FACULTY OF SCIENCE AND TECHNOLOGY

MASTER THESIS

Study programme / specialisation: Petroleum Engineering / Drilling and Well Engineering	The spring semester, 2022 Open / ~~Confidential~~
Author: Abdelgadir Mohammed, Mohammed Elsiddig	……………………………………………… (signature author)
Course coordinator: Anita Malde Rasmussen Supervisor(s): Fjelde, Kjell Kåre	
Thesis title: Simulation Study and Modelling Improvements for the Bullheading Process	
Credits (ECTS): 30	
Keywords: Bullheading Bully Flow, Slug Flow, Suspended Gas Gas Slip Velocity Critical Bullheading Rate Drift Flux Model AUSMV scheme	Pages: 99 + appendix: 52 Stavanger, May 28th, 2022

Acknowledgement

I would like to express my sincere gratitude to my supervisor Professor Fjelde, Kjell Kåre for his time, support, and guidance throughout this project. His knowledge and experience in the field made working on this study very enjoyable.

I would also like to extend my sincere thanks to Rein Bootsma and Robert te Gussinklo Ohmann, the directors of my current company: Spidron Weco B.V. - Well Engineers and Consultants - for supporting me during this master study. I am particularly grateful for the time off I was offered to prepare for exams and to complete the writing of this thesis.

Special thanks to my friend Muhammed Elsheikh for hosting me in his house during my stays in Stavanger. Muhammed also gave me great tips about the course since he is one of the school graduates.

I would like to acknowledge my work colleague, Marieke Bootsma, for her encouragement during this study.

Lastly, I would be remiss in not mentioning my family. Their continuous support and belief in me have kept my spirits and motivation high during my study.

Abstract

In a bullheading operation, selecting the right pumping rate is vital to the job's success. A too low pumping rate may not displace the undesired fluid back into the formation, and a too-high rate generates high surface pressures that challenge the well equipment and the exposed formation. The optimum liquid rate for bullheading gas with liquid depends on the gas slip velocity and the gas volume fraction at the front of the gas bubble; both vary with time and depth during bullheading. Therefore, a transient model is needed to predict the flow properties and pressures at any time and depth in the well to select the pumping rate that safely injects the gas into the formation.

This study aimed to numerically simulate a bullheading process using a fully transient flow model based on the drift flux model. The first part of the study replicated the simulation of a small-scale experiment to improve its results to match the experimental results. The improvement was tested by making the kick more concentrated and reducing the numerical diffusion by increasing the number of cells in the simulation. The second part simulated a hypothetical well to study the effect of kick sizes and gas suspension in non-Newtonian fluids on bullheading. Two kick sizes were selected so that one was migrating with a bubbly flow velocity and the other was migrating with a slug flow velocity; the suspension was modeled by assuming a 10% suspended gas. In simulating the small-scale experiment, a single gas slip model was used; meanwhile, a complete gas slip model that considers bubble, slug, and suspension of gas was implemented in the well simulation.

No improvement was seen in the simulation results of the experiment. Hence the large discrepancy between the experiment and the simulation remains. However, it was observed that the critical bullheading suggested by the experiment was much higher than what one would expect for the setup of the experiment, suggesting that the reported pumping rates were probably higher than real. Therefore, this study recommends redoing the experiment and remeasuring the bullheading rates. The second part's results showed that it was easier to bullhead smaller kick sizes and kicks with suspended gas. In addition, the bullheading process was completed in a shorter time, and the surface pressure was much lower when bullheading a small kick and suspended gas kick than when bullheading a large kick with no suspended gas.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

MASP	*Maximum Allowable Surface Pressure (bar)*
KMW	*Kill Mud Weight (s.g.)*
MW	*Mud Weight in use (s.g.)*
FP	*Formation Pressure (bar)*
TVD	*True Vertical Depth (m)*
SIDPP	*Shut In Drill Pipe Pressure (bar)*
Ph	*Hydrostatic Pressure (bar)*
RWP	*Rating working pressure*
LSU	*Louisiana State University*
ppf	*Pound per foot*
AUSMV	*Advection upwind splitting method*
CFL	*Courant Friedrichs Lewy*
MPD	*Managed pressure drilling*
l/s	*Litre per second*
Qliq	*Critical liquid rate*
nobox	*Number of boxes in the simulation*
s	*Seconds*
m	*Meter*
kg	*Kilogram*
lpm, l/min	*Litre per minute*
l/s	*Litre per second*

List of Symbols

Pb	*Bottomhole pressure*
$\mathrm{v_d}$	*Displacement velocity (m/s)*
K	*Flow distribution coefficient for the slip model*
f	*Friction factor*
Fw	*Friction gradient (Pa/m)*
ρ_g	*Gas phase density (kg/m3)*
$\mathrm{v_g}$	*Gas phase velocity (m/s)*
μ_g	*Gas phase viscosity (Pa.s)*
α_g	*Gas phase volume fraction (0 – 1)*
S	*Gas Slip velocity (m/s)*
g	*Gravitational acceleration due to gravity (9.81 m/s2)*
R	*Ideal gas parameter for air = 286.9 J/Kg.*
θ	*Inclination angle from vertical (degrees)*
$\mathrm{d_o}$	*Inside diameter of the outer pipe/hole (m)*
ρ_l	*Liquid phase density (kg/m3)*
$\mathrm{v_l}$	*Liquid phase velocity (m/s)*
μ_l	*Liquid phase viscosity (Pa.s)*
α_l	*Liquid phase volume fraction (0 – 1)*
$Maxs$	*Maximum suspension limit (assumed 12% in this study)*
$Mins$	*Minimum suspension limit (assumed 10% in this study)*
ρ_mix	*Mixture density (kg/m3)*
$\mathrm{v_{mix}}$	*Mixture velocity (m/s)*
μ_mix	*Mixture viscosity (Pa.s)*
$\mathrm{d_i}$	*Outside diameter of the inner pipe (m)*
D	*Pipe inside diameter*
z	*Position along the flow path (m)*
ρ_0	*Reference density of water (1000 kg/m3)*
$\mathrm{P_0}$	*Reference pressure (100000 Pa)*
$\mathrm{T_o}$	*Reference temperature (293.15 K)*
Re	*Reynolds number*
KB, KB	*Slip Model Parameters in pure bubble flow*

Ks, Ss	*Slip Model Parameters in pure slug flow*
ω	*Sound propagation velocity*
a_g	*Speed of sound in gas (316 m/s)*
a_l	*Speed of sound in water (1500 m/s)*
σ	*Surface tension between the liquid phase and the gas phase*
P	*System pressure (Pa)*
T	*System temperature (K)*
λ_2	*The gas mass transport*
β	*The reciprocal of water compressibility factor (2.2 x 109 Pa)*
t	*Time (s)*
α	*Water thermal expansion factor = 207x10-6 K-1.*
P_{wh}	*Wellhead pressure*

This page was intentionally left blank

1 Introduction

The primary objective of any well operations: drilling, completion, or workover/intervention is to complete the job tasks safely. This can be achieved by maintaining pressure control or well control during the operations. Pressure control is maintained on live wells using pressure control equipment; meanwhile, overbalanced fluid maintains well control on dead wells.

