives can be eliminated by substituting in the governing differential equation, Equation 2.41, giving

$$v_{S_w} = \left(\frac{dx_D}{dt_D}\right)_{S_w} = \left(\frac{df_w}{dS_w}\right)_{S_w} \tag{2.46}$$

This means that the change in saturation over time along a path equals 0. The specific velocity of a constant saturation S_w is equal to the derivative of the fractional-flow curve at that saturation. We can plot the solutions on the $x_D - t_D$ plane where the saturation varies from S_{w_I} to S_{w_J}.

For all solutions between J and I, they advance with velocity equal to df_w/dS_w. The initial condition, I, is set to $S_w = 1$. The gas injection condition, J, is set at $S_w = S_{wr}$ (Lake, 1989). Applying fractional-flow theory, the Buckley-Leverett theory allows us to solve for the velocity of the foam bank throughout the reservoir over time. An example is shown in Figures 2.1 and 2.2.

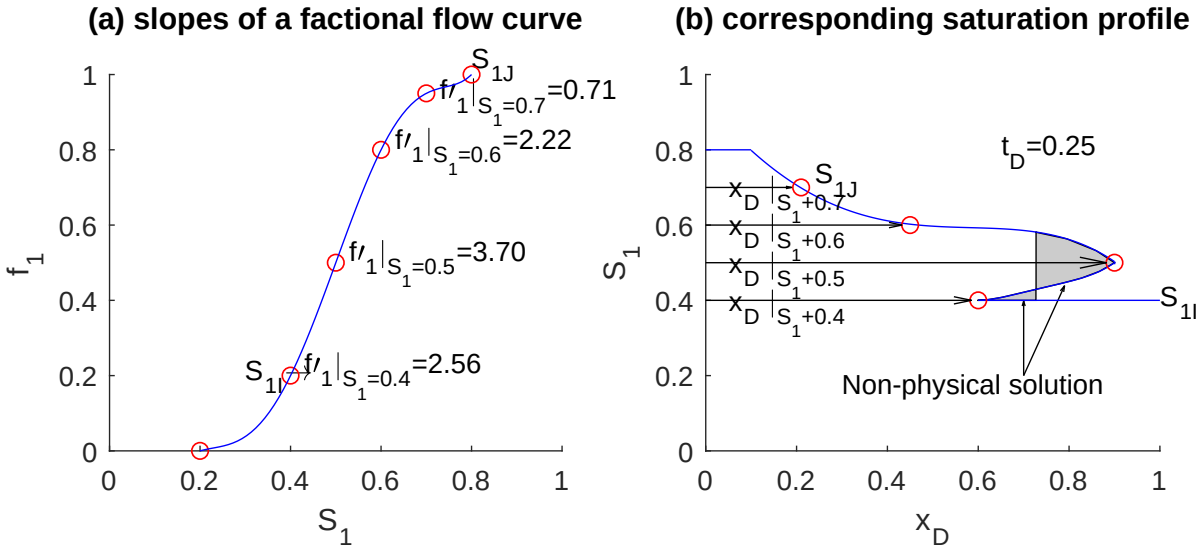

Figure 2.1: Slope of a fractional-flow curve (figure adapted from Lake, 1989, p. 134)

2.9.1. Shock Formation

Fractional-flow curves are plotted as a function of saturation. The curve has an S-shape due to the nonlinear exponent in the Corey relative permeabilities, because flow becomes extremely inefficient for phases just above residual saturation.

The shock occurs due to the abrupt transition between the low- and high- quality regimes. The material upstream is moving faster than downstream, causing accumulation and a shock. The fractional-flow curve's saturation profile will have non-monotonically increasing water saturations and thus have three solutions for the same x_D and t_D. This is shown in the saturation probile in Figure 2.1b corresponding to the fractional-flow curve in Figure 2.1a. While this is mathematically sound, it isn't physically possible. To remove this physical inconsistency, the region from initial condition I to the tangency of the fractional-flow curve is represented with a jump in saturation called the shock. In Figure 2.1a, the gray area would be removed with a jump.

A shock is an abrupt change in conditions that propagates downstream. This is most easily described with a time-distance diagram since it shows both profiles and histories of lines of constant saturation. Figure 2.2 is the associated time-distance diagram for our example in Figure 2.1.

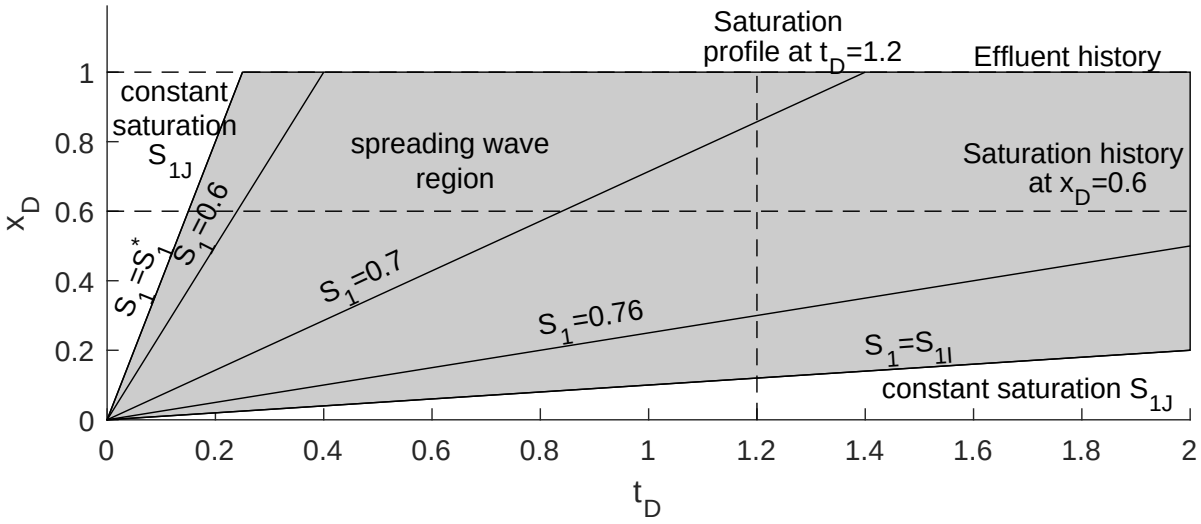

Figure 2.2: Time-distance diagram for displacement of Figure 2.1(figure adapted from Lake, 1989, p. 138)

2.10. SAG Theory

2.10.1. Newtonian SAG

Foam changes gas mobility by altering its relative permeability and effective viscosity. Sequential injection is the preferred method of foam injection, where alternating slugs of surfactant and gas are injected. This is commonly referred to as a SAG process (Kibodeaux and Rossen, 1997). During gas injection the entire foam bank is in the high-quality regime and the fractional-water content is low (Rossen et al., 1999). Boeije and Rossen (2014) found that for single-slug SAG processes, the pressure difference across the foam bank during radial flow is nearly constant in time during gas injection. The main advantage of SAG is its low mobility at the foam front and good injectivity.

2.10.2. Non-Newtonian SAG

The changing velocities of the shock and characteristics make the non-Newtonian case more complicated. Rossen et al. (2011) extended fractional-flow theory for foam injection in non-Newtonian cases. We discuss two cases - shear-thinning and shear-thickening - separately below. For details on the calculation of the shock, refer to Section 3.4.

Shear-thinning For shear-thinning foams, the shock slows down as it progresses outwards (Cheng et al., 2000). The $fmdry$ value decreases outwards from the wellbore, changing the slope of the fractional-flow curve. The tangency point of the shock also shifts, changing the slopes of all the characteristics as well. The velocity of the shock will decrease, while the characteristics increase in speed.

The mobility at the foam front decreases as the foam bank moves away from the injection well. As x_D increases and foam mobility decreases, the fractional-flow curve moves towards smaller S_w values for a given f_w (Rossen et al., 2011). The shock curves downward over time while the characteristics bend upwards, leading to collisions. The characteristics do not collide with each other within the spreading wave, but do collide with the shock.

Shear-thickening Similarly, but in the opposite direction as for the shear-thinning foam, the characteristic's velocities change. The $fmdry$ value increases outwards from the wellbore, shifting the tangency point on the fractional-flow curve. The shock's velocity increases while the characteristics slow down. Instead of characters colliding with the shock, new characteristics branch off as the shock advances. Foam mobility increases as the foam bank moves away from the injection well. While x_D increases, the fractional-flow curves moves to the right, i.e. larger S_w values for their respective f_w values.

Literature Review The STARS foam model doesn't account for the changing S_w* values for non-Newtonian foams (Cheng et al., 2000; Rossen et al., 2011). It assumes Newtonian behavior in the high-quality regime, which is an oversimplification (Rossen et al., 2011). Leeftink et al. (2015) determined that the Method of Characteristics analytical approach was more accurate at solving for mobility and injectivity of non-Newtonain foams than finite-difference models such as STARS. Previous extensions of the theory include non-Newtonian behavior in the high-quality regime only after the shock has left the near-wellbore region and do not solve for injectivity (Rossen et al., 2011). Further research included the shock at the near-wellbore region and calculated injectivity for a shear-thinning foam. However, the model was inefficient when a high accuracy at the near-wellbore region was needed (ter Haar, 2017).

3

Methodology

The model is explained in Section 3.1. The solver method's precision is given in Section 3.2, before explaining the calculations required to calculate the $fmdry$, shock, and characteristics in Sections 3.3, 3.4, and 3.5 respectively. The input parameters are listed in Section 3.6. The model is validated in Section 3.7 and it's most recent updates are outlined in Section 3.8.

3.1. The Model

Our model considers only radial flow during the first gas injection, where the reservoir is initially fully-saturated with surfactant solution. To represent a non-Newtonian foam, x_D is divided into discrete increments, each with its own $fmdry$ value. A simplified version of the model is shown in Figure 3.1. Each layer's properties are constant so the fractional-flow curve and saturation profile can be calculated. When a characteristic enters the next layer, it's $f_w(S_w)$ function is fixed (Rossen et al., 2011). The S_w value is recalculated in the new layer from f_w and the $f_w(S_w)$ function for that layer.

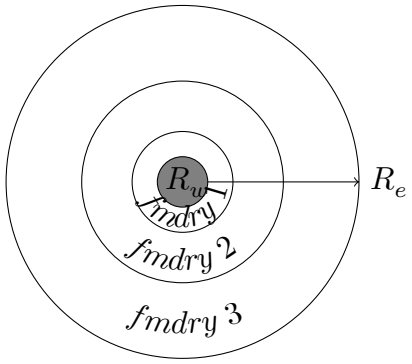

Figure 3.1: Simplified model (top view)

The characteristics will be straight lines within each layer because the properties are constant. However, since the reservoir is divided into many small layers, the characteristics will still curve as they move across many layers. This is conceptually similar to the Riemann sum in calculus for a integral.

For shear-thinning foams, the $fmdry$ value will decrease outwards from the outer wellbore radius. As a result, the shock slows down in velocity while the characteristics will increase in velocity, eventually colliding with the shock. In the calculations shown below assume that the shock does not alter in velocity as a result of the collision, and that the characteristic does not propagate further. This assumption is later proved to be incorrect in Chapter 4, and will be fixed in future versions of the model.

The shear-thickening case is the inverse. The $fmdry$ values increase as dimensionless position increases. The shock wave speeds up as it propagates, while the characteristics slow down. Additional characteristics split off from the shock. There are not included in the calculations shown below, but will be included in future work. Preliminary results show that the addition of the branching-off shocks does not significantly effect the injectivity calculation.

For both cases, the characteristics do not collide with each other because the slope is monotonically increasing with f_w for all x_D. The leading characteristics within the spreading wave will always be ahead of those behind them, unless they collide with the shock and disappear.

3.2. Solver Method

Matlab R2017b was used to model the two-phase flow throughout the reservoir using the Method of Characteristics. Double-precision is used throughout. The exact algorithms are attached as code in Appendix A.

3.3. Calculating $fmdry$ Values

The foam model fits power-law behavior at fixed foam quality in the high-quality regime if the $fmdry$ value varies with r. To calculate the change in $fmdry$ value, we derive an equation based on the power-law relation

$$\Delta p = -c(n){u_t}^n \begin{cases} 0 < n < 1, & \text{shear-thinning} \\ 1, & \text{Newtonian} \\ 1 < n < \infty, & \text{shear-thickening} \end{cases} \tag{3.1}$$

where $c(n)$ is a constant. We assume injection at fixed f_w in the high-quality regime, where $S_w = fmdry$. Use Darcy's Law in Equation 2.3,

$$\vec{u_w} = -\frac{kk_{rw}(S_w = fmdry)}{\mu_w}\nabla p \tag{3.2}$$

to solve for u_t by substituting in Equation 2.13.

$$\vec{u_t} = -\frac{kk_{rw}(fmdry)}{\mu_w f_w}\nabla p \tag{3.3}$$

Solve Equation 3.1 and Equation 3.3 for Δp and set them equal to each other.

$$\Delta p = -\frac{u_t f_w \mu_w}{kk_{rw}(fmdry)} = -c_1(n)u_t^n \tag{3.4}$$

Isolate k_{rw} and set equal to Equation 2.5 at $S_w = fmdry$.

$$k_{rw}(fmdry) = \frac{f_w\mu_w}{kc_1(n)}u_t^{1-n} = k_{rw_e}\left(\frac{fmdry - S_{wr}}{1 - S_{wr} - S_{gr}}\right)^{n_w} \tag{3.5}$$

Solving for $fmdry$,

$$fmdry = \underbrace{S_{wr} + (1 - S_{wr} - S_{gr})\left(\frac{f_w\mu_w}{kc_1(n)k_{rw_e}}\right)^{\frac{1}{n_w}}}_{=c_2} u_t^{\frac{1-n}{n_w}} \tag{3.6}$$

Use the relationship for radial flow, $u_t \propto c_3 r^{-1}$, and combine constants c_2 and c_3 into one constant called c_4, giving us the equation for $fmdry$.

$$fmdry = S_{wr} + c_4 r^{\frac{n-1}{n_w}} \tag{3.7}$$

To solve for c_4, use the $fmdry$ value at the reservoir radius, r_e. Equation 3.7 can be used to calculated the $fmdry$ values at all positions. For each layer, the radius at the beginning is used to calculate the new $fmdry$ value.

3.4. Calculating the Shock

The shock's water saturation at the wellbore is calculated from the input parameters given in Section 3.6 and the $fmdry$ calculated by Equation 3.7. The shock is located at the fractional-flow curve's tangency point. The tangency line passes through $(S_w, f_w) = (1, 1)$, so the tangency point can be solved for using

$$\frac{df_w}{dS_w} = \frac{1 - f_w}{1 - S_w} \tag{3.8}$$

To calculate df_w/dS_w, we use finite-difference with a step-size of 10^{-8}.

As the shock progresses into the next layer, its S_w is recalculated using the $fmdry$ value at the start of the layer from Equation 3.7 and its conserved value of f_w carried over from the previous layer. The slope of the shock is calculated using Equation 3.8.

3.5. Calculating the Characteristics

Use finite-forward difference to solve for the characteristics. Use a step size of 10^{-8} for S_w. The fractional-water flow is determined using Equation 2.13. The slope of the characteristic is then equal to

$$\frac{df_w}{dS_w} = \frac{f_{w_2} - f_{w_1}}{S_{w_2} - S_{w_1}} \tag{3.9}$$

In our calculations, if a characteristic collides with the shock, the characteristic disappears without affecting the velocity of the shock.

As the characteristics progress into the next layer, their S_w values are calculated using the new layer's $fmdry$ value and their conserved f_w rate carried over from the previous layer. The slope of the characteristics are calculated using Equation 3.9.

3.6. Input Parameters

3.6.1. Well Parameters and Layer Thickness

The wellbore radius, r_w, is 0.1 m. The external reservoir radius, r_e, is a 100 m open outer boundary. The total superficial velocity changes 1,000 times between the wellbore and the outer radius.

Since the most interesting and crucial behavior takes place at the near-wellbore region, it's computationally more efficient to have more-closely-spaced layers with changing $fmdry$ values at the beginning. To achieve this, we use logarithmically-spaced layers based on radial position between the wellbore radius and reservoir radius. The proportional velocity change for every layer is thus constant. For 1,000 layers, the velocity changes 0.7% from one layer to the next.

3.6.2. Number of Characteristics

200 characteristics are analyzed. They are determined by using linearly-spaced S_w values between the shock and the residual water saturation for the first layer.

3.6.3. Foam and Petrophysical Properties

For the shear-thinning model, our power-law constant, n, equals 0.33 (Osterloh and Jante, 1992). We look at two shear-thickening models with n values of 1.34 and 1.67 (Alvarez et al., 2001).

The other input parameters are consistent throughout all three trials. They are listed

in Table 3.1.

Table 3.1: Input Parameters

Water			Gas				Foam		
μ_w	$0.001^{\underline{a}}$	$Pa \cdot s$	μ_g	$0.00002^{\underline{a}}$	$Pa \cdot s$		$fmdry_{r_e}$	$0.271^{\underline{b}}$	–
S_{wr}	$0.25^{\underline{b}}$	–	S_{gr}	$0.2^{\underline{b}}$	–		$fmmob$	$47,700^{\underline{b}}$	–
k_{rw_e}	$0.2^{\underline{b}}$	–	k_{rg_e}	$0.59^{\underline{b}}$	–		$epdry$	$400^{\underline{b}}$	–
n_w	$4.2^{\underline{b}}$	–	n_g	$1.3^{\underline{b}}$	–				

Superscript a refers to parameters taken from Al Ayesh et al. (2016) and superscript b refers to parameters taken from Kapetas et al. (2015).

3.7. Model Validation

A simplified version of this model was validated in ter Haar (2017). The accuracy was tested against Al Ayesh et al. (2016) and Bos (2017) for the injectivity of a Newtonian foam and against Al Ayesh et al. (2016) for the mobility of a Newtonian foam. The results were not significantly different, allowing us to use this model with confidence. The current version of this model was tested against Ponners (2017) and the results were not significantly different.

3.8. Model Comparison

The model is significantly improved compared to the one presented in ter Haar (2017). Crucially, the resolution at the near wellbore region is improved by introducing radial, instead of linear, spacing for the layer thickness. The model is extended to include shear-thickening foams. Better insights are gained through the addition of several new graphs, with optional add-ins for better analysis.

Numerical methods are also improved, so that a wider range of input parameters are accepted, the calculations for the $S_w(f_w)$ function are more accurate, and all observable rounding and overflow errors are removed. The program can now also handle more characteristics and layers, while reducing computation time.

4

Results and Discussion

The shear-thinning example is discussed in Section 4.1. The shear-thickening examples are in Section 4.2 for $n = 1.34$ and Section 4.3 for $n = 1.67$.