Bullheading is a well control method in which a kill liquid is pumped down the well to inject any undesired fluids back into the formation and regain well control by an overbalanced hydrostatic column. It is usually used to kill producing wells before workover operations, but it can also be an alternative kill method during drilling where circulation methods cannot be implemented, or circulation is not desired.

But the formation fluids are less dense (particularly gases), and due to buoyancy, they tend to migrate up the well while the kill fluid is being pumped. Therefore, to prevent this tendency, one needs to bullhead with a pump rate higher than the migration rate. The higher the pumping rate is, the faster the displacement process. However, higher pumping rates generate higher surface pressures that can exceed the well's surface equipment, tubular strength, or the exposed formation's maximum allowable surface pressure (MASP).

Consequently, the bullheading rate needs to be carefully designed. Selecting a too low bullheading rate may not displace the formation fluid down the well and back into the formation, but rather the fluid will bypass the kill fluid and migrate to the top of the well. On the other hand, selecting a too high bullheading rate challenges both the well equipment and the exposed formation, and depending on their strength, one or both might yield. Another reason that emphasizes the importance of the pumping rate is that it is the only parameter that, besides the kill fluid density, we have control over during bullheading.

For bullheading gas bubbles, the selected pumping rate needs to exceed a minimum critical bullheading rate that depends on the bubble front velocity, a function of the flow pattern (bubble, slug, etc.). To identify the flow pattern, one needs to predict the gas volume fraction at the front, which varies with time and location in the well during the bubble migration. Moreover, as reported in many experiments, large gas kicks start migrating up the well as large Taylor bubbles, but they leave trails of small bubbles as they migrate [1]. These small bubbles accumulate with time, reducing gas volume fraction in the bulk. If the well fluid is non-Newtonian, some bubbles will be trapped in suspension. Depending on how far the kick has to migrate and/or the severity of the suspension, one can expect a flow pattern transition to a bubble flow regime or fully suspended gas.

As a result, selecting the optimum bullheading rate is not straightforward as it depends on several parameters that vary with time and location during bullheading. This situation requires a transient model to predict the flow parameters (gas volume fraction, phase velocities, and pressures) at any time and depth during bullheading operation. With a good transient model, one can simulate various bullheading rates and select the one that leads to a successful operation. In addition, one can also study the effect of various parameters, e.g., suspension, on the bullheading process. The simulation can also be valuable for predicting the surface pressures, the duration of bullheading, and the volume of liquid required for job completion.

In this study, we shall use a fully transient two-phase flow model based on the drift flux model to simulate a bullheading process of air kick with water.

The study is divided into two main parts: the first part re-simulates a small-scale bullheading experiment that was conducted at the University of Stavanger in 2020. Besides the experiment, which was part of a master thesis, a simulation was built, and the results were compared [2]. Unfortunately, the results were significantly different. Therefore, we will replicate the simulation and attempt to improve the results for better matching. The second part of this thesis will implement a complete gas slip model that considers bubble flow and slug flow as well as suspension effect in non-Newtonian fluids. We will use this advanced model to study the effect of kick sizes and the effect of suspension on bullheading.

The structure of the thesis is as follows:

- Chapter 2: introduces the bullheading technique and its application during drilling and well intervention/production. Then it conducts literature reviews on the reported experiments and field tests on bullheading.
- Chapter 3: introduces the Transient Flow Model that will be used in this study. Here we describe the model formulation and its closure laws, focusing more on

the gas slip model and its implementation procedure. The second part of the chapter introduces the numerical scheme used in the solution of the model.

- Chapter 4: is a re-simulation of the small-scale bullheading experiment. The chapter starts by describing the experiment setup and the measurement procedure. Then it describes the simulation procedure and the main models and inputs used in the simulation. The results will be shown, and the steps to improve them will be discussed. The last section of the chapter discusses the results of the simulation and gives recommendations
- Chapter 5: represents a simulation of bullheading on a hypothetical well using the complete gas slip model. The model is then used to study the effect of kick size and suspension on bullheading, and the results will be discussed in the last part of the chapter.
- Chapter 6: concludes the study and gives recommendations for future work.

2 Literature Review

2.1 Summary

This chapter gives a brief introduction to the bullheading technique and its applications. Then we conduct a literature review on the reported experiments and studies.

2.2 Introduction to Bullheading

Bullheading is a kill method in which a heavy fluid is pumped down the well to inject any undesired fluids into the formation. It is the most applied technique in killing wells before intervention or workover operations [3]. It can also be an alternative well control method to the circulation methods [4] when circulation is not possible or desirable. Other applications of bullheading include acid treatment and hydrate removal, but they are out of the scope of this study. Instead, this study focuses on bullheading gas with water in a vertical flow orientation that can occur during drilling or when killing live wells before intervention/workover.

A successful bullheading operation requires certain conditions: firstly, the injection rate needs to overcome the migration rate of the gas up the well [3]. Secondly, the formation needs to have a good permeability to permit injection [3]. Finally, the bottomhole pressure must exceed the formation pressure and provide the required overbalance for the desired injection rate. The bottomhole pressure, however, should not fracture the formation unintentionally.

2.3 Bullheading During Drilling

During conventional drilling, the drilling fluid column is the primary barrier against any potential flow of the formation fluids. Therefore, it is designed to provide an overbalance of about 300 psi over the formation pressure. If this overbalance is lost, the formation fluids (gas or liquid) can enter the well, causing a kick. It is then necessary to shut in the well and apply a well control method to remove the influx and regain a controlled well before the operation continues. The Driller’s and Wait-and-Weight Methods are the most applied well control methods during drilling and are considered the safest and most efficient [5]. However, there are situations where using these circulation methods is not possible or unfavorable. In such cases, the bullheading method can be applied to force the kick back into the formation where it was initiated. If the well status is underbalanced, a kill mud is used to push the kick and the wellbore fluid into the formation [6]. The situations where bullheading is used as a well control method during drilling include [4]:

- Unable to establish circulation due to plugged drillstring or no drillstring in the hole
- Hazardous kicks such as H_2S
- The kick volume is significant, and handling at the surface is difficult
- There is an underground blowout

Pumping can be performed through the drillstring, annulus, or both [7]. Figure 2-1 (taken from [7]) shows an example of a bullheading kill during drilling.

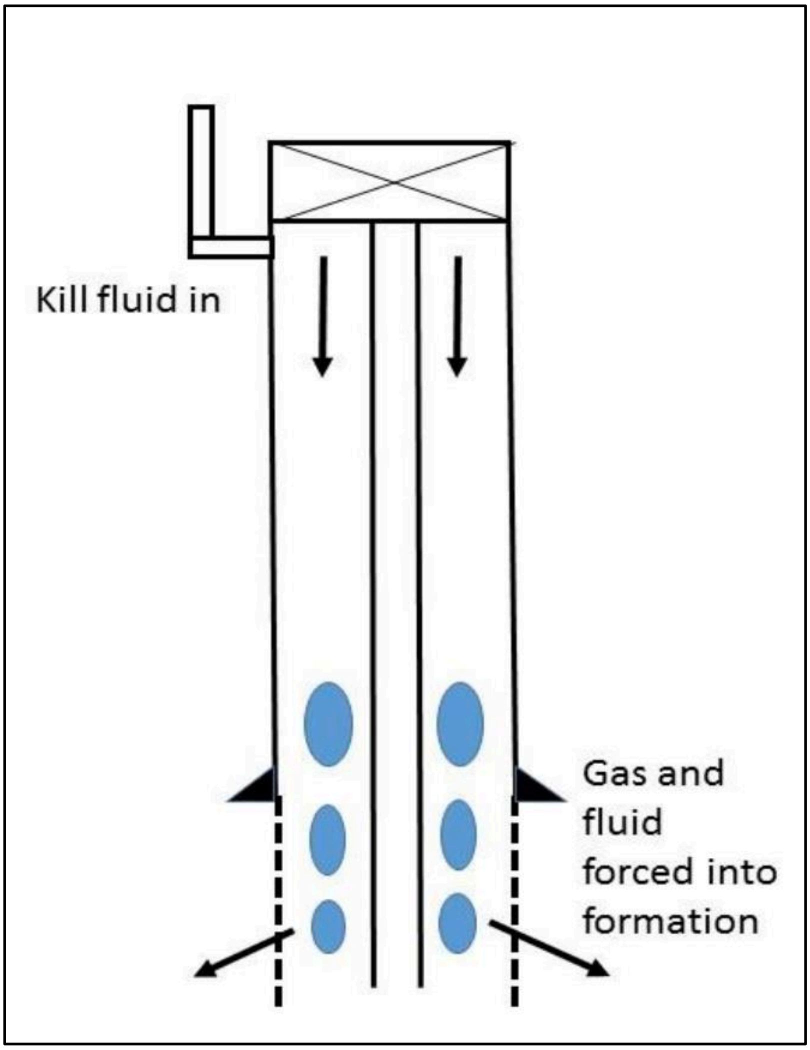


Figure 2-1: Example of Bullheading During Drilling Taken from [7].

2.4 Bullheading before Intervention or Workover

Producing wells, in many cases, are killed before carrying out an intervention or workover. Both circulation and non-circulation killing techniques are used for this purpose. The circulation techniques are better as they are performed under low pressures, and the risk of damaging the formation is minimal [3]. Circulation killing method such as a reverse circulation is used where the kill fluid is pumped through the annulus, and returns are received through the tubing. If the circulation is not achievable, the bullheading method can be applied. "*Bullheading is the most widely used kill method when preparing a well for a workover or intervention*" Howard Crumpton [3]. This is because it is simple, easy to use, and the well is controlled without handling kick fluids at the surface [3]. Here the objective is to push the produced fluids back into the reservoir through the perforation interval. As a result, the bottomhole pressure is high and above the reservoir pressure during bullheading, which may lead to the fracturing of the formation. Injecting the fluids back into the formation can also lead to formation damage. Despite these costly risks, bullheading is particularly favored over the circulation methods in the following situations [3]:

- Circulation is not achievable, e.g., due to the well configuration
- The influx contains deadly gas such as H2S
- When there is no tubing in the well
- A well with good formation permeability and the formation not easily damaged
- Before abandonment with no concerns about formation damage

Figure 2-2 (taken from [3]) shows an example of a bullheading kill for a production well.

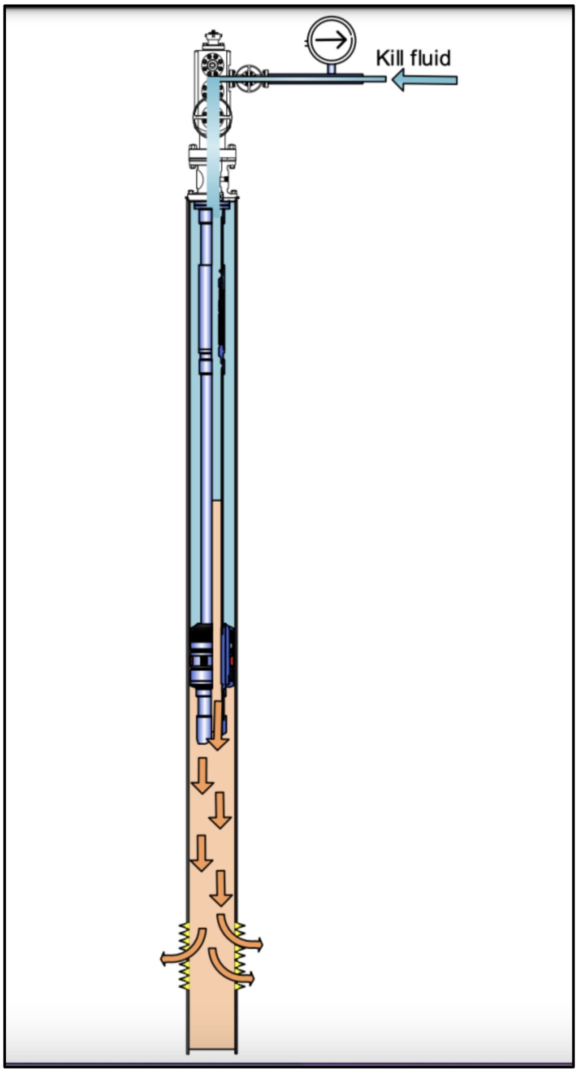


Figure 2-2: Example of Bullheading Kill on a Producing Well Taken from [3]

2.5 Important Parameters for a Successful Bullheading

During bullheading, we only have control over the kill fluid density and the pumping rate [6]. Therefore, they need to be carefully designed to ensure a successful and safe operation. By successful, we mean the influx is pushed back into the formation, and by safe, we mean the pressures during bullheading do not exceed the formation fracture pressure or the well equipment ratings. In the following, we will discuss the main parameters for bullheading operation.

2.5.1 Kill Fluid Density, KMW

The kill fluid density is calculated based on the formation pressure and the depth where the formation started to flow. Its value should provide a static overbalance against the formation pressure at the depth of interest to prevent the future flow of any formation fluids. Equations 1 and 2 calculate the minimum kill mud weight (KMW) that balances the formation pressure (FP). The value of KMW should be rounded up number.

$$KMW = \frac{FP}{0.0981 * TVD} \qquad 1$$

$$FP = SIDPP + 0.0981 * MW * TVD \qquad 2$$

Where: KMW – Kill Mud Weight (s.g.); FP – Formation Pressure (bar); TVD– True Vertical Depth (m); MW – Mud Weight (s.g.); SIDPP - Shut-In Drillpipe Pressure (bar). An overbalance pressure (200 – 300 psi [3] for most workover operations) can be included as a safety margin.

2.5.2 The Pumping Rate

When we pump the kill fluid down the well, the light hydrocarbon migrates upward due to buoyancy. For the bullheading to succeed, the pumping rate needs to exceed this upward migration rate to push the hydrocarbon back down the well and into the formation [3]. Therefore, there is a lower limit of the successful pumping rate called the critical bullheading rate. Also, the pumping rate should not result in pump pressure that exceeds the MASP of the exposed formation or the well equipment ratings. "*In practical terms, the pump rate will be as high as possible while remaining below the fracture pressure (with a safety factor), or the mechanical limit.*" Howard Crumpton [3].