4.1. Shear-thinning (n=0.33) Foam

4.1.1. Results

The input parameters are listed in Table 3.1. We use $n = 0.33$ (Osterloh and Jante, 1992). The value of $fmdry$ decreases from 0.356 at the wellbore to 0.271 at the reservoir radius. The resulting dimensionless time-distance plot is shown below in Figure 4.1. Fifty-three of the characteristics collide with the shock before its breakthrough point at $t_D = 0.740$. The shock slows down from an initial velocity of 1.43 to 1.35 at breakthrough.

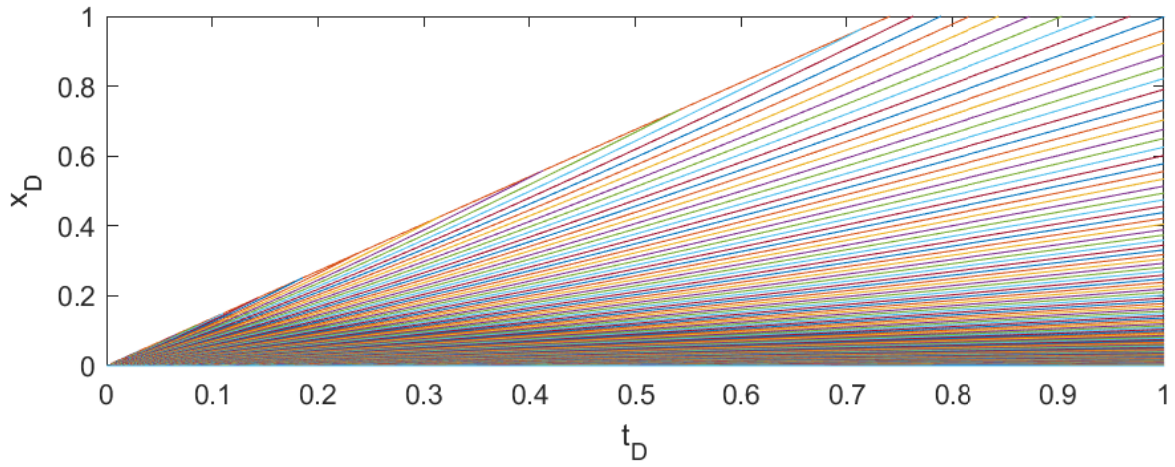

Figure 4.1: Dimensionless time-distance diagram

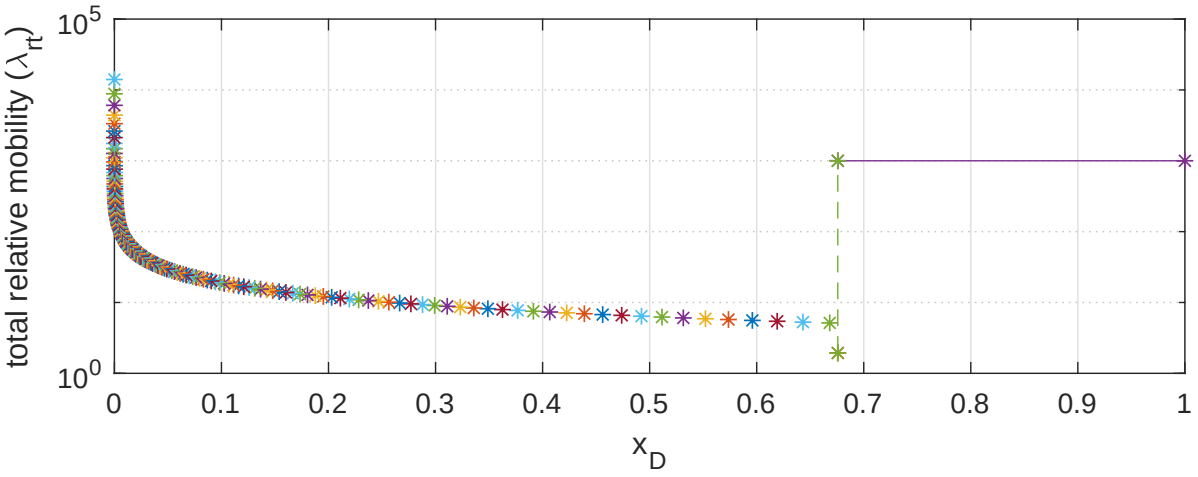

Figure 4.2: Total relative mobility at time slice of $t_D = 0.5$

Figure 4.2 shows how the total relative mobility increases as the water saturation decreases throughout the spreading wave at a fixed time. The shock has the lowest relative mobility. There is a jump in mobility where the shock and the water are. The water is at a fixed total relative mobility ahead of the spreading wave.

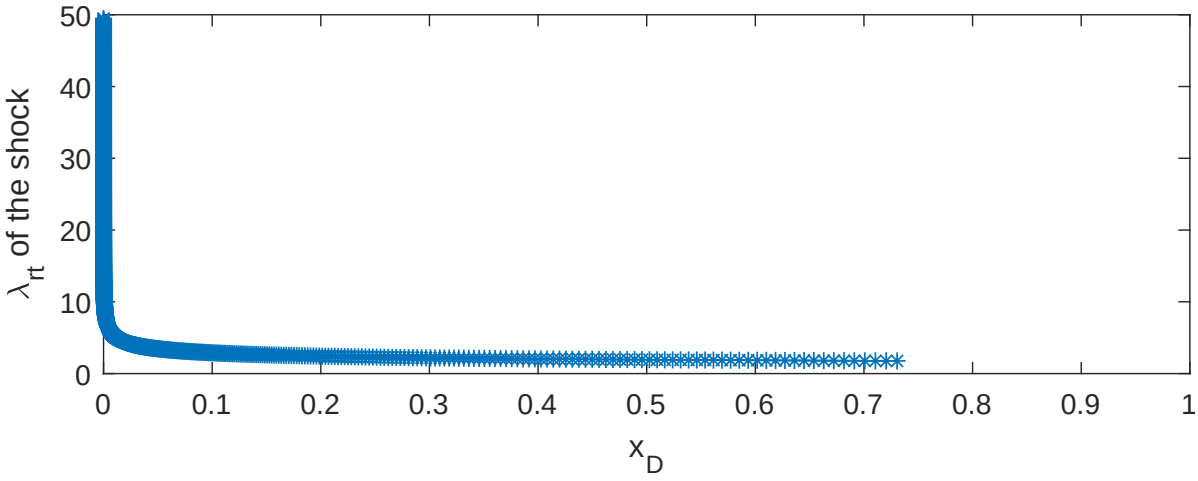

Figure 4.3: Total relative mobility of the shock versus position

The total relative mobility of the shock drops from 49.5 at r_w to 1.77 at r_e.

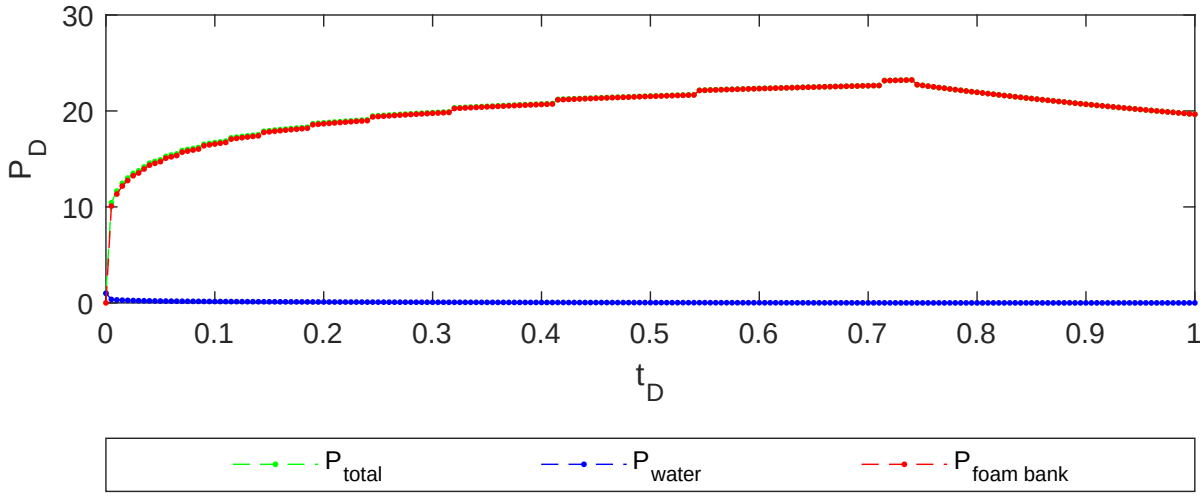

Figure 4.4: Dimensionless pressure over time

The dimensionless pressure increases quickly as the shock progresses. The max dimensionless pressure is at breakthrough, where $P_D = 23.2$. The dimensionless pressure drops afterwards, but more slowly than before to $P_D = 20.1$ at $t_D = 1$. Prior to SAG injection, the dimensionless pressure was one.

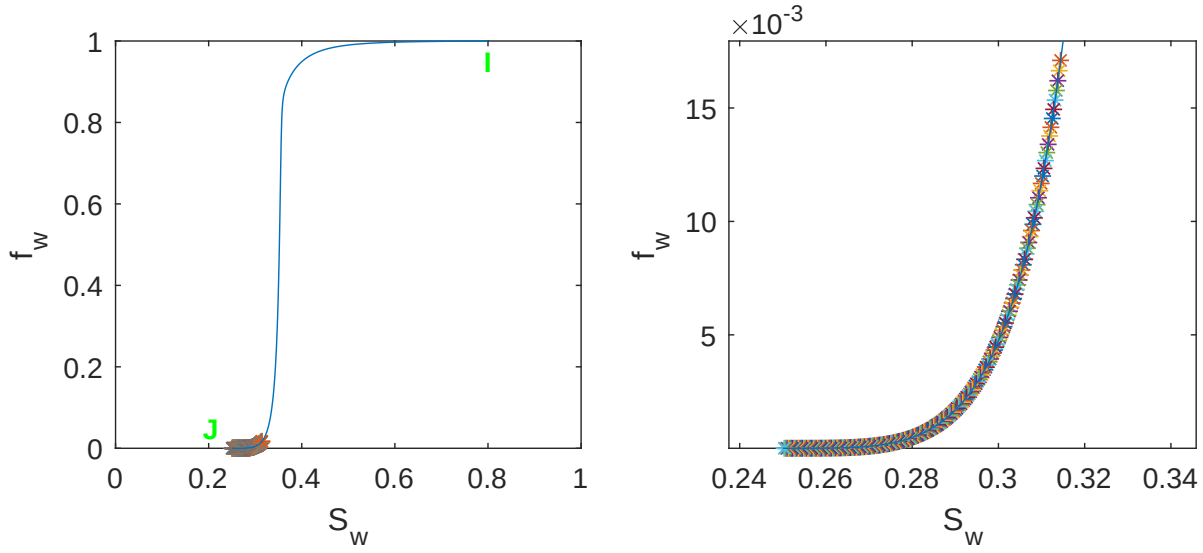

Figure 4.5: Fractional-flow curve at r_w, $fmdry = 0.356$

The dimensionless pressure for a Newtonian case with the $fmdry$ value at r_w would be 3.29.

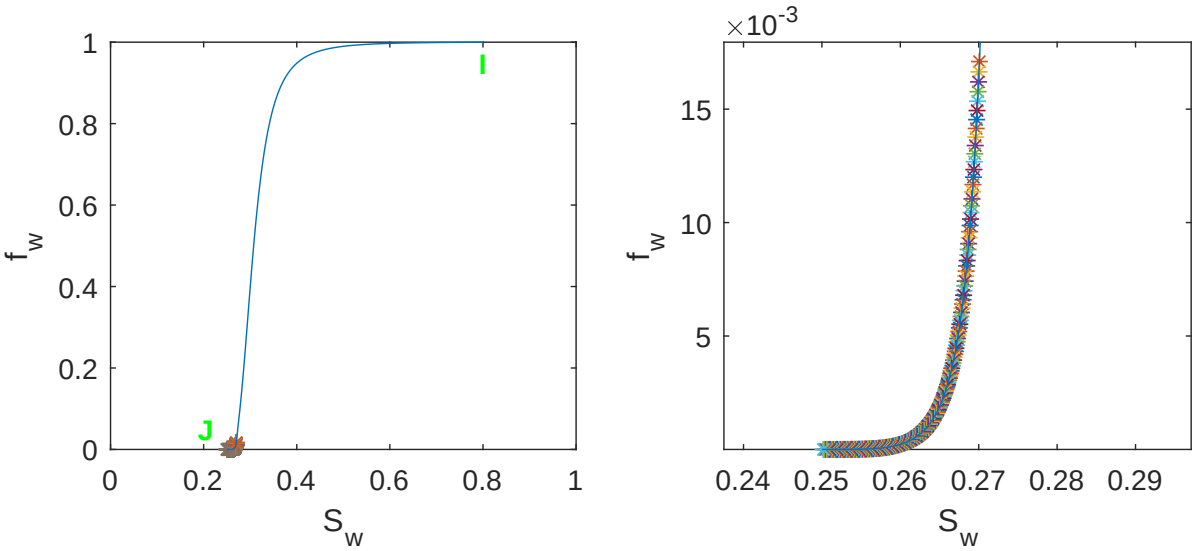

Figure 4.6: Fractional-flow curve for r_e, $fmdry = 0.271$

The dimensionless pressure for a Newtonian case with the $fmdry$ value at r_e would be 29.4.

4.1.2. Discussion

These results are mostly what is expected for a strongly shear-thinning foam. The top quarter of the leading characteristics collide with the shock in Figure 4.1. The shock slows down over time, and has a relatively low mobility compared to the rest of the characteristics. This is easily observed in Figure 4.2 where the λ_{rt} for the shock is much lower than the leading characteristics. This is most likely an error. We assumed that when characteristics collide with the shock, that the characteristics disappear. We should have done the opposite; the colliding characteristic should become the new shock and the old shock disappears. This is confirmed by Figure 4.7. It shows that our shock calculated by $S_w(f_w)$ is higher than the actual tangency point, and that this difference grows larger as the foam propagates. If the colliding characteristic were to be the new shock, the gap between the S_w of the tangency point and our S_w calculated from (f_w) would be much smaller.

At injection, the total relative mobility of the shock is at its highest, $\lambda_{rt} = 49.5$. It falls quickly, before remaining nearly stable around $\lambda_{rt} = 1.77$ at a t_D of 0.01 in Figure 4.3. The injectivity increases with time, but not as much as with a Newtonian process with the $fmdry$ value at r_e. The mobility ratio becomes more favorable as the foam front propagates, going from a $\lambda_f : \lambda_w$ ratio of 0.05 at r_e to a ratio of $1.8 : 1000$ at breakthrough.

The dimensionless pressure plot, Figure 4.4, shows that after the initial jump in P_D due to the SAG injection, the dimensionless pressure slowly increases until breakthrough. The pressure difference across the foam bank is not constant over time. This is unlike what is seen in the Newtonian case (Boeije and Rossen, 2014). Mobility at the shock front decreases as the distance it travels increases meaning that mobility control improves over time. The injectivity decreases until breakthrough, and then improves slightly.

The fractional-flow curves in Figures 4.5 and 4.6 also behave in the expected manner. The value of $fmdry$ decreases, meaning that for the same f_w value, a characteristic's new S_w value is smaller. Graphically, this is seen as the slope of the fractional-flow curve changing and the bottom part moving towards the left in Figure 4.8.

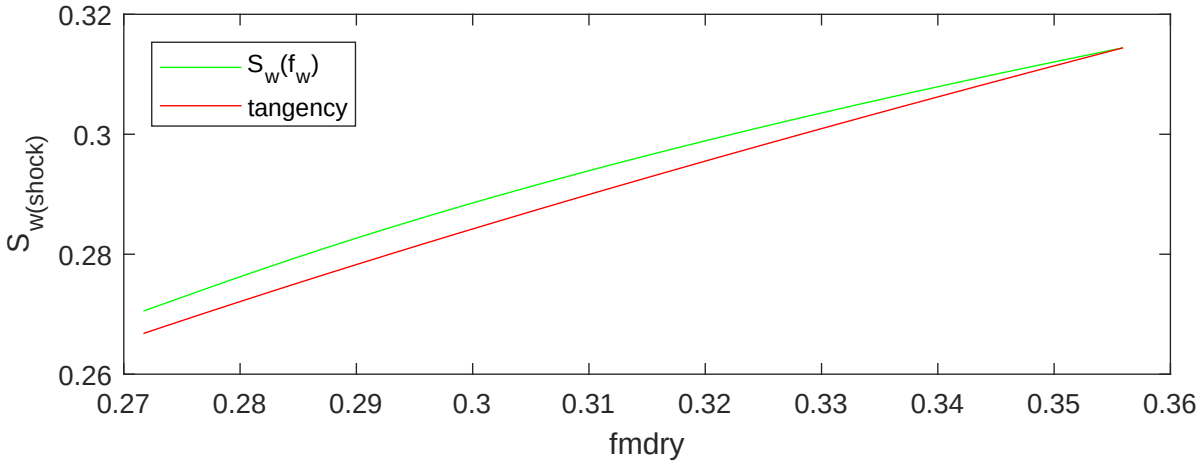

Figure 4.7: Difference between the shock calculated by the $f_w(S_w)$ function and via its tangency point

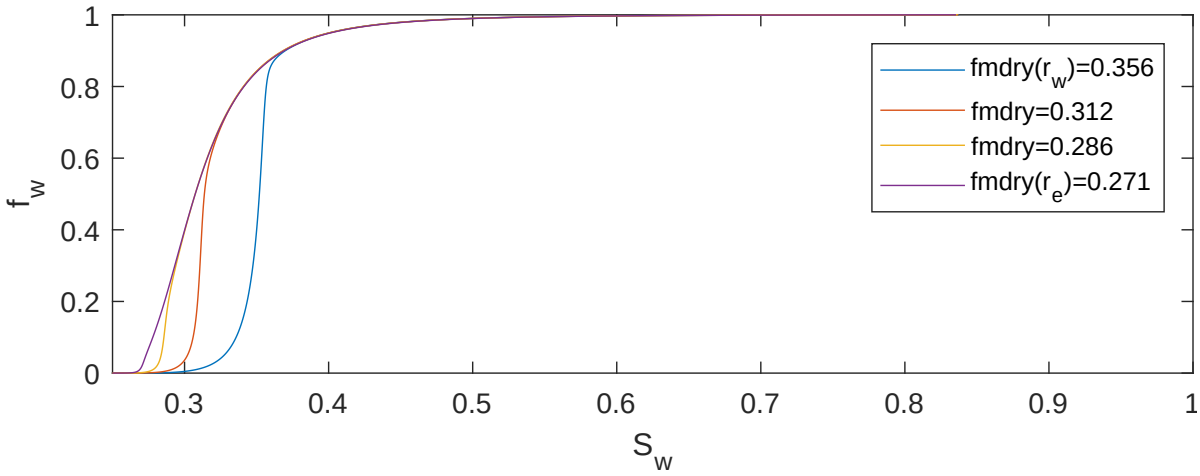

Figure 4.8: Fractional-flow curves for four $fmdry$ values

4.2. Shear-thickening (n=1.34) Foam

4.2.1. Results

The input parameters are listed in Table 3.1. We use $n = 1.34$ (Alvarez et al., 2001). The value of $fmdry$ increases from 0.259 at the wellbore to 0.271 at the reservoir radius. The resulting dimensionless time-distance plot is shown below in Figure 4.9. None of the characteristics collide with the shock before breakthrough at $t_D = 0.727$. The shock increases in velocity from an initial velocity of 1.35 to 1.36 at breakthrough.