2.5.3 Pump Pressure

During bullheading, the pump pressure should not exceed the maximum allowable surface pressure (MASP) of the exposed formation or the well's surface equipment ratings. The MASP value for the formation is determined based on its strength and the mud weight in use from equation 3:

$$\mathrm{MASP} = \mathrm{P}_{\mathrm{fracture}} - P_h \qquad 3$$

$$P_h = 0.0981 * MW * TVD \qquad 4$$

Where: MW – mud weight in use, at the beginning of bullheading, MW is the well fluid density, and at the end, it is the kill fluid density calculated by equation 1. Again one can subtract any safety margin of 100 to 500 psi depending on the company policy [3]:

Since the bullheading process challenges all the well equipment, one needs to evaluate the MASP value for the burst strength of tubing/casing, the wellhead, and Christmas tree rating working pressures. The MASP should then be the minimum of the following:

- Wellhead or Christmas tree rating working pressure (RWP)
- The MASP for the Formation fracture pressure as calculated from equation 3
- 80% of casing/tubing burst strength: compared directly if the worst burst load occurs at the surface. Otherwise, one needs to calculate the MASP based on the worst-case burst load.

2.6 Literature Review on Bullheading Process

Adam T. Bourgoyne, Jr et al. [4] performed experiments on bullheading operations on a gas storage well at the LSU Petroleum Engineering Research and Technology. They used natural gas as the gas phase and water and low viscosity muds as the liquid fluids that bullheaded the gas. A total of 12 experimental tests were run; seven of them used water, and five used low viscosity mud. For all the tests, the results showed that the removal efficiency of bullheading increased linearly as the pumping rate increased, the removal efficiency was higher in the mud than in water, and up to 100% gas removal efficiency was reported when the liquid velocity was above 0.35 ft/s for muds and above 0.7 ft/s for water. For all the cases, a liquid rate greater than 1 ft/s removed the gas completely from the well. The initial height of the gas was found to have a minimal effect on the removal efficiency [4].

Oudeman [8] carried out a field test on a near-vertical gas well to investigate the feasibility of bullheading in a gas field. The tested well was completed with 7 5/8" mono-bore, perforated over 10m only, and has a permeability height product of 75 mD.m. Ahead of bullheading, 5m3 of viscous pill was pumped, followed by KCl brine (1.05 s.g) at a pumping rate of 2000 l/minin. The bottomhole pressure was measured continuously using a real-time gauge above the tubing shoe. Pumping stopped when the bottomhole pressure reached 450 bar (the formation fracture pressure). Initially, the wellhead pressure increased slightly from 268bar to 279bar and then decreased after about 15 minutes, reaching 242 bar at the end of pumping. Immediately after the pump stopped, the wellhead and bottomhole pressures decreased rapidly to less than 124 bar and reservoir pressure, respectively. The measurements showed that the pressure difference between the wellhead and the bottomhole pressure increased linearly during pumping and was constant during the close-in period after the pump stopped; the author [8] concluded that the liquid column did not disintegrate because of the yield strength of the viscous pill. The displacement velocity calculated from $d(P_{bh} - P_{wh})/\mathrm{dt} = v_d(\rho_l - \rho_g)g$ was compared with the pumping velocity of 1.45m/s and showed a good agreement during pumping period indicating a piston-like displacement. This showed that there was in practice no slip between the viscous pill and the gas. Then the well was produced, but the gas was seen at the surface after 25m3 of liquid production, indicating the gas was trapped. The main recommendation was that using a high viscosity pill ahead of the kill brine suppressed gas migration in the liquid column.

Oudeman et al. [9] developed a numerical simulator to predict the pressure and flow during bullheading. The purpose of the study was to optimize the selection of pump rate, required horsepower, density, and fluid type for bullheading operation [9]. The model used was based on the drift flux model, and the slip velocity was calculated from the wellhead pressure decline rate and the phase densities. They applied the simulator to field test data from a bullheading test on a live gas well [9]. The tested well was a deviated gas well completed with a 7” 32ppf and brine of 1050 kg/m^3. The test concluded that a pump rate of 2.0 m^3/min was able to kill the well, although the 2.0 m^3/min was not achieved at the start of pumping because of the pressure limits of the kill line.

3 Transient Flow Model

Modeling of two-phase flow can be performed using a two-fluid type model or a Drift Flux type model. The two-fluid model considers each phase separately. Here we end up with one momentum equation for each phase, which presents mathematical complications and uncertainties in defining the interfacial tensions between the phases [10]. The Drift Flux Model is a much simpler two-phase flow model that considers the mixture as a whole rather than separate phases [11]. The Drift Flux Model is simple and applicable to many two-phase flow systems in petroleum engineering, including well control. This chapter presents the one-dimensional Drift Flux Model for unsteady compressible two-phase flow used in the simulation of the bullheading process in this study. Firstly, the model is introduced, and then the hydrodynamic and thermodynamic closure laws are provided. Here the focus will be on the slip model. After that, the model properties are discussed, and a numerical solver based on the AUSMV scheme will be presented.

3.1 Drift Flux Model

The Drift Flux Model consists of two mass conservation laws, one for gas and another for liquid, and a mixture momentum equation [12]. The model formulation here assumes no mass exchange between the liquid and gas, constant temperature, and uniform flow area [7]. Therefore, the energy balance is neglected, and the two phases are assumed to be water and gas.

3.1.1 Conservations Laws

Equations 5, 6, and 7 represent the set of equations of the Drift Flux Model [13]:

Conservation of gas mass:

$$\frac{\partial}{\partial t}(\alpha_g \rho_g) + \frac{\partial}{\partial z}(\alpha_g \rho_g v_g) = 0 \qquad 5$$

Conservation of liquid mass:

$$\frac{\partial}{\partial t}(\alpha_l \rho_l) + \frac{\partial}{\partial z}(\alpha_l \rho_l v_l) = 0 \qquad 6$$

Conservation of mixture momentum:

$$\frac{\partial}{\partial t}(\alpha_g\rho_g v_g + \alpha_l\rho_l v_l) + \frac{\partial}{\partial z}(\alpha_g\rho_g v_g^2 + \alpha_l\rho_l v_l^2 + P) = -(F_w + \rho_{mix} * cos\theta) \qquad 7$$

Where: t - the time, and z- the position along the flow direction, α_g and α_l – the gas and liquid volume fractions, ρ_g and ρ_l – the gas and liquid densities, v_g and v_l – the gas and liquid velocities, P –pressure in the fluid mixture.

We have seven unknowns (α_g, α_l, ρ_g, ρ_l, v_g, $v_{l,}$ and P) and only three equations. As a result, we need to define four other closure laws to be able to solve the model.