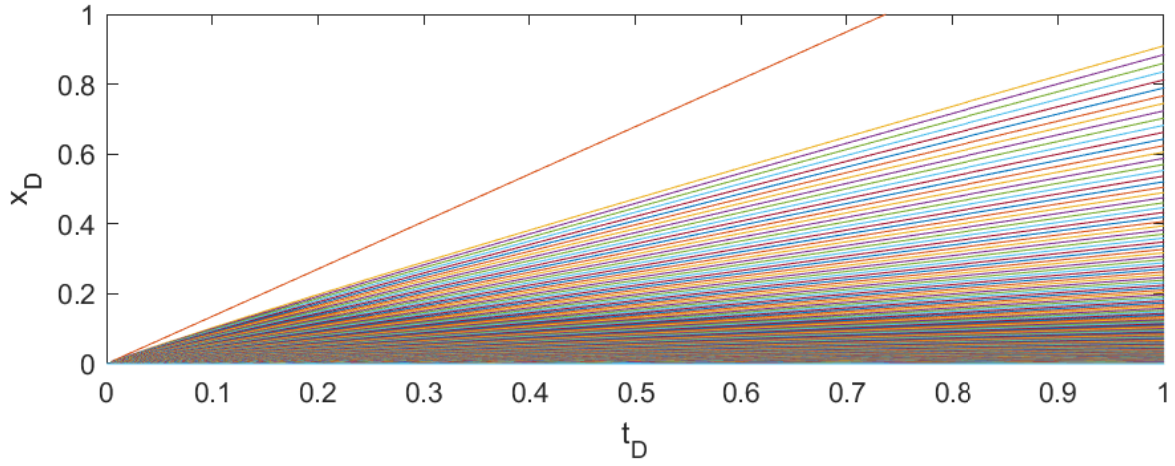

Figure 4.9: Dimensionless time-distance diagram

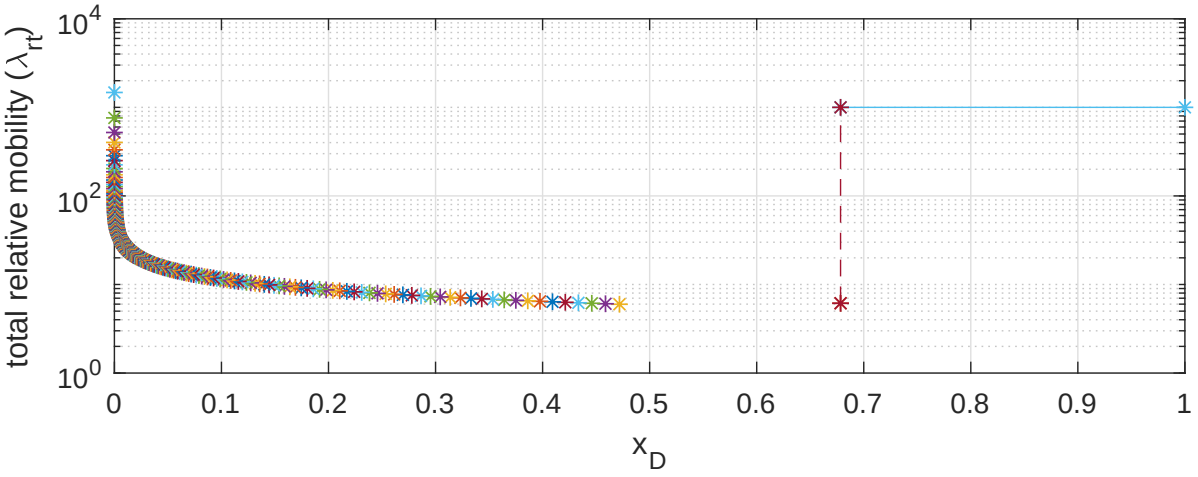

Figure 4.10: Total relative mobility at time slice of $t_D = 0.5$

Figure 4.10 shows how the total relative mobility increases throughout the spreading wave as the water saturation decreases. The shock has the lowest relative mobility. In front of the shock is water with a constant total relative mobility.

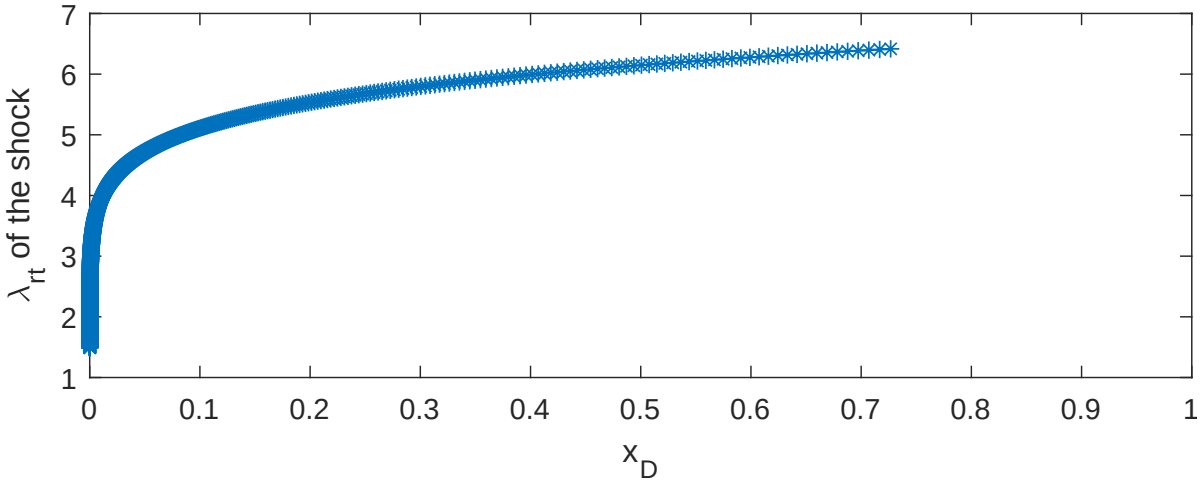

Figure 4.11: Total relative mobility of the shock versus position

The total relative mobility of the shock increases from 1.49 at r_w to 6.42 at r_e.

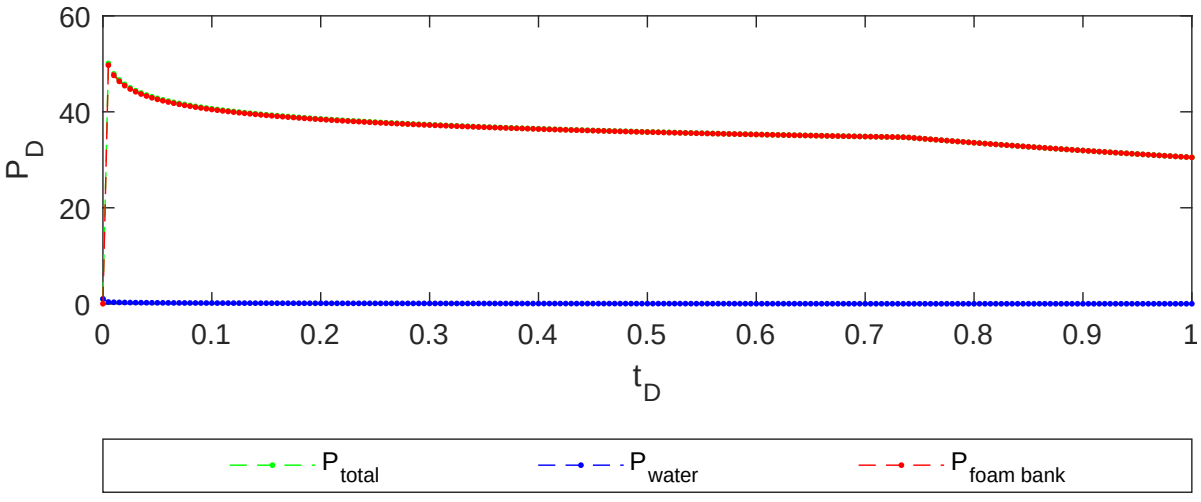

Figure 4.12: Dimensionless pressure over time

The dimensionless pressure decreases slowly as the shock progresses. The maximum dimensionless pressure is at the beginning, with a P_D of 50.1. The dimensionless pressure drops slowly to 34.7 at breakthrough and then further to 30.5 at $t_D = 1$.

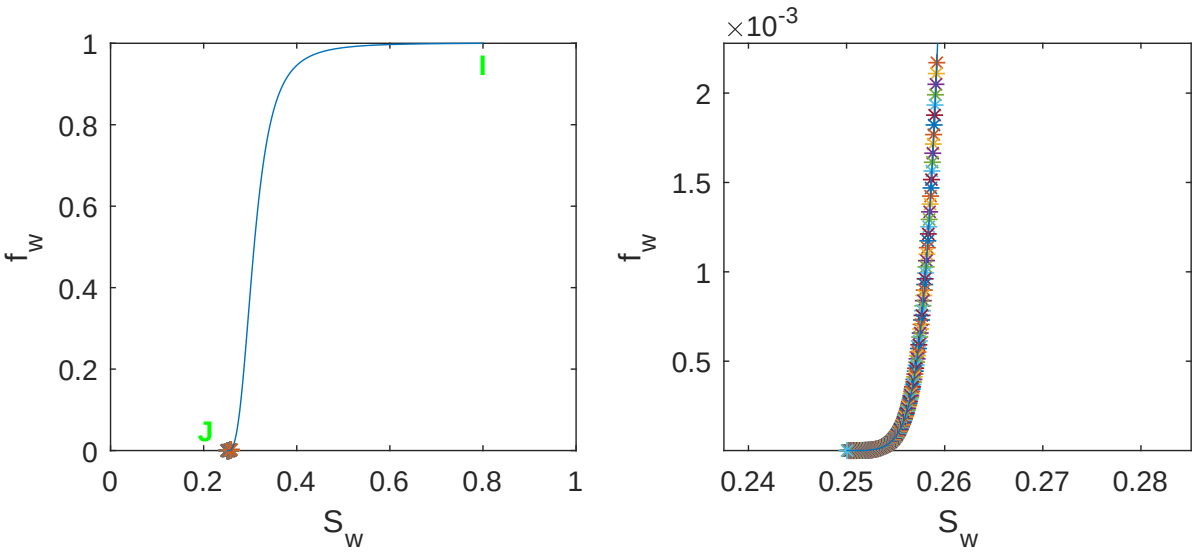

Figure 4.13: Fractional-flow curve at r_w, $fmdry = 0.259$

The dimensionless pressure for a Newtonian case with the $fmdry$ value at r_w would be 90.6.

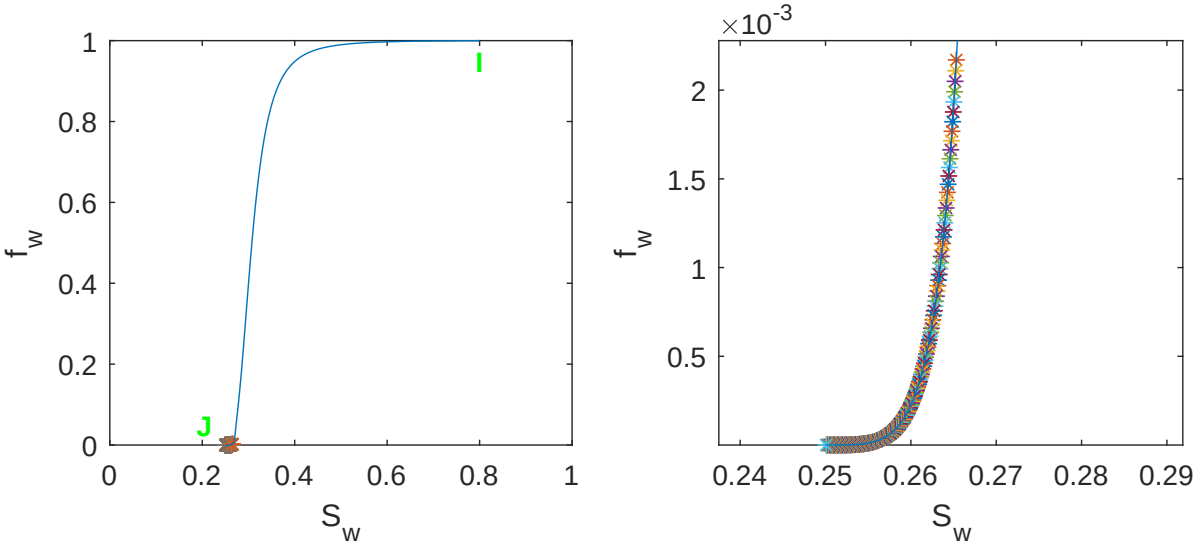

Figure 4.14: Fractional-flow curve for r_e, $fmdry = 0.271$

The dimensionless pressure for a Newtonian case with the $fmdry$ value at r_e would be 29.5.

4.2.2. Discussion

Everything behaves as expected for a moderately shear-thickening foam. Figure 4.9 shows the characteristics decreasing in velocity while the shock speeds up. At larger x_D positions, there is a big gap between the shock and the leading characteristics. For shear-thickening foams, new characteristics branch off. This is not visualized, so a gap exists in dimensionless position between the shock and the leading characteristics in Figure 4.10. Future versions of this model would benefit from adding in extra characteristics to help smooth out the results. Adding in branching off characteristics from the shock would improve the 'resolution' of the total relative mobility throughout the spreading wave. However, since this has only a negligible impact on the injectivity calculations, it's a low-priority issue to fix.

Unlike with the shear-thinning example, the difference between our calculated $S_w(f_w)$ for the shock and the S_w calculated by the tangency point is nearly negligible. Figure 4.15 shows that the difference increases as the foam propagates, but the error is still relatively small. Still, a more accurate method for calculating the shock would improve the reliability of the results and decrease the gap between the shock and the leading characteristics.

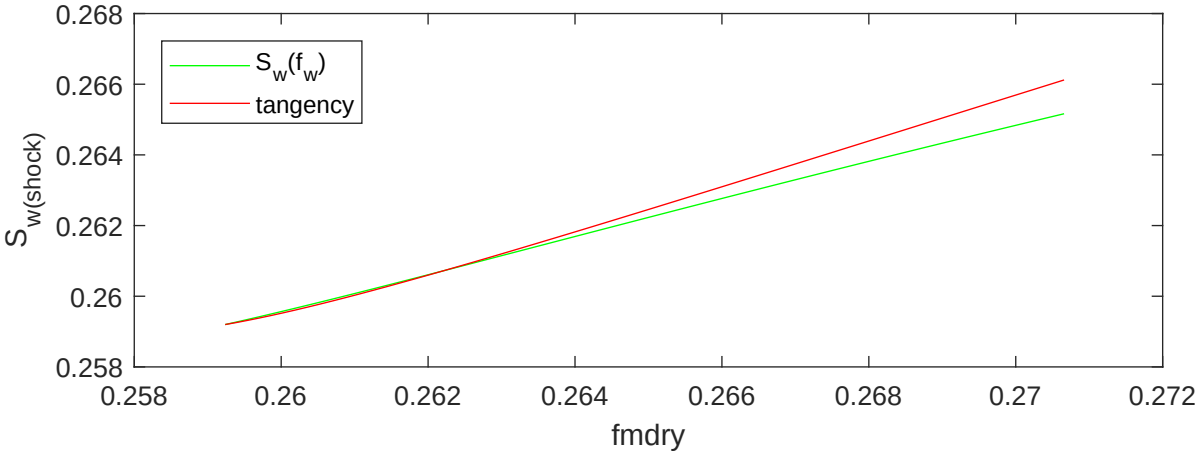

Figure 4.15: Difference between the shock calculated by the $f_w(S_w)$ function and via its tangency point

The total relative mobility of the shock increases quickly at first in Figure 4.11, then steadies around $\lambda_{rt} = 6.42$. The mobility control of the foam front decreases as it progresses through the reservoir and the mobility ratio becomes less favorable. Figure 4.12 shows the result. The injectivity improves slowly as the shock progresses, as well as after breakthrough. The injectivity decreases with time, but is still less favorable than a Newtonian process with the $fmdry$ value at r_e.

The fractional-flow curves in Figures 4.13 and 4.14 also behave in the expected manner. The $fmdry$ value increases, meaning that for the same f_w value, a characteristic's new S_w value is larger. Graphically, this is seen as the slope of the curve changing and the bottom of the curve moving towards the right in Figure 4.16.

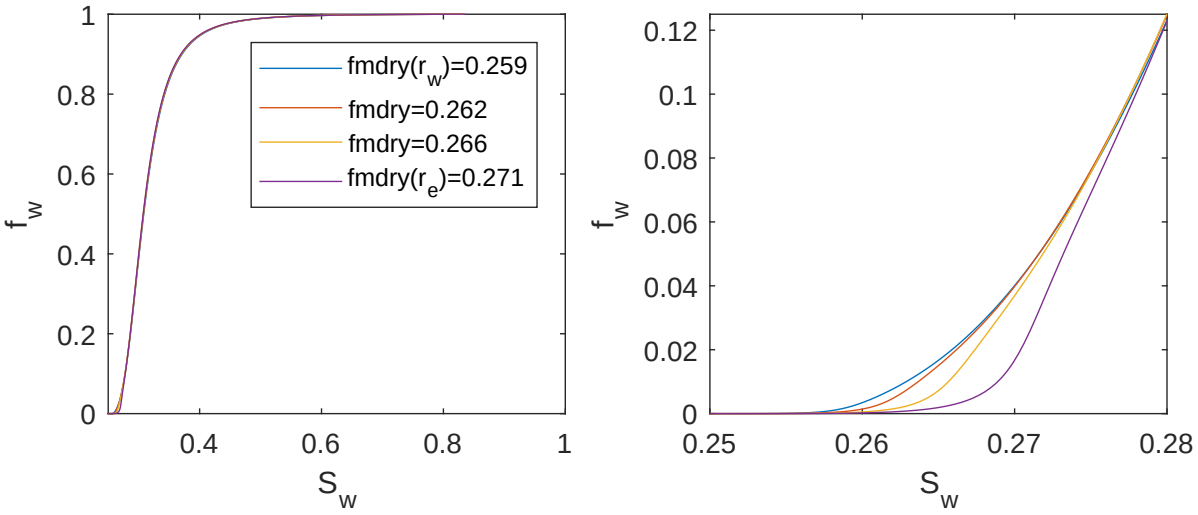

Figure 4.16: Fractional-flow curves for four $fmdry$ values

4.3. Shear-thickening (n=1.67) Foam

4.3.1. Results

The input parameters are listed in Table 3.1. We use $n = 1.67$ (Alvarez et al., 2001). The value of $fmdry$ increases from 0.254 at the wellbore to 0.271 at the reservoir radius. The resulting dimensionless time-distance plot is shown below in Figure 4.17. Interestingly, 18 characteristics collide with the shock before foam breakthrough at $t_D = 0.745$. The shock increases in velocity from 1.35 to 1.36 at breakthrough.

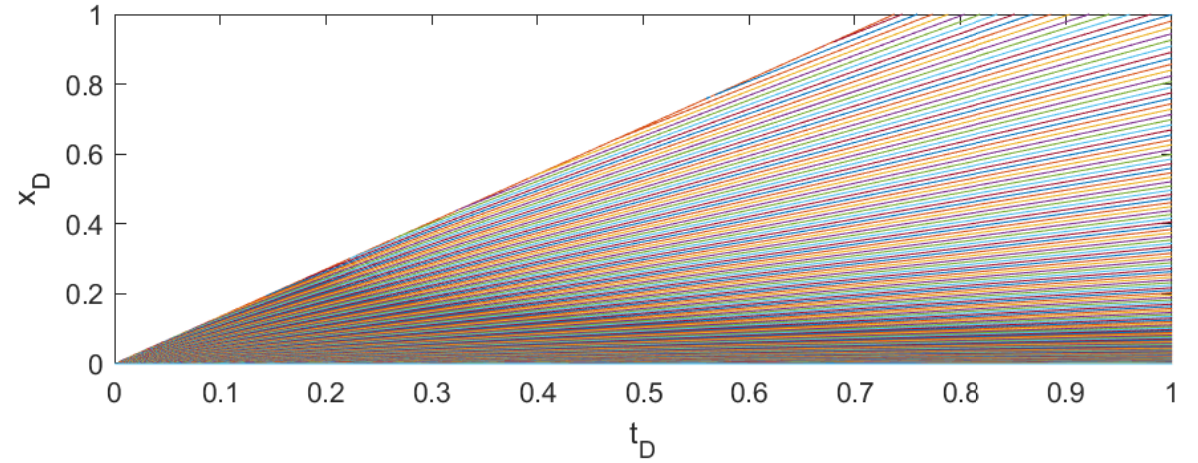

Figure 4.17: Dimensionless time-distance diagram

Figure 4.18 shows how the total relative mobility increases as the saturation decreases throughout the spreading wave. The shock has the lowest relative mobility. The water front has a constant total relative mobility.

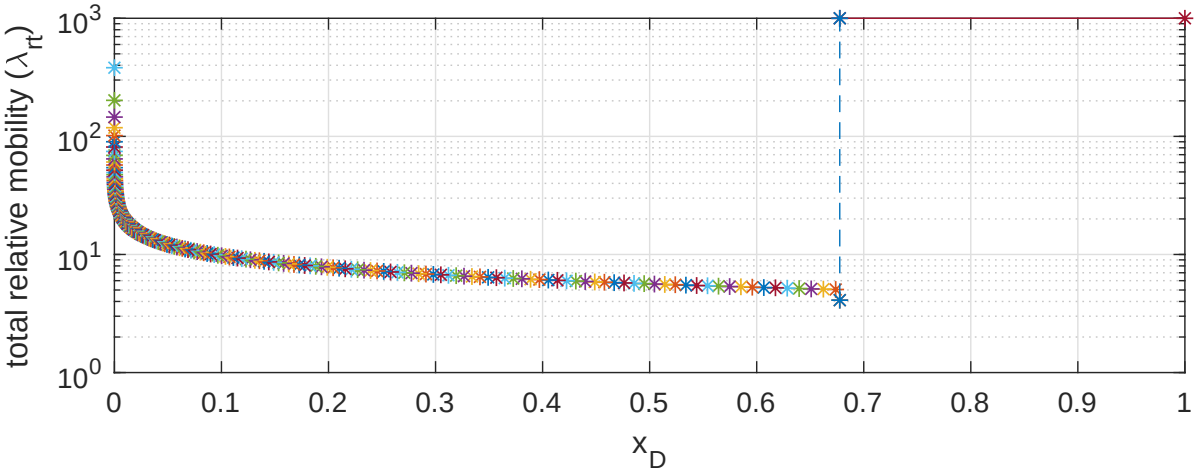

Figure 4.18: Total relative mobility at time slice of $t_D = 0.5$

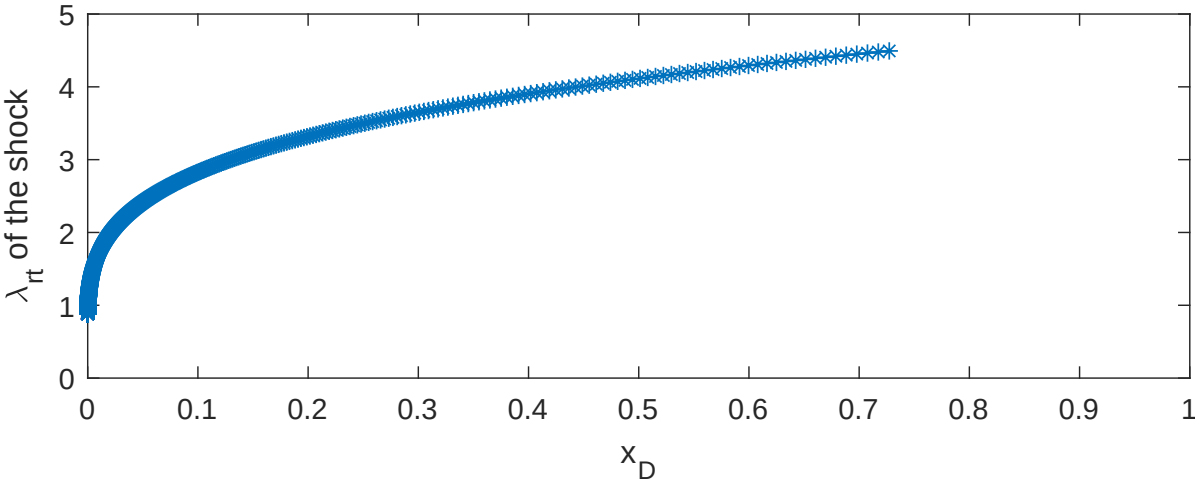

Figure 4.19: Mobility of the shock versus position

The total relative mobility of the shock increases from 0.890 at r_w to 4.49 at r_e.

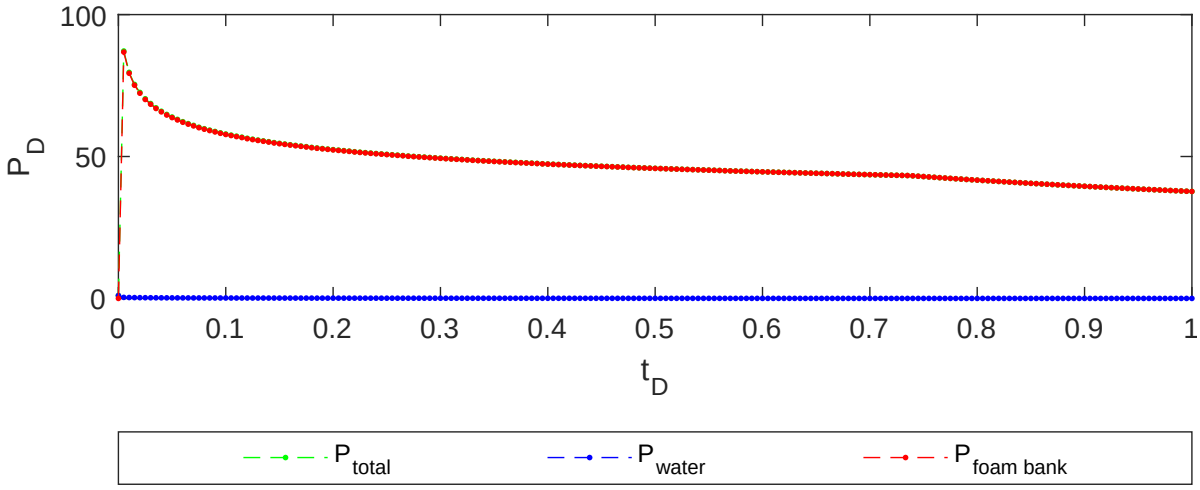

Figure 4.20: Dimensionless pressure over time

The dimensionless pressure decreases slowly as the shock progresses, with the biggest decrease happening at the very beginning. The max dimensionless pressure is at the beginning, where $P_D = 87.2$. The dimensionless pressure slowly to 43.3 at breakthrough and then further to 37.7 at $t_D = 1$.

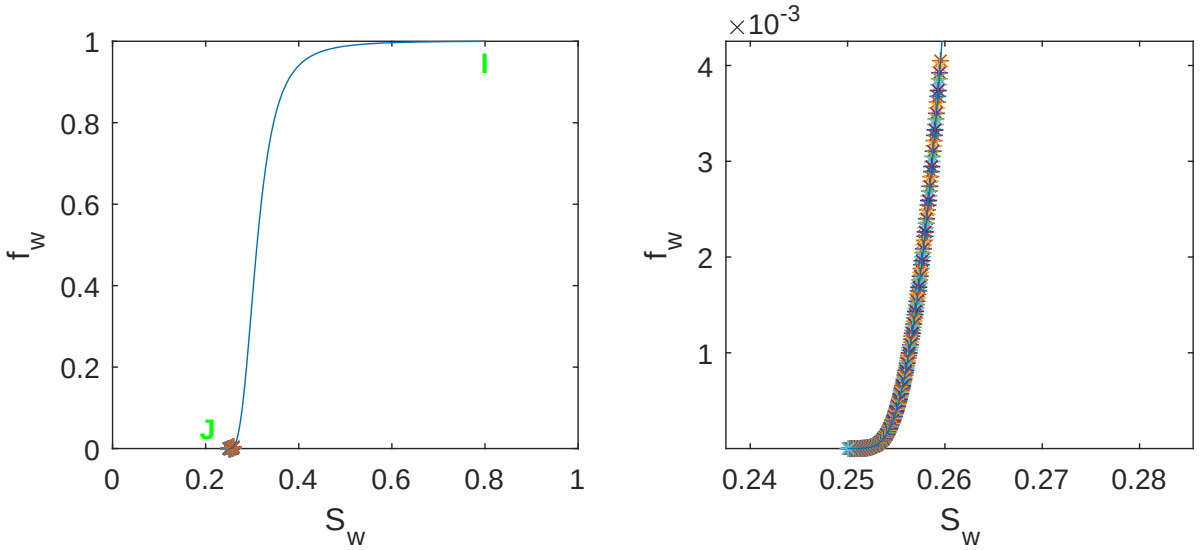

Figure 4.21: Fractional-flow curve at r_w, $fmdry = 0.254$

The dimensionless pressure for a Newtonian case with the $fmdry$ value at r_w would be 206.

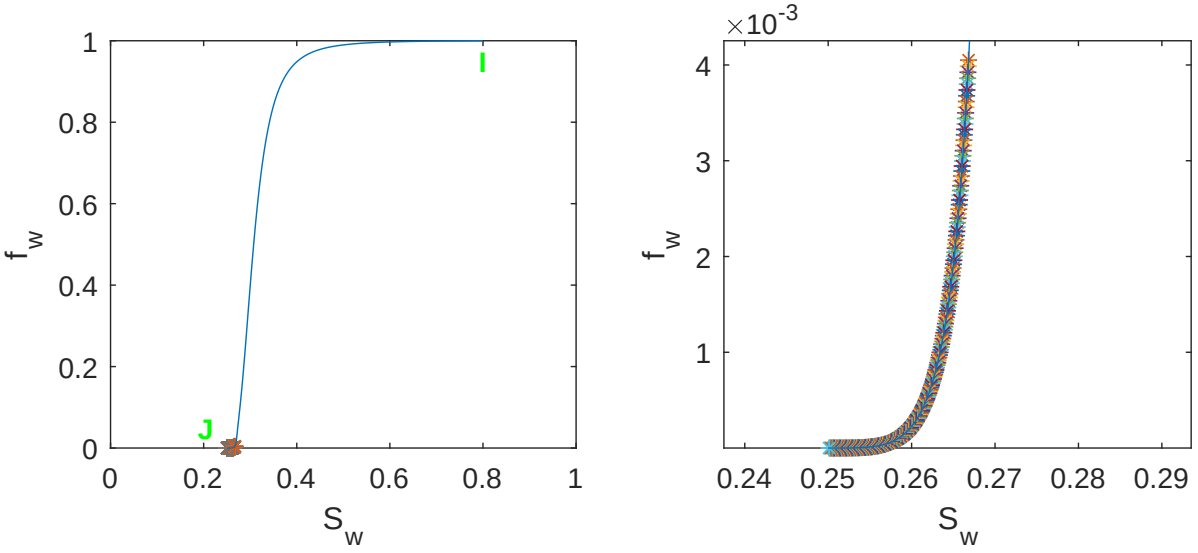

Figure 4.22: Fractional-flow curve for r_e, $fmdry = 0.271$

The dimensionless pressure for a Newtonian case with the $fmdry$ value at r_e would be 29.6.

4.3.2. Discussion

The behavior is unexpected for a shear-thickening foam. Figure 4.17 shows the characteristics colliding with the shock. This is shear-thinning, not shear-thickening, behavior. This explains why the total relative mobility profile for the spreading wave resembles more that of a shear-thinning foam (see Figure 4.2) than that of a shear-thickening foam (Figure 4.10). However, Figure 4.19 shows the total relative mobility of the shock increasing over time, as expected. The mobility control is decreasing as the foam front progresses, and the mobility ratio is becoming less favorable, which is typical of a shear-thickening foam. The dimensionless pressure profile in Figure 4.20 is also typical of a shear-thickening foam; the injectivity improves as the shock progresses, with the majority of the improvement occurring in the beginning. The injectivity decreases with time, but is still less favorable than a Newtonian process with the $fmdry$ value at r_e.

Upon closer investigation, it appears that the S_w values calculated from its f_w value of the shock are higher than the actual tangency point. This is shown in Figure 4.23. This means that the slopes from J to I are no longer monotonically increasing, which causes collisions.

Like we noted with the shear-thinning example, our current method for treating the shock during collisions is an oversimplification. Our current method of solving for new S_w values is by carrying over the f_w to the next layer and recalculating the S_w from it, instead of recalculating the shock via the point of tangency. When a characteristic collides with the shock, we eliminate the characteristic at the point of collision without changing the shock's velocity. It would be more accurate to eliminate the shock at the

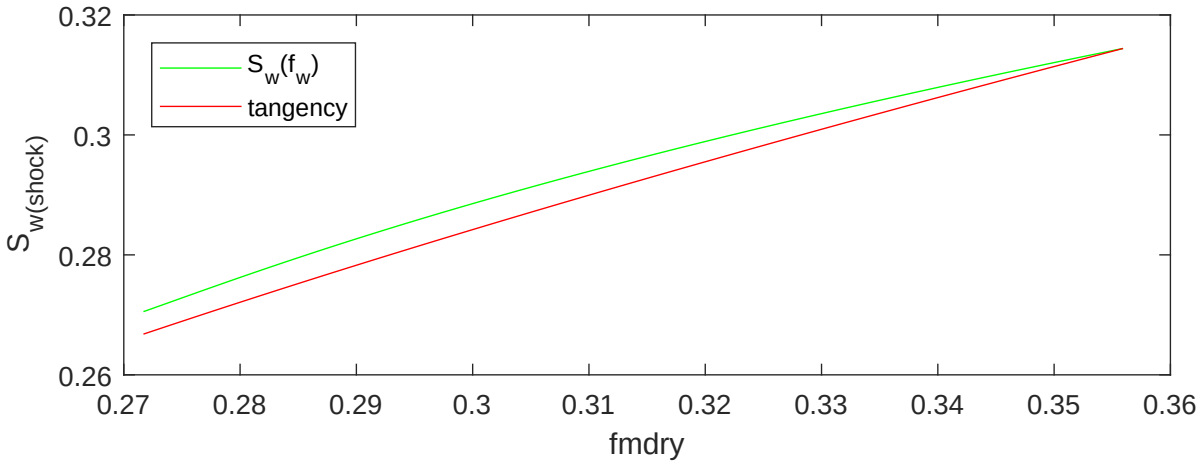

Figure 4.23: Difference between the shock calculated by $f_w(S_w)$ and via its tangency point

point of collision, and then let the characteristics carry on as the 'new' shock. In this way, the velocity of the shock would be impacted by the collisions. The shock and shape of the spreading wave has a large influence on the dimensionless pressure of the foam, so we expect to see the injectivity change as a result.

The fractional-flow curves in Figures 4.21 and 4.22 also appear to behave normally. The $fmdry$ value is decreasing, and bottom of the curve shifts to the right as x_D increases. However, this is misleading. In Figure 4.24 we take a closer look at the fractional-flow curves for four $fmdry$ values. These $fmdry$ values correspond to the inital injection, the minimum $fmdry$ value for the tangency point, slightly afterwards, and at the external reservoir radius. The circles represent the shock saturation calculated by the $f_w(S_w)$ function. As expected, the circles move to the right for S_w. The triangles represent the shock saturation calculated by the tangency point. At the initial $fmdry$ value, this point is the same for both methods. We expect to see the S_w increase, i.e. the triangle moves to the right as $fmdry$ increases. However, for the second $fmdry$ value, the S_w decreases and is smaller than the initial S_w of the shock. Instead of the bottom of the fractional-flow curves moving smoothly to the right, the curves shift and cross each other.

There are several plausible explanations for this, and the actual reason is most likely a combination of these factors. The reasons include our extreme extrapolation of parameters, a shift from the high- to the low-quality regime, and the Namdar Zanganeh et al. correction.

For the input parameters, we take a $fmdry$ value at the reservoir radius and extrapolate it to the wellbore. The velocity changes 1,000 times. This is a significantly larger velocity change that what is observed in labs. It's not possible to extend over such a wide range for an extreme n value. The shifts in $fmdry$ values become so large it stops affecting SAG performance.