3.1.2 Closure Laws for the Drift Flux Model

3.1.2.1 Assume a closed volume filled with liquid and gas [7]

$$\alpha_l + \alpha_g = 1 \qquad 8$$

3.1.2.2 Simplified liquid density model

$$\rho_l = \rho_0 + \frac{P - P_0}{a_l{}^2} \qquad 9$$

3.1.2.3 Simplified gas density model:

$$\rho_g = \frac{P}{a_g^2} \qquad 10$$

Where:

$a_l = 1500\frac{m}{s}$ *the speed of sound in water*;

$\rho_0 = 1000\ kg/m^3$ *reference liquid density*

$P_0 = 100{,}000\ Pa$ *reference pressure*;

$a_g = 316\ m/s$ *the speed of sound in gas*

As can be noticed, simplified liquid and gas density models are used in the Drift Flux model, where the density depends only on pressure. However, at any depth in the well, the density of the fluids varies with both temperature and pressure. For more accurate results, one can expand the models to include the temperature effect. One example of this will be shown in section 4.3.2.

3.1.2.4 Friction model

The following model, which applies to all flow patterns [12] is used. The model is presented in [12] [14].

$$Fw = \frac{2f\rho_{mix}v_{mix}abs(v_{mix})}{d_o - d_i} \quad 11$$

$$\rho_{mix} = \alpha_g\rho_g + \alpha_l\rho_l \quad 12$$

$$v_{mix} = \alpha_g v_g + \alpha_l v_l \quad 13$$

Where: f – the flow friction factor. Its value is determined from Reynold Number (R_e) for different flow regimes (Turbulent, Laminar, or Transitional).

Reynold Number (R_e) for two-phase flow depends on the mixture fluid properties (ρ_{mix}, μ_{mix}) flow geometry ($d_o - d_i$) and mixture flow properties (v_{mix}). It can be calculated as follows:

$$Re = \frac{\rho_{mix}\, abs(v_{mix})\, (d_o - d_i)}{\mu_{mix}} \quad 14$$

$$\mu_{mix} = \alpha_g\, \mu_g + \alpha_l\, \mu_l \quad 15$$

Reynolds number determines the flow regimes: Turbulent, Laminar, or Transitional. It is used to calculate the friction factor (f) as follows:

For Re ≥3000 – the flow is Turbulent, and the friction factor is calculated from the following correlation:

$$f = (0.052Re^{-0.19}) \quad 16$$

For Re ≤ 2000 – the flow is Laminar, and the friction factor is calculated from the following:

$$f = \frac{24}{Re} \quad 17$$

For 2000 < Re < 3000 – the flow is a transitional, and linear interpolation is used to calculate the friction factor:

$$f = (1 - x)\frac{24}{Re} + x\frac{0.052}{Re^{0.19}} \qquad 18$$

$$x = \frac{Re - 2000}{3000 - 2000} \qquad 19$$

3.1.3 Gas Slip Model

The Drift Flux Model uses the gas slip model derived by Zuber and Findlay [15] as follows:

$$v_g = Kv_{mix} + S \qquad 20$$

$$v_{mix} = (\alpha_l v_l + \alpha_g v_g) \qquad 21$$

Where: K – the flow distribution coefficient, S – the slip velocity of the gas relative to liquid [13].

The coefficient K is dependent on the bubble's distribution and their relative velocities across the flow area [16] [17]. Zuber and Findlay [15] suggested that the K values range between 1.0 and 1.5. Hassan et al. [18] showed that the flow model parameters K and S depend on both the flow type (Upward Cocurrent, Countercurrent, Downward) and flow regime (Bubbly, Slug, Churn, Annular).

Upward Concurrent is when the two phases flow upward, opposite the Downward type, where the two phases flow downward. The Countercurrent represents a flow of fluids in opposite directions. The focus of this study is on Bubble and Slug flow regimes. Therefore, they will be discussed in the following subsections.

3.1.3.1 Bubble flow

In a two-phase flow of liquid and gas in a vertical pipe, bubble flow is described as small gas bubbles distributed in a continuous liquid phase that fills the pipe.

In the bubble flow, the rise velocity of a single bubble depends on the density difference between the liquid phase (ρ_l) and the gas phase (ρ_g) and the surface tension (σ) between the two phases [17]. It is calculated from the Harmathy correlation [19] as in equation 22:

$$S = 1.53\left(g\sigma\frac{(\rho_l - \rho_g)}{\rho_l^2}\right)^{\frac{1}{4}} \qquad 22$$

Where: g – gravitational constant (9.81 m/s^2)

Nickens [20] used K = 1.0 for bubble flow. Hassan et al. [18] suggested K = 1.2 for Downward and Upward Cocurrent flows and K = 2.0 for Countercurrent flow.

It was suggested in several models that the transition from bubble flow to slug flow occurs at a gas volume fraction of 0.25 [21] [17] [18]. Caetano [22] suggested 0.20 for flow in annulus.

3.1.3.2 Slug flow

The gas flows as large bubbles in the slug flow known as Taylor [21]. One needs to differentiate between the shape of the Taylor bubble in a pipe and the annulus between two pipes. In pipe flow, the Taylor bubble shape is a bullet [21], and this is what was observed in the small-scale experiment that will be re-simulated in chapter 4. For slug flow in the annulus, the Taylor bubbles are wrapped around the external wall of the inner pipe [21].

The Taylor bubble rise velocity for vertical flow is obtained from equation 23, which was derived by Davies and Taylor [23]:

$$S = 0.35\left(gD\frac{(\rho_l - \rho_g)}{\rho_l}\right)^{\frac{1}{2}} \qquad 23$$

Where: D – the pipe inside diameter, g – gravitational constant (9.81 m/s^2)

Hassan et al. [18] suggested K = 1.2 for Upward Cocurrent flow and Countercurrent flow and K = 1.12 for Downward flow.

3.1.3.3 Suspended gas flow

Many studies have observed the suspension of gas bubbles in non-Newtonian fluids [1] [8] [24]. Johnson et al. [1] showed that for a gas concentration of more than 10%, the bubble would rise fast with a velocity of about 0.5m/s, leaving a trail of bubbles in suspension due to the mud yield stress. They added that the entire influx could be suspended for small gas volumes. The suspended gas moves with the liquid velocity and has no-slip velocity (S = 0).

One can observe that there is a singularity in the gas phase velocity formula. If we substitute 21 in 20 and re-arrange, we end up with the following expression for v_g:

$$v_g = \frac{\alpha_l v_l K + S}{1 - \alpha_g K} \qquad 24$$

The singularity in 24 occurs when we divide by zero or $\alpha_g = \frac{1}{K}$. For K = 1.12 and 1.2 (slug flow), $\alpha_g = 0.83\ and\ 0.89$,respectively. Also, for the single gas phase $\alpha_g = 1.0$, K = 1 and S = 0 (no slip). To avoid the problem of dividing by zero in the simulation, K is interpolated to 1.0 in the interval $\alpha_g = [0.7, 0.8]$ and S to 0.0 in the interval $\alpha_g = [0.9, 1.0]$; this will ensure a no-slip case when transitioning to the single gas flow [13].

3.1.3.4 Gas slip models used in this study

The drift flux model uses the gas slip model in equation 20 and combines it with the mixture momentum conservative variable to find the phase velocities. Now, this slip model requires two parameters to be defined, namely K and S, and depending on the accuracy and complexity of the solution, one needs to decide on the values of K and S.

The old model used in [7] used a simple slip model for simulating a bullheading process. In the simple gas slip model, the K and S are assumed constant with time and depth for one flow regime throughout the simulation process. We will use the simple model when we replicate the simulation of a small-scale bullheading experiment in chapter 4. Since a clear Taylor bubble was observed during the experiment, we will provide a fixed S value calculated from equation 23 and fixed K values as 1.20 during the gas injection and migration phase and 1.12 during the bullheading phase.