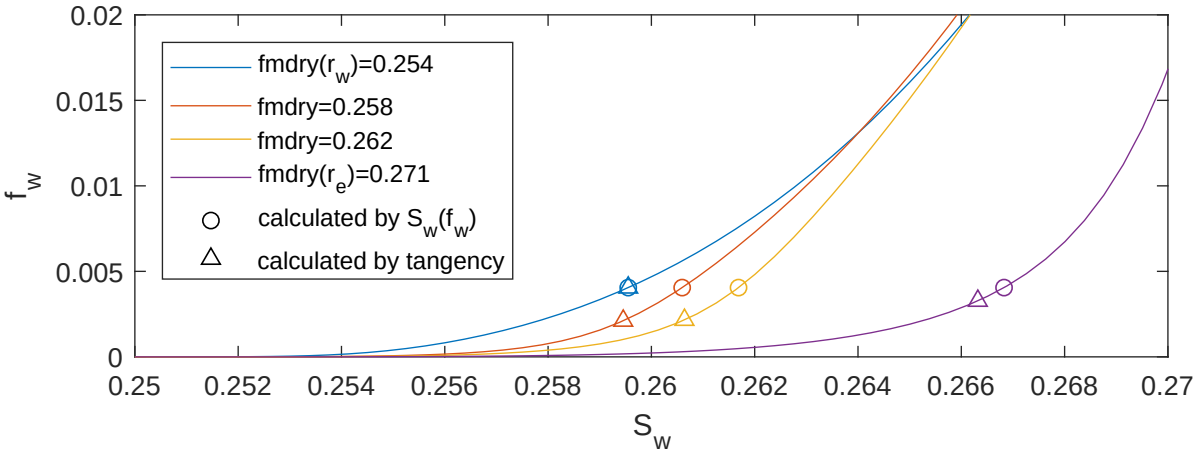

Figure 4.24: Fractional-flow curves for four $fmdry$ values

Additionally, we assume that we are in the high-quality regime. It's possible that we are either at the transition point, or cross over entirely, into the low-quality regime. This happens when S_w^* is close to S_{wr}. This could explain the shifting of the fractional-flow curves that we see in Figure 4.24. Shear-thickening behavior can't happen in the low-quality regime (Cheng et al., 2000). To check for this in future models, we recommend including the shear-thinning factor for the low-quality regime, F_5.

The Namdar Zanganeh et al. (2011) correction has a stronger effect at lower water saturations. This 'tilts' the fractional-flow curve, and contributes to their crossing as well.

5

Conclusions and Recommendations

5.1. Conclusions

It is possible to extend fractional-flow theory for non-Newtonian SAG injection to include the shock with a high-resolution near the wellbore region. This model has numerous improvements compared to the one presented in ter Haar (2017), including an extension for the shear-thickening regime, better resolution near the wellbore, logarithmic spacing of the layers, eliminated noticeable rounding and overflow errors, a wider range of input parameters are accepted, more accurate calculations for the $S_w(f_w)$ function, increased speed, and better analysis through additional graphs.

For shear-thinning foam, the shock slows down while the characteristics speed up. This leads to collisions. The mobility ratio becomes more favorable as the foam front propagates. The injectivity decreases until breakthrough, then improves slightly.

For both shear-thickening foams the mobility control at the foam front worsens. As the foam propagates, injectivity improves, even before breakthrough. The extremely shear-thickening foam showed shear-thinning behavior. Several reasons for this were suggested: the extreme extrapolation of parameters over too wide of a range, a transition between the high-quality and low-quality regime, and the Namdar Zanganeh et al. correction. Our combination of input parameters potentially placed us at the border region between the high-quality and low-quality regime, while the foam is assumed to be in the high-quality regime. The parameters are extrapolated over a wide range of velocities – a 1,000 times increase, while lab experiments use a relatively narrow range of velocities. In fact, its not possible to extend parameters over a wide range. Extreme shifts in $fmdry$ values to large or small values cease to affect SAG performance. The Namdar Zanganeh correction exacerbated this problem by further 'tilting' the fractional flow curve due to its stronger effect at low water saturations.

5.2. Recommendations

The assumptions made about the interaction between the shock and colliding characteristics are oversimplified. We propose a new way of considering the shock, where the colliding characteristic becomes the new shock and the old shock disappears. This way the shock's velocity takes the collisions into account. This should reduce the difference between the shock calculated by the tangency point and the shock calculated by the fractional-flow function. It will be interesting to see if this gap closes completely, or if it remains due to a lag.

The shear-thinning factor for the low-quality regime, $F5$, should be included in the FM calculations. This can either be done to extend the model to include the low-quality regime or as a 'flag' to warn users that the foam has left the high-quality regime.

Characteristics should be added in for shear-thickening foams once the difference between the S_w of the shock and the leading characteristics is significantly large. This would improve the detail on the mobility graphs, but will have little effect on the injectivity calculations.

Works Cited

Al Ayesh, A. H., Salazar, R., Farajzadeh, R., Vincent-Bonnieu, S., & Rossen, W. R. (2017, October 1). Foam Diversion in Heterogeneous Reservoirs: Effect of Permeability and Injection Method. Society of Petroleum Engineers. doi:10.2118/179650-PA

Alvarez, J. M., Rivas, H. J., & Rossen, W. R. (2001, September 1). Unified Model for Steady-State Foam Behavior at High and Low Foam Qualities. Society of Petroleum Engineers. doi:10.2118/74141-PA

Boeije, C. S., & Rossen, W. (2014, February 1). Gas-Injection Rate Needed for SAG Foam Processes To Overcome Gravity Override. Society of Petroleum Engineers. doi:10.2118/166244-PA

Bos, M. (2017). *Numerical Simulation of Radial Non-Newtonian Foam Flow In A Reservoir* (Unpublished Bachelors thesis). Delft University of Technology, Delft, the Netherlands.

Buckley, S. E., & Leverett, M. C. (1942, December 1). Mechanism of Fluid Displacement in Sands. Society of Petroleum Engineers. doi:10.2118/942107-G

Cheng, L., Reme, A. B., Shan, D., Coombe, D. A., & Rossen, W. R. (2000, January 1). Simulating Foam Processes at High and Low Foam Qualities. Society of Petroleum Engineers. doi:10.2118/59287-MS

Eftekhari, A. A., & Farajzadeh, R. (2017). Effect of Foam on Liquid Phase Mobility in Porous Media. Scientific Reports, 7, 43870. doi:10.1038/srep43870

Kapetas, L., Vincent Bonnieu, S., Farajzadeh, R., Eftekhari, A. A., Shafian, S. R. M., Bahrim, R. Z. K., & Rossen, W. R. (2017). Effect of permeability on foam-model parameters: An integrated approach from core-flood experiments through to foam diversion calculations. Colloids and Surfaces A: Physicochemical and Engineering Aspects, 530, 172–180. doi:10.1016/j.colsurfa.2017.06.060

Khatib, Z. I., Hirasaki, G. J., & Falls, A. H. (1988, August 1). Effects of Capillary Pressure on Coalescence and Phase Mobilities in Foams Flowing Through Porous Media. Society of Petroleum Engineers. doi:10.2118/15442-PA

Kibodeaux, K. R., & Rossen, W. R. (1997, January 1). Coreflood Study of Surfactant-Alternating-Gas Foam Processes: Implications for Field Design. Society of Petroleum

Engineers. doi:10.2118/38318-MS

Lake, L. W. (1989). *Enhanced oil recovery.* Englewood Cliffs, N.J: Prentice Hall.

Leeftink, T. N., Latooij, C. A., & Rossen, W. R. (2015). Injectivity errors in simulation of foam EOR. Journal of Petroleum Science and Engineering, 126, 26–34. doi:10.1016/j.petrol.2014.11.026

Namdar Zanganeh, M., Kam, S. I., LaForce, T., & Rossen, W. R. (2011). The Method of Characteristics Applied to Oil Displacement by Foam. SPE Journal, 16(01), 8–23. doi:10.2118/121580-PA

Osterloh, W. T., & Jante, M. J., Jr. (1992). Effects of Gas and Liquid Velocity on Steady-State Foam Flow at High Temperature. Society of Petroleum Engineers. doi:10.2118/24179-MS

Ponners, C. (2017). *SAG Foam EOR Buckley-Leverett Fractional Flow Theory for Non-Newtonian Foam Rheologies* (Unpublished Bachelors thesis). Delft University of Technology, Delft, the Netherlands.

Pope, G. A. (1980). The Application of Fractional Flow Theory to Enhanced Oil Recovery. *Society of Petroleum Engineers Journal*, 20(03), 191–205. doi:10.2118/7660-PA

Rossen, W.R. and Zhou, Z.H. (1995). "Modeling Foam Mobility at the Limiting Capillary Pressure," *SPE Advanced Technolgy.* 3, 146-152.

Rossen, W. R. (1996). Foams in Enhanced Oil Recovery. In R. K. Prud'homme and S. Khan (Eds), *Foams: Theory, Measurements and Applications* (pp. 413–464). New York: Marcel Dekker, Inc.

Rossen, W. R., Zeilinger, S. C., Shi, J. X., & Lim, M. T. (1999, September 1). Simplified Mechanistic Simulation of Foam Processes in Porous Media. Society of Petroleum Engineers. doi:10.2118/57678-PA

Rossen, W. R., Venkatraman, A., Johns, R. T., Kibodeaux, K. R., Lai, H., & Tehrani, N. M. (2011). Fractional Flow Theory Applicable to Non-Newtonian Behavior in EOR Processes. *Transport in Porous Media*, 89(2), 213–236. doi:10.1007/s11242-011-9765-2

Rossen, W. R., Ocampo, A. A., Restrepo, A., Cifuentes, H. D., & Marin, J. (2016). Long-Time Diversion in Surfactant-Alternating-Gas Foam Enhanced Oil Recovery From a Field Test. Society of Petroleum Engineers. doi:10.2118/170809-PA

Shan, D., & Rossen, W. R. (2004, June 1). Optimal Injection Strategies for Foam IOR. Society of Petroleum Engineers. doi:10.2118/88811-PA

ter Haar, S. F. (2017). *A Buckley-Leverett Model for Shear-thinning SAG for Enhanced Oil Recovery* (Unpublished). Delft University of Technology, Delft, the Netherlands.

Zhou, Z., & Rossen, W. R. (1995, March 1). Applying Fractional-Flow Theory to Foam Processes at the "Limiting Capillary Pressure." Society of Petroleum Engineers. doi:10.2118/24180-PA

A

Appendix: Matlab Code

A.1. Main Program

```
%% Non-newtonian SAG
% by Sterre ter Haar
% last updated: 2018 January 14
close all; clc
format long

% Input Parameters
% Well Parameters (WP struct)
WP.re=100; WP.rw=0.1;

% Line Parameters (LP struct)
% Parameters affecting the amount of lines analyzed
LP.lin=400; % number of characteristics analyzed after the
   shock (meaning: lower saturations)
LP.r2=linspace(0,log(WP.re),LP.lin+1);
LP.r=LP.r2(2:end);
LP.num_layers=2; % number of layers
LP.num_stops=LP.num_layers+1; % don't touch this
LP.stopsR=logspace(-1,2,LP.num_stops); %the stops based off
   radial logarithmic spacing
LP.end_time=500; % dummy variable - this is to calculate the
   dimensionless position, this doesn't need to be altered to
   graph correctly

% Foam and Petrophysical Parameters (FP struct)
% Kapetas et al. (2017) - Bentheimer sandstone
FP.mu_water=0.001; FP.mu_gas=2e-5; % Density Parameters
```

```
24 FP.krwo=0.39; FP.nw=2.86; % Corey exponenents; relative
      permeability of water
25 FP.krgo=0.59; FP.ng=0.7; % Corey exponenents; relative
      permeability of gas
26 FP.Swr=0.25; FP.Sgr=0.20; % Corey exponenents
27 FP.fmmob=47700; FP.epdry=400; % foam properties
28 FP.fmdry100=0.271; %fmdry value at r_e
29 FP.krw_fmdry_wre=FP.krwo*((FP.fmdry100-FP.Swr)/(1-FP.Swr-FP.
      Sgr))^FP.nw;
30 FP.n=1; %1=newtonian; 0.33; 1.34; 1.67 (Osterloh & Jante 1992;
       Alvarez et al. 2001)
31 FP.C4=(FP.fmdry100-FP.Swr)/((WP.re)^((FP.n-1)/FP.nw));
32
33 %calculates the fmdry throughout the reservoir
34 i=1:1:LP.num_layers;
35 FP.fmdry=FP.Swr+FP.C4*(LP.stopsR(i).^((FP.n-1)/FP.nw));
36
37 %convert LP.stops from r to x_D
38 i=1:1:LP.num_stops;
39 LP.stops=(LP.stopsR(i).^2-WP.rw^2)/(WP.re^2-WP.rw^2);
40
41 sw_shock = Shock(FP,FP.fmdry(1)); %calculates the shock
42 %linear distribution of characteristics below the shock
43 i=1:1:LP.lin+1;
44 LP.spacing=(sw_shock-FP.Swr)/(LP.lin+1);
45 sw=double(sw_shock-(i-1)*LP.spacing)';
46
47 [S,slope]=structureStartEnd(LP,FP,sw); % creates the structure
       containg details about all of the layers
48 FY=diff(slope,1,1); %change in velocity between layers
49 FX=diff(slope,1,2); %change in velocity between
      characteristics
50
51 CM=plotDimTimeVsDimPos(sw,S,LP,FP,1,1); % Plot dimensionless
      time versus dimensionless position
52
53 PD=dimPressureGraphCalculations(sw,S,WP,LP,CM,FP,0.01); % does
       the dimensionless pressure calculations
54 dimPressureGraph(sw,S,WP,LP,CM,FP,PD); % plots the
      dimensionless pressure over time
55
56 %%
57 fw_plot(S,FP,1,0); %fractional-flow plot at r_w
58 fw_plot(S,FP,max(LP.num_layers),0); %fractional-flow plot at
      r_e
59
```

```
60 shockMobility(FP,LP,S) %graphs the mobility of the shock
      throughout x_D
61
62 mobility_versus_radius(S,0.5,WP,LP,CM,FP,1); %graphs the
      mobility of a spreading wave at t_D=0.5
63
64 figure;
65 mobilityOverTime(0.001,0.72,0.74,1,S,WP,LP,CM,FP,'end');
66 mobilityOverTime(eps,eps,10*eps,1,S,WP,LP,CM,FP,'beginning');
67 mobilityOverTime(0.005,0.005,1,1,S,WP,LP,CM,FP,'total');
68
69 mobilityOverDistance(0.005,0.005,1,1,S,WP,LP,CM,FP,'total');
70
71 %% Callable functions - examples of usage
72 fw_plot(S,FP,1,0); % pulls up the water fractional-flow plots
73 [~]=Sw_versus_radius(S,0.1,WP,LP,CM,1,FP); % pulls up a graph
      of Sw versus radius at time t
74 [~]=mobility_versus_radius(S,0.5,WP,LP,CM,FP,1); %graphs the
      mobility of a spreading wave at t_D=0.5
75
76 %% Graphs the fmdry over x_D
77 width = 17.4/2;
78 height = 17.4/2;
79 figname=strcat('fmdry_n',num2str(FP.n*100));
80 figure('units','centimeters','Position',[0 0 width height],'
      name',figname);
81 syms x
82 g=(x^2-WP.rw^2)/(WP.re^2-WP.rw^2);
83 xlim([0, 1])
84 ylim([0.23,0.33])
85 hold on
86 plot(LP.stops(2:end),FP.fmdry,'-')
87 hold on
88 hline=refline(0,0.25);
89 hline.Color = 'r';
90 hold on
91 plot(0,FP.Swr,'b*');
92 hold on
93 legend('fmdry','S_{w_r}')
94 ylabel('fmdry')
95 xlabel('x_D')
96
97 %% Calculates the shock via tangency at every point and
      compares it to the fw(Sw)
98 for i=1:20:LP.num_stops-1
99     sw_shock_p=Shock(FP,FP.fmdry(i));
```

```
100     sw_shock_plot(i)=sw_shock_p;
101     sw_fw_plot(i)=S(i).sw(1);
102 end
103 width = 17.4;
104 height = 17.4/2.8;
105 figname=strcat('shock_difference_n',num2str(FP.n*100));
106 figure('units','centimeters','Position',[0 0 width height],'
       name',figname);
107 plot(FP.fmdry(1:20:end),sw_fw_plot(1:20:end),'g-')
108 hold on
109 plot(FP.fmdry(1:20:end),sw_shock_plot(1:20:end),'r-')
110 xlabel('fmdry')
111 ylabel('S_{w(shock)}')
112 legend('S_w(f_w)','tangency','Location','northwest')
113 tightfigIC
114 print(figname,'-depsc');
115
116 %% plots the fw(sw) curves and fmdry values (only really
       useful for n=1.67)
117 width = 17.4;
118 height = 17.4/2.8;
119 figure('units','centimeters','Position',[0 0 width height],'
       name','fw plot')
120 layernumber=1;
121
122 for k=1:4 %so the legend has the right colors
123     plot(-5,-5,'color',CM(k,:))
124     hold on
125 end
126 plot(-5,-5,'ko')
127 hold on
128 plot(-5,-5,'k^')
129
130 for j=1:4
131     if j==1
132         layernumber=1;
133     elseif j==2
134         layernumber=534;
135     elseif j==3
136         layernumber=659;
137     else
138         layernumber=1000;
139     end
140     sw=S(layernumber).sw;
141     SW = linspace(FP.Swr,1,2000);
142     krw=SW; Krg0=SW; Fw=SW; FM=SW; Krgf=SW; fw=SW;
```

```
143     for i=1:numel(SW) %plots the fractional flow line
144         krw(i) = FP.krwo*((SW(i)-FP.Swr)/(1-FP.Swr-FP.Sgr)).^
               FP.nw;
145         Krg0(i) = real((FP.krgo*((1-SW(i)-FP.Sgr)/(1-FP.Swr-FP
               .Sgr)).^FP.ng));
146         Fw(i) = 0.5+atan(FP.epdry*(SW(i)-FP.fmdry(layernumber)
               ))/pi()-(0.5+atan(FP.epdry*(FP.Swr-FP.fmdry(
               layernumber)))/pi());
147         FM(i) = 1/(1+FP.fmmob*Fw(i));
148         Krgf(i) = Krg0(i)*FM(i);
149         fw(i) = 1/(1+Krgf(i)/krw(i)*FP.mu_water/FP.mu_gas);
150     end
151     plot(SW,fw,'color',CM(j,:))
152     hold on
153     shockie=Shock(FP,FP.fmdry(layernumber));
154     [~, ~, ~, fw, ~] = EPNL(shockie,FP,FP.fmdry(layernumber),
           LP.end_time);
155     plot(shockie,fw,'^','color',CM(j,:))
156     hold on
157     plot(S(layernumber).sw(1),S(layernumber).fw(1),'o','color'
           ,CM(j,:))
158 end
159
160 xlabel('S_w')
161 ylabel('f_w')
162 xlim([FP.Swr 0.27])
163 ylim([0 0.02])
164 legend('fmdry(r_w)=0.254','fmdry=0.258','fmdry=0.262','fmdry(
       r_e)=0.271','calculated by S_w(f_w)','calculated by
       tangency','Location','northwest')
165 tightfigIC
166 figname=strcat('fwplotcomparison_n',num2str(100*FP.n));
167 print(figname,'-depsc');
168
169 %% plots the fmdry values and fractional-flow curves
       throughout x_D
170 width = 17.4;
171 height = 17.4/2.8;
172
173 if FP.n>1
174     figure('units','centimeters','Position',[0 0 width height
           ],'name','fw plot')
175     subplot(1,2,1)
176 else
177     figure('units','centimeters','Position',[0 0 width height
           ],'name','fw plot')
```

```
178 end
179
180 for k=1:4 %so the legend has the right colors
181     plot(-5,-5,'color',CM(k,:))
182     hold on
183 end
184 plot(-5,-5,'ko')
185 hold on
186 plot(-5,-5,'k^')
187
188 for j=1:4
189     if j==1
190         layernumber=1;
191     elseif j==2
192         layernumber=333;
193     elseif j==3
194         layernumber=666;
195     else
196         layernumber=1000;
197     end
198     sw=S(layernumber).sw;
199     SW = linspace(FP.Swr,1,2000);
200     krw=SW; Krg0=SW; Fw=SW; FM=SW; Krgf=SW; fw=SW;
201     for i=1:numel(SW) %plots the fractional flow line
202         krw(i) = FP.krwo*((SW(i)-FP.Swr)/(1-FP.Swr-FP.Sgr)).^
               FP.nw;
203         Krg0(i) = real((FP.krgo*((1-SW(i)-FP.Sgr)/(1-FP.Swr-FP
               .Sgr)).^FP.ng));
204         Fw(i) = 0.5+atan(FP.epdry*(SW(i)-FP.fmdry(layernumber)
               ))/pi()-(0.5+atan(FP.epdry*(FP.Swr-FP.fmdry(
               layernumber)))/pi());
205         Krgf(i) = Krg0(i)*FM(i);
206         fw(i) = 1/(1+Krgf(i)/krw(i)*FP.mu_water/FP.mu_gas);
207     end
208     plot(SW,fw,'color',CM(j,:))
209     hold on
210 end
211
212 xlabel('S_w')
213 ylabel('f_w')
214 xlim([FP.Swr 1])
215 ylim([0 1])
216 if FP.n==0.33
217     legend('fmdry(r_w)=0.356','fmdry=0.312','fmdry=0.286','
          fmdry(r_e)=0.271')
218 elseif FP.n==1.34
```

```
219     legend('fmdry(r_w)=0.259','fmdry=0.262','fmdry=0.266','
           fmdry(r_e)=0.271')
220 end
221 if FP.n>1
222     subplot(1,2,2)
223     for j=1:4
224         if j==1
225             layernumber=1;
226         elseif j==2
227             layernumber=333;
228         elseif j==3
229             layernumber=666;
230         else
231             layernumber=1000;
232         end
233         sw=S(layernumber).sw;
234         SW = linspace(FP.Swr,1,2000);
235         krw=SW; Krg0=SW; Fw=SW; FM=SW; Krgf=SW; fw=SW;
236         for i=1:numel(SW) %plots the fractional flow line
237             krw(i) = FP.krwo*((SW(i)-FP.Swr)/(1-FP.Swr-FP.Sgr)
                   ).^FP.nw;
238             Krg0(i) = real((FP.krgo*((1-SW(i)-FP.Sgr)/(1-FP.
                   Swr-FP.Sgr)).^FP.ng));
239             Fw(i) = 0.5+atan(FP.epdry*(SW(i)-FP.fmdry(
                   layernumber)))/pi()-(0.5+atan(FP.epdry*(FP.Swr-
                   FP.fmdry(layernumber)))/pi());
240             FM(i) = 1/(1+FP.fmmob*Fw(i));
241             Krgf(i) = Krg0(i)*FM(i);
242             fw(i) = 1/(1+Krgf(i)/krw(i)*FP.mu_water/FP.mu_gas)
                   ;
243         end
244         plot(SW,fw,'color',CM(j,:))
245         hold on
246     end
247 end
248
249 xlabel('S_w')
250 ylabel('f_w')
251 if FP.n==1.34
252     xlim([FP.Swr 0.28])
253     ylim([0 0.125])
254 elseif FP.n==1.67
255     xlim([FP.Swr 0.27])
256     ylim([0 0.02])
257 else
258     xlim([FP.Swr 1])
```

```
259     ylim([0 1])
260 end
261 tightfigIC
262 figname=strcat('fwplotcomparison_n',num2str(100*FP.n));
263 print(figname,'-depsc');
```