In chapter 5, we will implement a gas slip relation that considers both bubble and slug flow as well as suspension effects for Non-Newtonian Fluids. This model was used in previous work [25] to simulate gas migration and pressure build-up in a closed-in well. We will use it in this study to investigate the effect of kick size on bullheading and the effect of suspension on bullheading. We will use the gas volume fraction to identify the flow pattern and thus the slip model parameters at any time and depth during the simulation. We will assume Bubbly flow for gas volume fraction < 0.20 and Slug flow for gas volume fraction > = 0.25. we will linearly interpolate K and S values between 0.2 and < 0.25. When we include suspension, we assume a certain limit (e.g., 0.10) and take 0.02 above that limit to linearly interpolate to Bubbly flow. The complete code for the new slip model is shown in Appendix C and summarised in Table 3-1.

Table 3-1: Summary of the Slip Code in the New Model

Volume fraction	Flow Pattern	K Upward, K_B = 1.20, K_S = 1.20 Bullheading, K_B = 1.20, K_S = 1.12	S (S_B = S from Equation 22) (S_S = S from Equation 23)
> = 0.00 & <=Min suspension	Suspended gas no slip	$K = 1.0$	S = 0.0
> Min suspension & < Max suspension	Transition from Suspended to bubble flow	$xint = \frac{\alpha_g - Mins}{Maxs - Mins}$ $K = xint * K_B + (1 - xint) * 1.0$	$xint = \frac{\alpha_g - Mins}{Maxs - Mins}$ $S = xint * S_B + (1 - xint) * 0.0$
>= Max suspension & < 0.20	Bubbly Flow	K_B	S_B
> = 0.20 & < 0.25	Transition from Bubble to Slug flow	$xint = \frac{\alpha_g - 0.20}{0.25 - 0.20}$ $K = xint * K_S + (1 - xint) * K_B$	$xint = \frac{\alpha_g - 0.20}{0.25 - 0.20}$ $S = xint * S_S + (1 - xint) * S_B$
> = 0.25	Slug Flow	K_S	S_S
> = 0.70 & < = 0.80	To avoid singularity*	$xint = \frac{\alpha_g - 0.70}{0.80 - 0.70}$ $K = xint * 1.0 + (1 - xint) * K_S$	S_S
> 0.80	To avoid singularity*	$K = 1.0$	S_S
> = 0.90 & < = 1.0	To avoid singularity*	$K = 1.0$	$xint = \frac{\alpha_g - 0.90}{1.00 - 0.90}$ $S = xint * 0.0 + (1 - xint) * S_S$

* Also implemented in the old model

3.1.4 Properties of the Drift Flux Model

Examining the model described in equations 5, 6, and 7 closer, one can write it in the following general formula [13]:

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial z} = Q(U) \qquad 25$$

Where U is a vector representing the conservative variables of gas mass, liquid mass, and mixture momentum:

$$U = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} \alpha_l \rho_l \\ \alpha_g \rho_g \\ \alpha_l \rho_l v_l + \alpha_g \rho_g v_g \end{pmatrix} \qquad 26$$

F(U) is the flux vector:

$$F(U) = \begin{pmatrix} \alpha_l \rho_l v_l \\ \alpha_g \rho_g v_g \\ \alpha_l \rho_l v_l^2 + \alpha_g \rho_g v_g^2 + P \end{pmatrix} \qquad 27$$

Q(U) is the term source vector. As no mass exchange is allowed between the gas and liquid, the mass sources are assumed to be zero. The momentum source is the sum of the friction force and gravitational force:

$$Q(U) = \begin{pmatrix} 0 \\ 0 \\ -(F_w + \rho_{mix} cos\theta) \end{pmatrix} \qquad 28$$

When a numerical scheme is used to update the conservative variables, one needs to convert these to physical variables like, e.g., pressure, phase volume fraction, phase densities, and phase velocities. This is done by combining the updated conservative variables with specified closure laws.

The mass conservative variables u_1 and u_2 are used to calculate the system pressure (P) and the gas and liquid volume fractions (α_g , α_{l}). This is done by considering the two closure models for liquid and gas density specified in Equations 3.5 and 3.6. The pressure is the first physical variable that is determined analytically or numerically. Equation 29 is the analytical equation used in calculating the pressure in the AUSMV. Details of the derivation are provided in [10, 14].

$$P = \frac{a_l{}^2(-b + \sqrt{(b^2 - 4ac)}\,)}{2} \qquad 29$$

Where: $a = \frac{1}{a_l{}^2}$; $b = k_0 - u_1 - \frac{a_g{}^2}{a_l{}^2} u_2$; $c = k_0\, a_g{}^2\, u_2$; $k_0 = \rho_0 - \frac{P_0}{a_l 2}$

Once the pressure is calculated, the fluid densities are updated using equations 9 and 10. The liquid and gas densities are combined with the mass conservative variables u_1 and u_2 to find the liquid volume fraction and gas volume fraction, respectively:

$$\alpha_l = \frac{u_1}{\rho_l} \qquad 30$$

$$\alpha_g = \frac{u_2}{\rho_g} \qquad 31$$

Note that the expressions in equation 26 will change if other density models shall be used. This will be done in Chapter 4, where we consider some more realistic density models to replicate some simulations of an experimental setup.

After finding the pressure, phase densities, and phase volume fractions, one must determine the phase velocities. The mixture momentum conservative variable u_3 and the slip model in equation 20 is combined to calculate the gas and liquid phase velocities.

We first recall the mixture momentum conservative variable u_3 as:

$$u_3 = \alpha_l \rho_l v_l + \alpha_g \rho_g v_g \qquad 32$$

Then we define the following two slip relations 33 and 34. The first is the gas slip model defined in equation 20. The second is artificial for the liquid phase velocity.

$$v_g = K v_{mix} + S \qquad 33$$

$$v_l = K_l v_{mix} + S_l \qquad 34$$

One needs to calculate $K_l, S_l\ and\ v_{mix}$ to be able to calculate the phase velocities. The derivations of the terms are left out for the reader, but instead, we provide the final results. Equations 32, 33 and 34 are combined to obtain K_l, S_l and v_{mix}:

$$K_l = \frac{1 - \alpha_g K}{\alpha_l} \quad 35$$

$$S_l = \frac{-\alpha_g S}{\alpha_l} \quad 36$$

$$v_{mix} = \frac{(u_3 - b)}{a} \quad 37$$

Where: $a = \rho_l \alpha_l K_l + \rho_g \alpha_g K$; $b = \rho_l \alpha_l S_l + \rho_g \alpha_g S$

These are the relations implemented in the MATLAB codes shown in Appendices, where the AUSMV scheme is used as the numerical solver for the Drift-Flux Model.

The system of equation 25 is a set of nonlinear partial differential equations of a hyperbolic type that describes wave propagation [26]. This system was analyzed in-depth mathematically in [27]. If the liquid is assumed incompressible and the gas density fraction is much lower than the liquid density fraction ($\alpha_g \rho_g \ll \alpha_l \rho_l$), the sound propagation velocity (ω) for the two-phase region can be obtained [14] [27]:

$$\omega = \sqrt{\frac{P}{\alpha_g \rho_l (1 - \alpha_g)}} \quad 38$$

And the eigenvalues of the hyperbolic system [26] [27] are:

$$\lambda_1 = v_l - \omega, \quad \lambda_2 = v_g, \quad \lambda_3 = v_l + \omega \quad 39$$

Here λ_1 and λ_3 represent the pressure pulses that travel upstream and downstream due to the pump rate changes or changes in valves that the flow is passing. These pulses are small and decay with time, as discussed further in section 4.6.2. They will be dampened by the friction term in the momentum equation. λ_2 represents the gas mass transport in the well [14].

3.2 Numerical Scheme

3.2.1 Discretization

The 1D Advection Upstream Splitting Method Hybrid Scheme (AUSMV) is used to solve the Drift Flux Model described in 3.1. However, details of the AUSMV are out of the scope of this study, and the reader is referred to [10, 14, 26].

The well is first discretized into N cells of equal height Δz as shown in Figure 3-1. Next, the cells are filled in with initial values of the physical variables (α_g, α_l, ρ_g, ρ_l, v_g, v_l, and P). Next, the conservative variables (u_1, u_2, u_3) are evaluated for each cell as shown in 3.1.4. The physical and conservative variables are considered constant in the mid of each cell [13]. Next, the friction and gravitational terms are evaluated using the physical variables. After that, the fluxes entering and leaving each cell ($F^n_{j+\frac{1}{2}}$ and $F^n_{j-\frac{1}{2}}$) are established as described in [14]. As a result, we have the physical variables, the conservative variables, the momentum source (Q), and the fluxes evaluated for each cell (j) at the initial time level. The conservation laws will then be used to update the conditions in the well for the new time level.

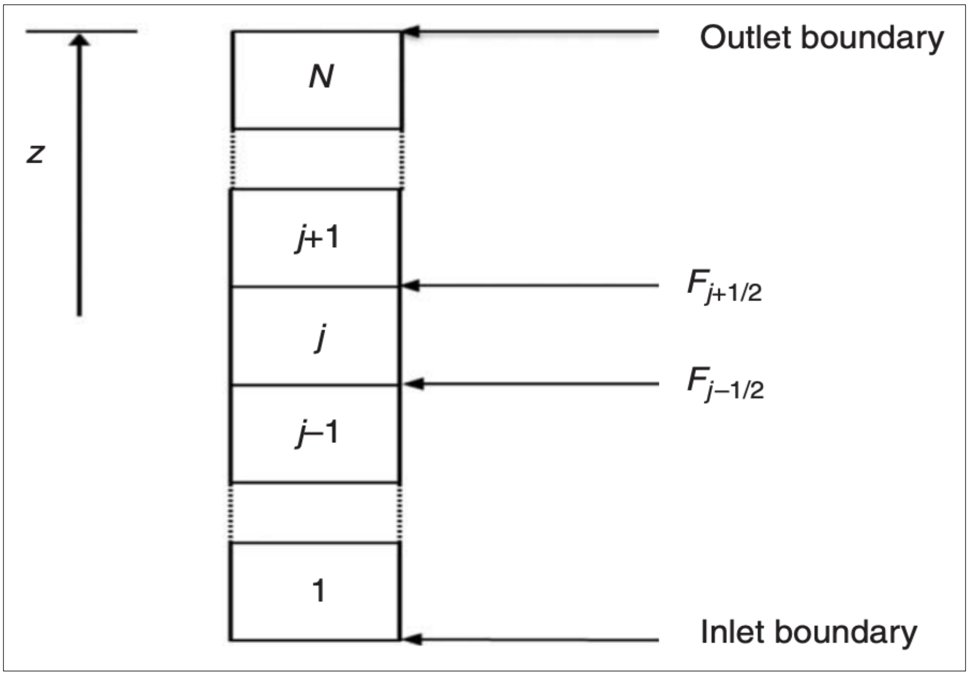


Figure 3-1: Discretization of the Well, Taken from [14].

The explicit solution starts by updating the conservative variables (u_1, u_2, u_3) using equation 40. Then, all the variables on the right-hand side are evaluated at the old-time level (n). Figure 3-2 demonstrates the idea.

$$U_j^{\ n+1} = U_j^{\ n} - \frac{\Delta t}{\Delta z}\left(F^n_{j+\frac{1}{2}} - F^n_{j-\frac{1}{2}}\right) + \Delta t Q_j^{\ n} \qquad 40$$

Where: U-the conservative variables, j- the cell number, n – old-time level, (n+1) – new time level, Δt – the time interval between (n) and (n+1), $F^n_{j+\frac{1}{2}}$ and $F^n_{j-\frac{1}{2}}$ - fluxes out and in of the cell evaluated at the old time. The formulae for the thesis for the AUSMV scheme can be found in [14].

The solution in equation 40 gives new values for all cells' conservative variables u1, u2 and u3. The conservative variables u_1, u_2 at new time levels are combined with the density models to find the pressure, phase volume fractions and densities. The conservative variables u_3 is combined with the gas slip model in equation 20 to find the phase velocities. After that, the friction and gravitation terms and the fluxes are evaluated at new time levels. Then the old values are set equal to the new values, and the calculation process is repeated for the next time step. This continues until the specified end time is reached.

Since the solution uses fluxes at the old-time level, it is explicit, and therefore it has a stability limit in terms of the Courant-Friedrichs-Lewy (CFL) criteria equation 41.

$$\Delta t \leq CFL \frac{\Delta z}{max\left(|\lambda 1|, |\lambda 2|, |\lambda 3| \right)} \tag{41}$$

The sonic wave velocity is maximum in water (the liquid phase), which is 1500 m/s. The simulation was tested for various CFL values and found stable for CFL values less than 0.25. In this study, the CFL was maintained at around 0.2, and the values of Δt and Δz were adjusted accordingly.

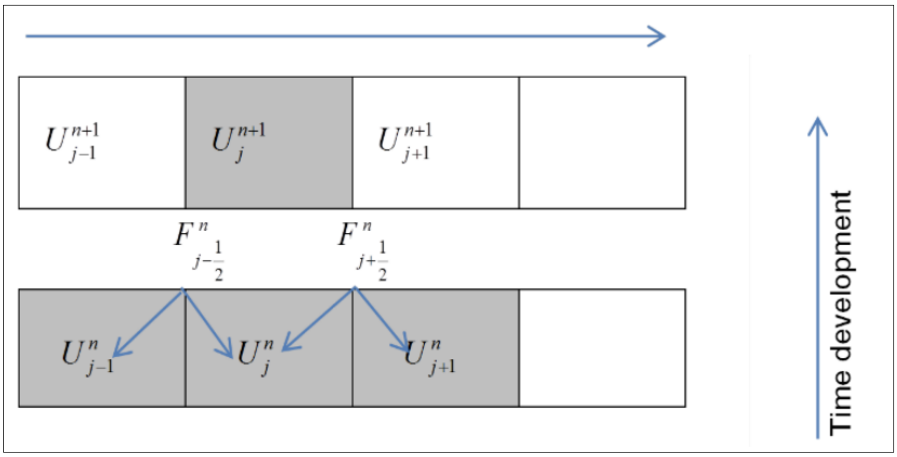


Figure 3-2: Updating the Conservative Variables, Taken from [13]

3.2.2 Initial Conditions

The initial condition for the bullheading simulations in this study is a well filled with liquid at static condition. This is the situation before the kick is taken into the well. The well is static, so only the hydrostatic term is considered. The densities are calculated from the pressure. The phase velocities and the gas volume fraction are considered zero. The liquid volume fraction equals unity.