A.2. Functions

A.2.1. dimPressureGraphCalculations.m

```
function [PD]=dimPressureGraphCalculations(sw,S,WP,LP,CM,FP,step)
% dimPressureGraphCalculations calculates the dimensionless pressure over
% xD and returns PD.
% PD: 1=PD, 2=shocklead, 3=interp lead, 4=char. pairs, 5=zero-vel pair,
% 6=time
%
% [PD]=dimPressureGraphCalculations(sw,S,WP,LP,CM,FP,step)
% step=step value for xD increment


% Calculates a 3D matrix PDSLICES that contains all the values needed for a PD
% PDSLICES(a,b,c): a=row, b=column (1=dim pos, 2=radius, 3=Sw,
% 4=lambda_rt), c=time (slotted by increment)
max_step=1/step; counter=1;
PDSLICES=zeros(numel(sw),4,max_step);
for i=step:step:1
    PDSLICES(:,:,counter)=Sw_versus_radius(S,i,WP,LP,CM,0);
    counter=counter+1;
end

% Calculates and plots the dimensionless pressure (PD) over time
bottom=FP.mu_water*log(WP.re/WP.rw);
k_gas=FP.krwo*((1-FP.Swr)/(1-FP.Swr-FP.Sgr))^FP.nw;
%PD: 1=PD, 2=shocklead, 3=interp lead, 4=char. pairs, 5=zero vel pair,
%6=time
PD=zeros(6,max_step+1);
PD(1,1)=FP.mu_water*log(WP.re/WP.rw)/bottom;
PD(2,1)=FP.mu_water*log(WP.re/WP.rw)/bottom;
for t=1:counter-1 % calculates the PDs, stepping through time
    PD(6,t+1)=t*step;
    pairsummation=0; shocklead=0; interplead=0;
    if PDSLICES(1,1,t)~=-9 % condition: while shock is still in range
        shocklead=FP.mu_water*log(WP.re/PDSLICES(1,2,t));
        i=1;
        PD(2,t+1)=shocklead/bottom;
```

```
36      elseif PDSLICES(end,1,t)==-9 % condition: no shock nor
           characteristics exist
37          PD(5,t+1)=(FP.mu_gas/k_gas)*log(WP.re/WP.rw)/bottom;
38          PD(1,t+1)=PD(5,t+1);
39      else % condition: shock is no longer in range, but
           characteristics still exist
40          i=1;
41          while PDSLICES(i,1,t)<0 % searches for the first
               characteristic that still exists
42              i=i+1;
43          end
44          lambdart_top=Interpolate(S(LP.num_stops).x(i-1),S(LP.
               num_stops).x(i),t*step,S(LP.num_stops-1).lambdart(i
               -1),S(LP.num_stops-1).lambdart(i));
45          interplead=0.5*((1/PDSLICES(i,4,t))+(1/lambdart_top))*
               log(WP.re/PDSLICES(i,2,t));
46          PD(3,t+1)=interplead/bottom;
47      end
48
49      if PDSLICES(end,1,t)~=-9 % case: characteristics exist
50          howmuch=0;
51          while i<numel(PDSLICES(:,1,1)) % calculates the number
               of present characteristics
52              j=i+1;
53              while PDSLICES(j,1,t)==-9
54                  j=j+1;
55              end
56              howmuch=howmuch+1;
57              pairsummation=pairsummation+0.5*((1/PDSLICES(i,4,t
                   )+(1/PDSLICES(j,4,t))))*(log(PDSLICES(i,2,t)/
                   PDSLICES(j,2,t)));
58              PS(t,i)=pairsummation;
59              i=j;
60          end
61          PD(7,t+1)=howmuch;
62          if PDSLICES(end,1,t)>-9
63              zerovelpair=0.5*(FP.mu_gas/k_gas+(1/PDSLICES(j,4,t
                   ))*log(PDSLICES(j,2,t)/WP.rw));
64              PD(1,t+1)=(shocklead+interplead+pairsummation+
                   zerovelpair)/bottom;
65              PD(4,t+1)=pairsummation/bottom;
66              PD(5,t+1)=zerovelpair/bottom;
67          end
68      elseif (PDSLICES(end,1,t)==-9 && PDSLICES(1,1,t)~=-9) %
           case: shock exists, but no characteristics
```

```
69          PD(5,t+1)=0.5*(FP.mu_gas/k_gas+(1/PDSLICES(1,4,t)))*
               log(WP.re/PDSLICES(1,2,t))/bottom;
70          PD(1,t+1)=PD(2,t+1)+PD(5,t+1);
71      end
72  end
73  end
```

A.2.2. dimPressureGraph.m

```
function [PD]=dimPressureGraph(sw,S,WP,LP,CM,FP,PD)
% dimPressureGraph plots the dimensionless pressure over xD and returns PD
% PD: 1=PD, 2=shocklead, 3=interp lead, 4=char. pairs, 5=zero-vel pair,
% 6=time
%
% [PD]=dimPressureGraph(sw,S,WP,LP,CM,FP,PD)

step=PD(6,2);

width = 17.4;
height = 17.4/2.5;
figname=strcat('PDInjectivity_n',num2str(FP.n*100));
figure('units','centimeters','Position',[0 0 width height],'name',figname);
ax=subplot(19,1,1:12);
plot(0:step:1,PD(1,:),'g.--')
hold on
plot(0:step:1,PD(2,:),'b.--')
hold on
plot(0:step:1,PD(1,:)-PD(2,:),'r.--')
hold on
xlabel('t_D')
ylabel('P_D')
legend('P_{total}','P_{foam bank}','P_{water}','Location','bestoutside')
lgd=legend('P_{total}','P_{water}','P_{foam bank}','Box','Off');
hold on

hL=subplot(19,1,19);
poshL=get(hL,'position');
lgd=legend(ax);
lgd.Orientation='horizontal';
set(lgd,'position',poshL); % Adjusting legend's position
axis(hL,'off'); % Turning its axis off

print(figname,'-depsc')
end
```

A.2.3. EPNL.m

```
function [dfwdSw, dimpos, lambda_rt, fw, slope] = EPNL(swj,FP,fmdry,end_time)
% EPNL calculates the End Point of the New Layer (y-axis) by returning
% the slope.
%
% [dfwdSw, dimpos, lambda_rt, fw, slope] = EPNL(swj,FP,fmdry,LP.end_time)
% swj = the water saturation
% fmdry is the fmdry at the layer, not the whole array

SW(1)=swj;
SW(2)=swj-10^-8; % defines the SW used for the calculations
for i=2:-1:1
    krw(i) = FP.krwo*((SW(i)-FP.Swr)/(1-FP.Swr-FP.Sgr)).^FP.nw ; % calculates end-point relative permeability
    if SW(1)>=1-FP.Sgr
        Krg0(i)=0;
    else
        Krg0(i) = (FP.krgo*((1-SW(i)-FP.Sgr)/(1-FP.Swr-FP.Sgr)).^FP.ng); % calculates krg0
    end
    Fw(i) = (0.5+atan(FP.epdry*(SW(i)-fmdry))/pi())-(0.5+atan(FP.epdry*(FP.Swr-fmdry))/pi()); % calculates FM
    FM(i) = 1/(1+FP.fmmob*Fw(i)); % calculates FM
    Krgf(i) = Krg0(i)*FM(i); % calculates Krgf
    fw_sym(i) = 1/(1+Krgf(i)/krw(i)*FP.mu_water/FP.mu_gas); % calculates fw
end

dfwdSw_sym = (fw_sym(2)-fw_sym(1))/(SW(2)-(SW(1))); % calculates dfw/dSw
Lambda_rt = (krw(1)/FP.mu_water)+Krgf(1)/FP.mu_gas; % calculates Lambda_rt
newdimpos_sym = end_time*dfwdSw_sym;
dfwdSw = double(dfwdSw_sym);
dimpos = double (newdimpos_sym);
lambda_rt = double(Lambda_rt);
fw = double(fw_sym(1));
x = linspace(0,end_time,end_time*25);
coefficients = polyfit(x, dfwdSw*x, 1);
slope = coefficients(1);
end
```

A.2.4. fw_plot.m

```
function [] = fw_plot(S,FP,layernumber,p)
% fw_plot plots the fractional-flow curve at a given fmdry as well as
% (optional) the water saturations of the characteristics
%
% [] = fw_plot(S,FP,i,p)
% i is the layer number
% p: 1=slopes of characteristics are shown, 0=off

sw=S(layernumber).sw;
width = 17.4;
height = 17.4/2.8;
figure('units','centimeters','Position',[0 0 width height],'name',strcat('fw plot at fmdry=',num2str(FP.fmdry(layernumber))))
subplot(1,2,1)
CM = lines(numel(sw)+1); % uses colormap lines to identify each sw on the plot

for i=numel(sw):-1:1 %plots the sw points of interest
    krw(i) = FP.krwo*((sw(i)-FP.Swr)/(1-FP.Swr-FP.Sgr)).^FP.nw; % calculates krw - end-point relative permeability
    Krg0(i) = (FP.krgo*((1-sw(i)-FP.Sgr)/(1-FP.Swr-FP.Sgr)).^FP.ng); % calculates krg0
    Fw(i) = (0.5+atan(FP.epdry*(sw(i)-FP.fmdry(layernumber)))/pi())-(0.5+atan(FP.epdry*(FP.Swr-FP.fmdry(layernumber)))/pi()); % calculate Fw - dryout function
    FM(i) = 1/(1+FP.fmmob*Fw(i)); % calculates FM - foam mobility reduction factor
    Krgf(i) = Krg0(i)*FM(i); % calculates Krgf
    FW(i) = 1/(1+Krgf(i)/krw(i)*FP.mu_water/FP.mu_gas); % calculates fw - water fractional flow
    legendInfo{i} = strcat('S_{w}=',num2str(double(sw(i))));
    plot(sw(i),FW(i),'*','color',CM(i+1,:))
    hold on
end

SW = linspace(FP.Swr,1-FP.Sgr,500);
krw=SW; Krg0=SW; Fw=SW; FM=SW; Krgf=SW; fw=SW;

for i=1:numel(SW) %plots the fractional flow line
    krw(i) = FP.krwo*((SW(i)-FP.Swr)/(1-FP.Swr-FP.Sgr)).^FP.nw; % calculates krw - end-point relative permeability
    Krg0(i) = real((FP.krgo*((1-SW(i)-FP.Sgr)/(1-FP.Swr-FP.Sgr)).^FP.ng)); % calculates krg0
```

```
34      Fw(i) = 0.5+atan(FP.epdry*(SW(i)-FP.fmdry(layernumber)))/
           pi()-(0.5+atan(FP.epdry*(FP.Swr-FP.fmdry(layernumber)))
           /pi()); % calculate Fw - dryout function
35      FM(i) = 1/(1+FP.fmmob*Fw(i)); % calculates FM - foam
           mobility reduction factor
36      Krgf(i) = Krg0(i)*FM(i); % calculates Krgf
37      fw(i) = 1/(1+Krgf(i)/krw(i)*FP.mu_water/FP.mu_gas); %
           calculates fw - water fractional flow
38  end
39  plot(SW,fw,'color',CM(1,:))
40
41  % draws a red box around the SW points
42  % plot([min(sw)*0.95 min(sw)*0.95],[min(FW)*0.9 max(FW)*1.1],'
       r-')
43  % plot([max(sw)*1.05 max(sw)*1.05],[min(FW)*0.9 max(FW)*1.1],'
       r-')
44  % plot([min(sw)*0.95 max(sw)*1.05],[min(FW)*0.9 min(FW)*0.9],'
       r-')
45  % plot([min(sw)*0.95 max(sw)*1.05],[max(FW)*1.1 max(FW)*1.1],'
       r-')
46
47  text(1-FP.Sgr,0.95,'I','color','green','HorizontalAlignment','
       center','FontWeight','bold')
48  text(FP.Swr*0.75,0.05,'J','color','green','FontWeight','bold')
49
50  if p==1
51      for nSw=1:numel(sw)
52          hold on
53          m = S(layernumber).slopes(nSw); b = S(layernumber).fw(
               nSw);
54          fplot(@(x)m*(x-S(layernumber).sw(nSw))+b, [S(
               layernumber).sw(nSw) 1],'color',CM(nSw+1,:));
55      end
56      hold on
57      plot([S(layernumber).sw(1),1],[S(layernumber).fw(1),1],'m-
           ')
58      hold on
59  end
60
61  xlabel('S_w')
62  ylabel('f_w')
63  xlim([0 1])
64  ylim([0 1])
65
66  subplot(1,2,2)
67  for i=1:numel(sw) %plots the sw points of interest
```

```
68      krw(i) = FP.krwo*((sw(i)-FP.Swr)/(1-FP.Swr-FP.Sgr)).^FP.nw
           ; % calculates krw - end-point relative permeability
69      Krg0(i) = (FP.krgo*((1-sw(i)-FP.Sgr)/(1-FP.Swr-FP.Sgr)).^
           FP.ng); % calculates krg0
70      Fw(i) = 0.5+atan(FP.epdry*(sw(i)-FP.fmdry(layernumber)))/
           pi()-(0.5+atan(FP.epdry*(FP.Swr-FP.fmdry(layernumber)))
           /pi()); % calculate Fw - dryout function
71      FM(i) = 1/(1+FP.fmmob*Fw(i)); % calculates FM - foam
           mobility reduction factor
72      Krgf(i) = Krg0(i)*FM(i); % calculates Krgf
73      FW(i) = 1/(1+Krgf(i)/krw(i)*FP.mu_water/FP.mu_gas); %
           calculates fw - water fractional flow
74      legendInfo{i} = strcat('S_{w}=',num2str(double(sw(i))));
75      plot(sw(i),FW(i),'*','color',CM(i+1,:))
76      hold on
77  end
78  SW = linspace(FP.Swr,max(sw)*1.1,500);
79  krw=SW; Krg0=SW; Fw=SW; FM=SW; Krgf=SW; fw=SW;
80
81  for i=1:numel(SW) %plots the fractional flow line
82      krw(i) = real(FP.krwo*((SW(i)-FP.Swr)/(1-FP.Swr-FP.Sgr)).^
           FP.nw); % calculates krw - end-point relative
           permeability
83      Krg0(i) = real((FP.krgo*((1-SW(i)-FP.Sgr)/(1-FP.Swr-FP.Sgr
           )).^FP.ng)); % calculates krg0
84      Fw(i) = (0.5+atan(FP.epdry*(SW(i)-FP.fmdry(layernumber)))/
           pi())-(0.5+atan(FP.epdry*(FP.Swr-FP.fmdry(layernumber))
           )/pi()); % calculate Fw - dryout function
85      FM(i) = 1/(1+FP.fmmob*Fw(i)); % calculates FM - foam
           mobility reduction factor
86      Krgf(i) = Krg0(i)*FM(i); % calculates Krgf
87      fw(i) = 1/(1+Krgf(i)/krw(i)*FP.mu_water/FP.mu_gas); %
           calculates fw - water fractional flow
88  end
89
90  plot(SW,fw,'color',CM(1,:))
91
92  if p==1
93      xlim([min(SW)*0.95 max(SW)*1])
94      ylim([min(FW)*0.95 max(FW)*1.05])
95
96      for nSw=1:numel(sw)
97          hold on
98          m = S(layernumber).slopes(nSw); b = S(layernumber).fw(
               nSw);
```

```
99          fplot(@(x)m*(x-S(layernumber).sw(nSw))+b, [S(
               layernumber).sw(nSw) 1],'color',CM(nSw+1,:));
100     end
101
102     hold on
103     plot([S(layernumber).sw(1),1],[S(layernumber).fw(1),1],'m-
           ')
104
105 end
106
107 xlabel('S_w')
108 ylabel('f_w')
109 xlim([min(SW)*0.95 max(SW)*1])
110 ylim([min(FW)*0.95 max(FW)*1.05])
111 tightfigIC;
112 figname=strcat('fwplot_fmdry',num2str(layernumber),'_n',
       num2str(100*FP.n));
113 print(figname,'-depsc')
114 end
```

A.2.5. fw_to_Sw.m

```
function swj = fw_to_Sw(j,FP,fmdry,guess)
% fw_to_Sw calculates the water saturation from the fractional-flow
%
% swj = fw_to_Sw(j,FP,fmdry,guess)
% j = fw at the point for which Sw is unknown
% guess = initial guess (recommended: last known Sw)

swe   = @(sw)(sw-FP.Swr)/(1-FP.Swr-FP.Sgr);
krw   = @(sw)(FP.krwo*swe(sw).^FP.nw);
lambda_w = @(sw)(krw(sw)./FP.mu_water);
krg   = @(sw)(FP.krgo*(1-swe(sw)).^FP.ng);
FM    = @(sw)(1+FP.fmmob*((0.5+atan(FP.epdry.*(sw-fmdry))/pi())
   -(0.5+atan(FP.epdry.*(FP.Swr-fmdry))/pi())));
krgf = @(sw)(krg(sw)./FM(sw));
lambda_f = @(sw)(real(krgf(sw)./FP.mu_gas));
fsurfactant    = @(sw)(1./(1+(lambda_f(sw))./(lambda_w(sw))));

syms x_SYM;
assume(x_SYM, 'real')
eq=fsurfactant(x_SYM)==j;
swj=double(vpasolve(eq,x_SYM,guess));
end
```

A.2.6. Interpolate.m

```
function [x3]=Interpolate(a,b,c,x1,x2)
% Interpolate interpolates between two characteristics at xD=1
%
% [x3]=Interpolate(a,b,c,x1,x2)
% a and b are the start and end points of the dimensionless position
% c is the dimensionless position of the interpolate characteristic
% x1 and x2 are the lambda values of a and b respectively

multiplier=(c-a)*(b-a)^-1;
x3=x1+(x2-x1)*multiplier;
end
```

A.2.7. Intersect.m

```
function [isInSegment,xi,yi]=Intersect(s1x1,s1x2,s2x1,s2x2,y1,y2)
% Intersect checks if/where two lines intersect
%
% [isInSegment,xi,yi] = Intersect(s1x1,s1x2,s2x1,s2x2,y1,y2)
% s1x1 and s1x2 refer to the x-positions of the first line
% s2x1 and s2x2 refer to the x-positions of the second line
% y1 and y2 refer to the beginning and end of the layers
% Basis of code is taken from: http://stackoverflow.com/questions/2050850/matlab-find-point-of-intersection-between-two-vectors
% Returns test (1=true, they intersect in the segment; 0=false)
% Returns p1,p2 (p1=x,p2=y; coordinates of the intersection)

x = [s1x1 y1; s1x2 y2];  % Starting points in first row, ending points in second row
y = [s2x1 y1; s2x2 y2];
dx = diff(x);  % Take the differences down each column
dy = diff(y);
denominator = dx(1)*dy(2)-dy(1)*dx(2);  % Precompute the denominator
if denominator == 0
    error('The lines are parallel.');
end
% close all;
% figure;
% plot(x(:,1),x(:,2))
% hold on
% plot(y(:,1),y(:,2))
% legend('show')
ua = (dx(2)*(x(1,2)-y(1,2))-dy(2)*(x(1)-y(1))) / denominator;
xi = x(1)+ua*dx(1);
yi = x(1,2) + ua*dx(2);
isInSegment = all((xi>=s1x1) & (xi<=s1x2) & (xi>s2x1 | abs(xi-s2x2)<1e-6) & (xi<s2x2 | abs(xi-s2x2)<1e-6) & (yi>=y1) & (yi<=y2) & (xi>0));
end
```

A.2.8. mobility_versus_radius.m

```
function [results] = mobility_versus_radius(S,t,WP,LP,CM,FP,
   mobgraph)
% mobility_versus_radius plots the Sw versus r at a given time
    and
% creates a results matrix with useful numbers.
%
% [results] = Sw_versus_radius(S,t,WP,LP,CM)
% t is the given dimensionless position slice between 0 and 1
% mobgraph: set to 1 to graph mobility versus radius
% results returns for each characteristic:
%  - the x position when it intersects with time t
%  - the corresponding radius
%  - the Sw at the corresponding time
%  - the lambda_rt value at the corresponding time
% eliminated characteristics are denoted with a -9

test=0; k=0;
if mobgraph==1
    width = 17.4;
    height = 17.4/2.8;

    f=figure('units','centimeters','Position',[0 0 width
       height],'name','Mobility versus Radius');

end
[~,n]=size(S);
results=ones(numel(S(1).x),4)*-9;
a=1; % starting char

while test==0 && k<n-1 && a<=LP.lin
    k=k+1;
    [test,~,p2]=Intersect(S(k).x(a),S(k+1).x(a),t,t,S(k).y,S(k
       +1).y);
    if k==n-1 && test==0
        k=0;
        a=a+1;
    end
end
if p2<=1.0 % checks to make sure that the characteristic is
   still in bounds
    results(a,1)=p2;
    results(a,2)=sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw^2);
    results(a,3)=S(k).sw(a);
    results(a,4)=S(k).lambdart(a);
    if mobgraph==1
```

```
41          plot(p2,S(k).lambdart(a),'*','color',CM(2,:))
42          legendInfo{1} = strcat('S_{w}=',num2str(double(S(k).
               lambdart(a))));
43      end
44  end
45  if mobgraph==1
46      hold on
47  end
48  for j=a+1:numel(S(1).x)
49      for i=k:-1:1
50          [test,~,p2]=Intersect(S(i).x(j),S(i+1).x(j),t,t,S(i).y
               ,S(i+1).y);
51          if test==1
52              if p2<results(a,1)
53                  results(j,1)=p2;
54                  results(j,2)=sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw
                       ^2);
55                  results(j,3)=S(i).sw(j);
56                  results(j,4)=S(i).lambdart(j);
57                  if mobgraph==1
58                      plot(p2,S(i).lambdart(j),'*','color',CM(j
                           +1,:))
59                      legendInfo{j} = strcat('S_{w}=',num2str(
                           double(S(k).lambdart(j))));
60                      k=i; % optimizes the searching by limiting
                            the next cycle to the length of this
                           one
61                  end
62              end
63          end
64      end
65  end
66  if mobgraph==1
67      hold on
68      i=1;
69      while results(i,1)==-9
70          i=i+1;
71      end
72      set(gca,'YScale','log')
73      plot([results(i,1),1],[1/FP.mu_water,1/FP.mu_water],'*-')
74      hold on
75      plot([results(i,1),results(i,1)],[results(i,4),1/FP.
           mu_water],'*--')
76      xlabel('x_D')
77      ylabel('total relative mobility ({\lambda}_{rt})')
78      xlim([0 1])
```

```
79      figname=strcat('mobilityProfile_n',num2str(FP.n*100));
80      set(gca,'YScale','log')
81      grid on
82      tightfigIC;
83      print(figname,'-depsc')
84  end
85  end
```

A.2.9. mobility_versus_radius_cumulative.m

```
function [results] = mobility_versus_radius_cumulative(S,t,WP,LP,CM,FP,mobgraph)
% mobility_versus_radius_cumulative plots Sw versus r at a given time
% and creates a results matrix with useful numbers.
%
% [results] = Sw_versus_radius(S,t,WP,LP,CM)
% t is the given time between 0 and 1
% mobgraph: set to 1 to graph mobility versus radius
% results returns for each characteristic:
%  - the x position when it intersects with time t
%  - the corresponding radius
%  - the Sw at the corresponding time
%  - the lambda_rt value at the corresponding time
% eliminated characteristics are denoted with a -9

test=0; k=0;
if mobgraph==1
    width = 17.4;
    height = 17.4/2.8;

    f=figure('units','centimeters','Position',[0 0 width height],'name','Mobility versus Radius');

end

for t=0.1:0.1:1
    [~,n]=size(S);
    results=ones(numel(S(1).x),4)*-9;
    a=1; % starting char

    while test==0 && k<n-1 && a<=LP.lin
        k=k+1;
        [test,~,p2]=Intersect(S(k).x(a),S(k+1).x(a),t,t,S(k).y,S(k+1).y);
        if k==n-1 && test==0
            k=0;
            a=a+1;
        end
    end
    if p2<=1.0 % checks to make sure that the characteristic is still in bounds
        results(a,1)=p2;
        results(a,2)=sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw^2);
        results(a,3)=S(k).sw(a);
```

```
41          results(a,4)=S(k).lambdart(a);
42          if mobgraph==1
43              plot(sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw^2),S(k).
                    lambdart(a),'*','color',CM(2,:))
44              legendInfo{1} = strcat('S_{w}=',num2str(double(S(k
                    ).lambdart(a))));
45          end
46      end
47      if mobgraph==1
48          hold on
49      end
50      for j=a+1:numel(S(1).x)
51          for i=k:-1:1
52              [test,~,p2]=Intersect(S(i).x(j),S(i+1).x(j),t,t,S(
                    i).y,S(i+1).y);
53              if test==1
54                  if p2<results(a,1)
55                      results(j,1)=p2;
56                      results(j,2)=sqrt(p2*(WP.re^2-WP.rw^2)+WP.
                            rw^2);
57                      results(j,3)=S(i).sw(j);
58                      results(j,4)=S(i).lambdart(j);
59                      if mobgraph==1
60                          plot(sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw
                                ^2),S(i).lambdart(j),'*-','color',
                                CM(j+1,:))
61                          legendInfo{j} = strcat('S_{w}=',
                                num2str(double(S(k).lambdart(j))));
62                          k=i; % optimizes the searching by
                                limiting the next cycle to the
                                length of this one
63                      end
64                  end
65              end
66          end
67      end
68      if mobgraph==1
69          hold on
70          i=1;
71          while results(i,1)==-9
72              i=i+1;
73          end
74          plot([results(i,2),WP.re],[1/FP.mu_water,1/FP.mu_water
                ],'*-')
75          hold on
```

```
76          plot([results(i,2),results(i,2)],[results(i,4),1/FP.
                mu_water],'*--')
77          xlabel('radius (m)')
78          ylabel('mobility ({\lambda}_{rt})')
79          xlim([0 100])
80      end
81      hold on
82  end
83  set(gca,'YScale','log')
84  grid on
85  figname=strcat('mobilityProfile_n',num2str(FP.n*100));
86  print(figname,'-depsc')
87  tightfigIC;
88  end
```

A.2.10. mobility_versus_sw.m

```
function [results] = mobility_versus_sw(S,t,WP,LP,CM,fmmob,
   mobgraph)
% mobility_versus_sw plots the mobility versus sw at a given
   time
% (if mobgraph=1) and creates a results matrix with useful
   numbers.
%
% [results] = mobility_versus_sw(S,t,WP,LP,CM,fmmob,mobgraph)
% t is the given time between 0 and 1
% mobgraph: set to 1 to graph mobility versus radius
% results returns for each characteristic:
%  - the x position when it intersects with time t
%  - the corresponding radius
%  - the Sw at the corresponding time
%  - the lambda_rt value at the corresponding time
% eliminated characteristics are denoted with a -9

test=0; k=0;
if mobgraph==1
    f=figure('units','normalized','outerposition',[0 0 1 1]);
end
[~,n]=size(S);
results=ones(numel(S(1).x),4)*-9;
a=1; % starting char

while test==0 && k<n-1 && a<=LP.lin
    k=k+1;
    [test,~,p2]=Intersect(S(k).x(a),S(k+1).x(a),t,t,S(k).y,S(k
       +1).y);
    if k==n-1 && test==0
        k=0;
        a=a+1;
    end
end
if p2<=1.0 % checks to make sure that the characteristic is
   still in bounds
    results(a,1)=p2;
    results(a,2)=sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw^2);
    results(a,3)=S(k).sw(a);
    results(a,4)=S(k).lambdart(a);
    if mobgraph==1
        plot(S(k).sw(a),S(k).lambdart(a),'*','color',CM(2,:))
        legendInfo{1} = strcat('S_{w}=',num2str(double(S(k).
           lambdart(a))));
    end
```

```
40 end
41 if mobgraph==1
42     hold on
43 end
44 for j=a+1:numel(S(1).x)
45     for i=k:-1:1
46         i;
47         [test,~,p2]=Intersect(S(i).x(j),S(i+1).x(j),t,t,S(i).y
               ,S(i+1).y);
48         if test==1
49             if p2<results(a,1)
50                 results(j,1)=p2;
51                 results(j,2)=sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw
                       ^2);
52                 results(j,3)=S(i).sw(j);
53                 results(j,4)=S(i).lambdart(j);
54                 if mobgraph==1
55                     plot(S(k).sw(a),S(i).lambdart(j),'*','
                           color',CM(j+1,:))
56                     legendInfo{j} = strcat('S_{w}=',num2str(
                           double(S(k).lambdart(j))));
57                     k=i; % optimizes the searching by limiting
                            the next cycle to the length of this
                           one
58                 end
59             end
60         end
61     end
62 end
63 if mobgraph==1
64     hold on
65     i=1;
66     while results(i,1)==9
67         i=i+1;
68     end
69     plot(0,fmmob)
70     hold on
71     plot(1,1000)
72     xlabel('Sw')
73     ylabel('mobility ({\lambda}_{rt})')
74     title(strcat('Mobility profile at dimensionless time t=',
           num2str(t)))
75     print -depsc mobilityprofile
76 end
77 set(gca,'YScale','log')
78 end
```

A.2.11. mobilityOverDistance.m

```
function []=mobilityOverDistance(step,startStep,endStep,
   charnumber,S,WP,LP,CM,FP,view)
%mobilityOverDistance plots the mobility profile of one
%characteristic over x_D
%
%mobilityOverDistance(step,startStep,endStep,charnumber,S,WP,
   LP,CM,FP,view)
%view: string you can use to mention if its beginning, end,
   total, etc.

mobs=nan(1,int16(endStep/step));

for i=startStep:step:endStep
    [results,mobs]=mobility_versus_radius_unified(S,i,WP,LP,CM
       ,FP,step,mobs,charnumber);
end

close;
width = 17.4; height = 17.4/2.8;
figname=strcat('Mobility versus Radius over time for n_',
   num2str(FP.n),' char_',num2str(charnumber));
figure('units','centimeters','Position',[0 0 width height],'
   name',figname);
plot(mobs(1,:)',mobs(2,:)','*-')
ylabel(strcat('{\lambda}_{rt} of char. #',num2str(charnumber))
   )
xlabel('x_D')
figtitle=strcat('mobprofDistance_n',num2str(FP.n*100),'_char',
   num2str(charnumber),'_',view);
print(figtitle,'-depsc');
end
```

A.2.12. mobilityOverTime.m

```
function []=mobilityOverTime(step,startStep,endStep,charnumber,S,WP,LP,CM,FP,view)
%mobilityOverTime plots the mobility profile of one
%characteristic over x_D
%
%mobilityOverTime(step,startStep,endStep,charnumber,S,WP,LP,CM,FP,view)
%view: string you can use to mention if its beginning, end, total, etc.

mobs=nan(1,int16(endStep/step));

for i=startStep:step:endStep
    [results,mobs]=mobility_versus_radius_unified(S,i,WP,LP,CM,FP,step,mobs,charnumber);
end

close;

width = 17.4; height = 17.4/2.8;
figname=strcat('Mobility versus Radius over time for n_',num2str(FP.n),' char_',num2str(charnumber));
figure('units','centimeters','Position',[0 0 width height],'name',figname);
y=linspace(step,endStep,(endStep)/step);
plot(y,mobs(2,:)','*-')
ylabel(strcat('{\lambda}_{rt} of char. #',num2str(charnumber)))
xlabel('t_D')
figtitle=strcat('mobprofTime_n',num2str(FP.n*100),'_char',num2str(charnumber),'_',view);
print(figtitle,'-depsc');
end
```

A.2.13. plotDimTimeVsDimPos.m

```
function [CM]=plotDimTimeVsDimPos(sw,S,LP,FP,nolayers,nolabels)
% plotDimTimeVsDimPos creates the xd-td graph and returns CM
% CM is the colors list used for plotting to keep all graphs consistent.
%
% [CM]=plotDimTimeVsDimPos(sw,S,LP,FP,nolayers,nolabels)
% options: 0 or 1
% if nolayers=1; lines denoting the layers aren't plotted
% if nolabels=1; fmdry isn't labeled per layer

% figure properties
width = 17.4;
height = 17.4/2.8;
figname=strcat('DtversusDp_n',num2str(FP.n*100));
figure('units','centimeters','Position',[0 0 width height],'name',figname);

CM = lines(numel(sw)+5); % uses colormap 'lines' to identify each sw on the plot
for i=1:numel(sw) % plots the lines of the first layer
    for k=1
        if S(k).cross(i)==1
            plot([S(k).x(i) p1],[S(k).y p2],'color',CM(i+1,:))
            hold on
        elseif S(k).cross(i)==0 && S(k).x(i)<1
            plot([S(k).x(i) S(k+1).x(i)],[S(k).y S(k+1).y],'color',CM(i+1,:))
            hold on
        end
    end
end
% legend(legendInfo{:,numel(LP.stops)-1},'Location','southoutside'); % plots legend immediately so the lines match the colors on the legend
for i=1:numel(sw) % plots the lines of the layers
    for k=1:LP.num_layers
        if S(k).cross(i)==1
            [~,p1,p2]=Intersect(S(k).x(1),S(k+1).x(1),S(k).x(i),S(k+1).x(i),S(k).y,S(k+1).y);
            plot([S(k).x(i) p1],[S(k).y p2],'color',CM(i+1,:))
            hold on
        elseif S(k).cross(i)==0 && S(k).x(i)<1
            if S(k).x(i)>=S(k).x(1)
```

```
                plot([S(k).x(i) S(k+1).x(i)],[S(k).y S(k+1).y
                    ],'color',CM(i+1,:))
                hold on
            end
        end
    end
end

hold on
xlim([0 1])
if nolayers==0 % adds lines denoting the layers on the plot
    for k=2:LP.num_layers
        refline(0,LP.stops(k))
        hold on
    end
end
if nolabels==0 % labels the layers on the plot
    for k=1:LP.num_layers
        text(1,LP.stops(k)+(LP.stops(k+1)-LP.stops(k))/2,
            strcat({' fmdry = '},num2str(FP.fmdry(k),3)))
    end
end
xlabel('x_D')
ylim([0 max(LP.stops)])
ylabel('t_D')
if nolabels==0
    tightfigxdxp;
else
    tightfigIC;
end
print(figname,'-depsc');
end
```

A.2.14. Shock.m

```
function sw_shock = Shock(FP,fmdry)
% Shock returns where the shock is with 32 digit precision.
% The calculation uses a stepsize of 10e-8.
%
% sw_shock = Shock(FP,fmdry)
% fmdry is the fmdry in that layer, not the whole array.

syms x;
digits(32)
assume(x, 'real')
SW(1)=x; SW(2)=x-10^-8; % defines the SW used for the
   calculations
for i=2:-1:1
    krw(i) = FP.krwo*((SW(i)-FP.Swr)/(1-FP.Swr-FP.Sgr)).^FP.nw
       ; % calculates krw - end-point relative permeability
    Krg0(i) = (FP.krgo*((1-SW(i)-FP.Sgr)/(1-FP.Swr-FP.Sgr)).^
       FP.ng); % calculates krg0
    Fw(i) = (0.5+atan(FP.epdry*(SW(i)-fmdry))/pi())-(0.5+atan(
       FP.epdry*(FP.Swr-fmdry))/pi()); % calculate Fw - dryout
        function
    FM(i) = 1/(1+FP.fmmob*Fw(i)); % calculates FM - foam
       mobility reduction factor
    Krgf(i) = Krg0(i)*FM(i); % calculates Krgf
    fw(i) = 1/(1+(Krgf(i)/krw(i))*(FP.mu_water/FP.mu_gas)); %
       calculates fw - water fractional flow
end
dfwdSw = (fw(2)-fw(1))/(SW(2)-(SW(1))); % calculates dfw/dSw
check = fw(1)+dfwdSw*(1-SW(1))-1; % calculates the check
eq=check==0;
if FP.n==1
    sw_shock=double(vpasolve(eq,x,[FP.Swr FP.fmdry100]));
    if isempty(sw_shock)==1
        sw_shock=double(vpasolve(eq,x,[0 1]));
    end
elseif FP.n<1
    sw_shock=double(vpasolve(eq,x,[FP.Swr 1-FP.Sgr]));
    if isempty(sw_shock)==1
        sw_shock=double(vpasolve(eq,x,[0 1]));
    end
elseif FP.n>1
    sw_shock=double(vpasolve(eq,x,[FP.Swr 1-FP.Sgr]));
    if isempty(sw_shock)==1
        sw_shock=double(vpasolve(eq,x,[0 1]));
    end
end
```

```
39 if isempty(sw_shock)==1
40     counter=0;
41     while (isempty(sw_shock)==1 & counter<1)
42         sw_shock=double(vpasolve(eq,x,[counter counter+0.2]))
43         counter=counter+0.2;
44     end
45 end
46 end
```

A.2.15. shockMobility.m

```
function []=shockMobility(FP,LP,S)
for i=1:max(LP.num_layers)
    x(i)=S(i).x(1);
    y(i)=S(i).lambdart(1);
end
width = 17.4; height = 17.4/2.8;
figname=strcat('Mobility versus Radius over time for n_',
   num2str(FP.n),' char_',num2str(1));
figure('units','centimeters','Position',[0 0 width height],'
   name',figname);
plot(x,y,'*-')
    xlabel('x_D')
    ylabel('total relative mobility ({\lambda}_{rt})')
    xlim([0 1])
ylabel(strcat('{\lambda}_{rt} of the shock'))
xlabel('x_D')
figtitle=strcat('mobprofDistance_n',num2str(FP.n*100),'_char',
   num2str(1),'_','total');
print(figtitle,'-depsc');
end
```

A.2.16. structureStartEnd.m

```
function [S,slopes]=structureStartEnd(LP,FP,sw)
% structureStartEnd creates a structure (S) containing information
% (x-position, y-position, lambda_rt, fw, slopes, sw, and cross) about all
% the characteristics in all the layers and an additional slope matrix.
%
% [S,slopes]=structureStartEnd(LP,FP,sw)
% sw is the array containing the initial sw of all the characteristics.
%
% Each i value within the array within the field refers to a characteristic
% i.e. S(2).x(3) refers to the x-value of the 3rd characteristic in the
% second layer.
% S.cross stores 0 (false) or 1 (true) if a characteristic intersects the
% slope in that layer.
% S(4).cross(1)=1 means that the first characteristic intersects with the
% shock in the fourth layer.

S(1).x=zeros(1,numel(sw)); S(1).y=0;
% Makes calculations for the slopes/points

%first layer - shock - via tangency
[~, ~, lambda_rt, fw, ~] = EPNL(sw(1),FP,FP.fmdry(1),LP.end_time); % calculates the slope of the shock for the first layer using the tangency
fw_end(1,1)=fw; slopes(1,1)=(1-fw)/(1-sw(1));
S(1).lambdart(1)=lambda_rt;
S(1).fw(1)=fw;
S(1).slopes(1)=slopes(1,1);
% legendInfo{1,1} = strcat('[1] S_{w}=',num2str(sw(1),'%1.3f'),'; {\lambda}_{rt}=',sprintf('%04s', num2str(lambda_rt,'%4.0f')));

%first layer - chars - via slope
for i=2:numel(sw) % calculates characteristic slopes, lambdart, & sw for the first layer via the slope of at the point
    swj=sw(i);
    [~, ~, lambda_rt, fw, slope] = EPNL(swj,FP,FP.fmdry(1),LP.end_time);
```

```
32      fw_end(i,1)=fw; slopes(i,1)=slope;
33      %     legendInfo{i,1} = strcat('[1] S_{w}=',num2str(swj
           ,'%1.3f'),'; {\lambda}_{rt}=',sprintf('%04s', num2str(
           lambda_rt,'%4.0f')));
34      S(1).lambdart(i)=lambda_rt;
35      S(1).fw(i)=fw;
36      S(1).slopes(i)=slope;
37  end
38  S(max(LP.num_layers)).sw=[]; % pre-allocates the structure for
        speed
39  S(1).sw(:)=sw;
40
41  % other layers
42  for k=2:max(LP.num_layers) % calculates slopes, lambdart, & sw
        for the rest of the layers
43      for i=1 % slope of the shock is calculated by the tangency
44          swj=fw_to_SwModified(fw_end(i),FP,FP.fmdry(k),sw(i));
45          %          [~, ~, lambda_rt, fw, ~] = EPNL(swj,FP,FP.
               fmdry(k),LP.end_time);
46          [~, ~, lambda_rt, ~, ~] = EPNL(swj,FP,FP.fmdry(k),LP.
               end_time);
47          S(k).sw(i)=swj;
48          S(k).lambdart(i)=lambda_rt;
49          fw=S(1).fw(i);
50          S(k).fw(i)=fw;
51          fw_end(i,k)=fw;
52          slopes(i,k) = (1-fw)/(1-swj);
53          S(k).slopes(i)=slopes(i,k);
54          %          legendInfo{i,k} = strcat(legendInfo{i,k
               -1},'; [',num2str(k,'%1.0f'),'] S_{w}=',num2str(
               double(swj),'%1.3f'),'; {\lambda}_{rt}=',sprintf
               ('%04s', num2str(lambda_rt,'%4.0f')));
55      end
56      for i=2:numel(sw) % the char's slopes are determined by
           the slope at the point
57          swj=fw_to_SwModified(S(1).fw(i),FP,FP.fmdry(k),sw(i));
58          [~, ~, lambda_rt, ~, slope] = EPNL(swj,FP,FP.fmdry(k),
               LP.end_time);
59          fw_end(i,k)=fw; slopes(i,k) = slope;
60          %          legendInfo{i,k} = strcat(legendInfo{i,k
               -1},'; [',num2str(k,'%1.0f'),'] S_{w}=',num2str(
               double(swj),'%1.3f'),'; {\lambda}_{rt}=',sprintf
               ('%04s', num2str(lambda_rt,'%4.0f')));
61          S(k).sw(i)=swj;
62          fw=S(1).fw(i);
63          S(k).fw(i)=fw;
```

```
64          S(k).lambdart(i)=lambda_rt;
65          S(k).slopes(i)=slope;
66      end
67  end
68  % Fills in the structure further
69  for i=1:numel(sw) % fills in the x and y values
70      for k=1:LP.num_layers
71          S(k+1).x(i)=((LP.stops(k+1)-LP.stops(k))/slopes(i,k))+
               S(k).x(i);
72          S(k+1).y=LP.stops(k+1);
73      end
74  end
75  for i=1:numel(sw)-1 %calculates in which layers the shock and
       characteristics intersect
76      for k=1:LP.num_layers
77          [test,~,~]=Intersect(S(k).x(1),S(k+1).x(1),S(k).x(i+1)
               ,S(k+1).x(i+1),S(k).y,S(k+1).y);
78          if test==1
79              S(k).cross(i+1)=1;
80          else
81              S(k).cross(i+1)=0;
82          end
83      end
84  end
85  slopes=slopes.';
86  end
```

A.2.17. Sw_versus_radius.m

```
function [results] = Sw_versus_radius(S,t,WP,LP,CM,satgraph,FP
   )
% Sw_versus_radius plots the Sw versus r at a given time and
% creates a results matrix with useful numbers.
%
% [results] = Sw_versus_radius(S,t,WP,LP,CM)
% t is the given time between 0 and 1
% satgraph: set to 1 to graph Sw versus radius
% results returns for each characteristic:
%  - the x position when it intersects with time t
%  - the corresponding radius
%  - the Sw at the corresponding time
%  - the lambda_rt value at the corresponding time
% eliminated characteristics are denoted with a -9

test=0; k=0;
if satgraph==1
    width = 17.4;
    height = 17.4/2.8;
    figname=strcat('saturationProfile_td',num2str(t),'_n',
       num2str(FP.n*10));
    f=figure('units','centimeters','Position',[0 0 width
       height],'name',figname);

end
[~,n]=size(S);
results=ones(numel(S(1).x),4)*-9;
a=1; % starting char

while test==0 && k<n-1 && a<=LP.lin
    k=k+1;
    [test,~,p2]=Intersect(S(k).x(a),S(k+1).x(a),t,t,S(k).y,S(k
       +1).y);
    if k==n-1 && test==0
        k=0;
        a=a+1;
    end
end
if p2<=1.0 % checks to make sure that the characteristic is
   still in bounds
    results(a,1)=p2;
    results(a,2)=sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw^2);
    results(a,3)=S(k).sw(a);
    results(a,4)=S(k).lambdart(a);
    if satgraph==1
```

```
41          plot(sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw^2),S(k).sw(a),'*'
               ,'color',CM(2,:))
42          legendInfo{1} = strcat('S_{w}=',num2str(double(S(k).sw
               (a))));
43      end
44  end
45  if satgraph==1
46      hold on
47  end
48  for j=a+1:numel(S(1).x)
49      for i=k:-1:1
50          [test,~,p2]=Intersect(S(i).x(j),S(i+1).x(j),t,t,S(i).y
               ,S(i+1).y);
51          if test==1
52              if p2<results(a,1)
53                  results(j,1)=p2;
54                  results(j,2)=sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw
                       ^2);
55                  results(j,3)=S(i).sw(j);
56                  results(j,4)=S(i).lambdart(j);
57                  if satgraph==1
58                      plot(sqrt(p2*(WP.re^2-WP.rw^2)+WP.rw^2),S(
                           i).sw(j),'*','color',CM(j+1,:))
59                      legendInfo{j} = strcat('S_{w}=',num2str(
                           double(S(k).sw(j))));
60                      k=i; % optimizes the searching by limiting
                            the next cycle to the length of this
                           one
61                  end
62              end
63          end
64      end
65  end
66  if satgraph==1
67      xlabel('r (m)')
68      ylabel('S_w')
69      tightfigIC;
70      figtitle=strcat('saturationProfile_n',num2str(FP.n*100));
71      print(figtitle,'-depsc')
72  end
73  end
```

A.2.18. tightfig.m

tightfig and it's variants (e.g. tightfigIC, tightfigDimPressureGraph) are the same except for changed constants in Lines 76, 80, and 82. This function can be removed without affecting the model's functionality. tightfig is written by Richard Crozier. Researchers wanting to build off of this model as advised to use 'JacobD10/tightfigadv' by Jacob D instead. Both functions are available on Mathworks File Exchange as well as on GitHub.

```
1  function hfig = tightfig(hfig)
2  % tightfig: Alters a figure so that it has the minimum size
       necessary to
3  % enclose all axes in the figure without excess space around
       them.
4  %
5  % Note that tightfig will expand the figure to completely
       encompass all
6  % axes if necessary. If any 3D axes are present which have
       been zoomed,
7  % tightfig will produce an error, as these cannot easily be
       dealt with.
8  %
9  % hfig - handle to figure, if not supplied, the current figure
        will be used
10 % instead.
11 yes=0;
12     if nargin == 0
13         hfig = gcf;
14     end
15
16     % There can be an issue with tightfig when the user has
          been modifying
17     % the contnts manually, the code below is an attempt to
          resolve this,
18     % but it has not yet been satisfactorily fixed
19 %     origwindowstyle = get(hfig, 'WindowStyle');
20     set(hfig, 'WindowStyle', 'normal');
21
22     % 1 point is 0.3528 mm for future use
23
24     % get all the axes handles note this will also fetch
          legends and
25     % colorbars as well
26     hax = findall(hfig, 'type', 'axes');
27
28     % get the original axes units, so we can change and reset
          these again
```

```
29      % later
30      origaxunits = get(hax, 'Units');
31
32      % change the axes units to cm
33      set(hax, 'Units', 'centimeters');
34
35      % get various position parameters of the axes
36      if numel(hax) > 1
37  %         fsize = cell2mat(get(hax, 'FontSize'));
38          ti = cell2mat(get(hax,'TightInset'));
39          pos = cell2mat(get(hax, 'Position'));
40      else
41  %         fsize = get(hax, 'FontSize');
42          ti = get(hax,'TightInset');
43          pos = get(hax, 'Position');
44      end
45
46      % ensure very tiny border so outer box always appears
47      ti(ti < 0.1) = 0.15;
48
49      % we will check if any 3d axes are zoomed, to do this we
            will check if
50      % they are not being viewed in any of the 2d directions
51      views2d = [0,90; 0,0; 90,0];
52
53      for i = 1:numel(hax)
54
55          set(hax(i), 'LooseInset', ti(i,:));
56  %         set(hax(i), 'LooseInset', [0,0,0,0]);
57
58          % get the current viewing angle of the axes
59          [az,el] = view(hax(i));
60
61          % determine if the axes are zoomed
62          iszoomed = strcmp(get(hax(i), 'CameraViewAngleMode'),
                'manual');
63
64          % test if we are viewing in 2d mode or a 3d view
65          is2d = all(bsxfun(@eq, [az,el], views2d), 2);
66
67          if iszoomed &&  ~any(is2d)
68              error('TIGHTFIG:haszoomed3d', 'Cannot make figures
                    containing zoomed 3D axes tight.')
69          end
70
71      end
```

```
72
73      % we will move all the axes down and to the left by the
           amount
74      % necessary to just show the bottom and leftmost axes and
           labels etc.
75      moveleft = min(pos(:,1) - ti(:,1));
76      movedown = min(pos(:,2) - 0*ti(:,2));
77
78      % we will also alter the height and width of the figure to
            just
79      % encompass the topmost and rightmost axes and lables
80      figwidth = max(pos(:,1) + pos(:,3) + ti(:,3) - moveleft);
81
82      figheight = max(pos(:,2) + pos(:,4) + ti(:,4) - movedown);
83
84      % move all the axes
85      for i = 1:numel(hax)
86
87          set(hax(i), 'Position', [pos(i,1:2) - [moveleft,
               movedown], pos(i,3:4)]);
88
89      end
90
91      origfigunits = get(hfig, 'Units');
92
93      set(hfig, 'Units', 'centimeters');
94
95      % change the size of the figure
96      figpos = get(hfig, 'Position');
97
98      set(hfig, 'Position', [figpos(1), figpos(2), figwidth,
           figheight]);
99
100     % change the size of the paper
101     set(hfig, 'PaperUnits','centimeters');
102     set(hfig, 'PaperSize', [figwidth, figheight]);
103     set(hfig, 'PaperPositionMode', 'manual');
104     set(hfig, 'PaperPosition',[0 0 figwidth figheight]);
105
106     % reset to original units for axes and figure
107     if ~iscell(origaxunits)
108         origaxunits = {origaxunits};
109     end
110
111     for i = 1:numel(hax)
112         set(hax(i), 'Units', origaxunits{i});
```

```
113     end
114
115     set(hfig, 'Units', origfigunits);
116
117 %       set(hfig, 'WindowStyle', origwindowstyle);
118
119 end
```