3.2.3 Boundary Treatment

The boundary at the well inlet and outlet are treated differently. The flux values (mass, momentum, and pressure) are found by extrapolating the mid values of the first and last cells in combination with the physical given boundary conditions. There are generally two boundary conditions considered: closed and open well [13].

3.2.3.1 Open well

The outlet of the well: at the outlet of the well, the pressure is either atmospheric pressure or a given choke pressure used, e.g., in a managed pressure drilling operation (MPD). The mass and momentum fluxes are found by extrapolating the variables from the mid of the top cell.

Inlet of the well: The inlet mass flow rates for both liquid and gas are specified directly, and these are used to derive the mass and momentum fluxes at the inlet. The inlet pressure is found from the pressure at the mid of the bottom cell P (1), and the cell variables and geometry are as follows:

$$P_{inlet} = P(1) + \frac{\Delta z}{2} \rho_{mix}\, g\, cos\theta + \frac{\Delta z}{2} F_w \qquad 42$$

3.2.3.2 Closed well

The outlet of the well: the mass and momentum fluxes are considered zero at the outlet. The outlet pressure is determined from 43

Inlet of the well: the mass and momentum fluxes are considered zero at the inlet. The inlet pressure is determined from 44

$$P_{outlet} = P(N) - \frac{\Delta z}{2} \rho_{mix}\, g\, cos\theta - \frac{\Delta z}{2} F_w \qquad 43$$

$$P_{inlet} = P(1) + \frac{\Delta z}{2} \rho_{mix}\, g\, cos\theta + \frac{\Delta z}{2} F_w \qquad 44$$

3.2.3.3 Bullheading process

The outlet of the well: the mass and momentum fluxes at the outlet (top of the well) are found from the liquid pumping rate. The outlet pressure is calculated from 45. Here one can note that the bullheading rate will be specified as negative since a positive direction is chosen upwards.

Inlet of the well: the inlet mass and momentum fluxes are determined by extrapolating the bottom cell values. The inlet pressure is fixed at a value usually equivalent to the reservoir pressure plus the required overbalance to enforce the kick into formation. One assumes that the formation can take all the fluids for this pressure.

$$P_{outlet} = P(N) - \frac{\Delta z}{2}\, \rho_{mix}\, g\, cos\theta - \frac{\Delta z}{2} F_w \qquad 45$$

3.2.4 Second-Order AUSMV Scheme

The scheme described in equation 40 is of first-order, which means its numerical solution contains numerical diffusion that removes any sharp transition zones [7] [13]. Therefore, to preserve the sharp transition zones, one must reduce the numerical diffusion to the lowest practical level. For example, the numerical diffusion can be reduced by making the discretization finer (increasing the number of cells).

Another approach is to apply slope limiters to make the numerical scheme of second order. In the first-order scheme, the physical variables were considered constant in the mid of each cell, and their values were used to calculate the fluxes between the cells. In the second-order scheme, the concept is to use a slope to evaluate the boundary values used to calculate the fluxes in and out of each cell [13]. Details of the use of slope limiters in multi-phase flow modeling can be found in [28]. The Minmod limiter is used, and its concept is shown in Figure 3-3 and Figure 3-4

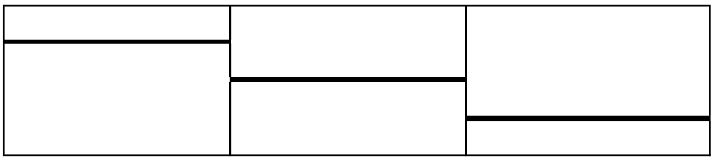


Figure 3-3: First Order Method, Taken from [7] [13].

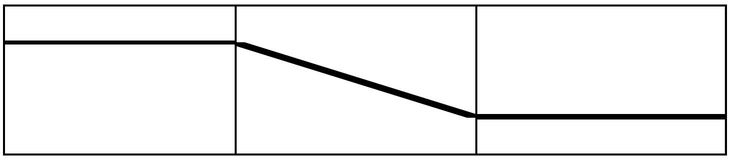


Figure 3-4: Second Order With Slope Limiter, Taken from [7] [13].

4 Re-Simulation of Bullheading Experiment in Small Scale Loop

4.1 Summary

This chapter aims to re-simulate a bullheading experiment conducted at the University of Stavanger in 2020 [2]. The chapter starts by giving a brief background about the experiment's set-up and its main conclusions; after that, the simulation performed as part of the experiment study will be replicated. Here, we will show the results of the new simulations and compare them to the previous results. We want to re-check the simulations because there was a rather large discrepancy between the experimental results and the simulations regarding how large a rate was required to push the gas bubble downwards. In the simulations, a much lower rate was required than what was seen in the experiments. The critical bullheading rate is defined as the minimum rate required to be able to push the gas bubble downwards. We will try to improve the simulation results by using two techniques: reducing the numerical diffusion and using a more concentrated kick. The question is then if we will get more agreement with the experimental results or if the simulation results will remain the same. The critical bullheading rate will also be estimated from the slip model and tubing area. It will also be determined from the simulations to check that the analytical formula and the simulations are in accordance with each other. The chapter closes by discussing the results and giving recommendations.

4.2 Background of the Experiment

As part of their master’s study, Patino Oscar and Alfonso Manuel [2] carried out a small-scale experimental investigation of the bullheading process at the University of Stavanger in 2020. This Master’s thesis project aimed to simulate a kick injected as air from the bottom of a tube filled with water, allowing it to migrate upwards for some time before pushing it downwards by pumping liquid water from the top. Besides the experiment, a numerical simulation model was built using the Advection Upstream Splitting Method (AUSMV) scheme to solve the Transient Drift Flux Model for the two-phase flow of the small-scale experiment.

A specific amount of air was injected through an inlet valve at the bottom of a tube filled with water in the experiment. This air was allowed to displace the water and gather below an injection valve. Then the injection valve was opened, and the air migrated into the water column of the closed-in tube. After that, the pump was turned on and started to pump water down the tube to push the gas bubble out of the tube through the injection valve. The initial test was performed with a liquid rate of 0.47 l/s, but this proved to be not enough to push the gas bubble downward. It turned out that the critical bullheading rate was somewhere between 0.47 l/s and 0.606 l/s. Four successful bullheading tests were performed with different liquid rates starting from 0.606 l/s. A slug flow with dispersed bubbles below the Taylor bubble was observed in all the tests. All the tests were performed with an ideal gas (air) and water at room temperature.

Figure 4-1 shows the equipment set-up used in the experimental measurements. The experimental procedure steps are summarized in Figure 4-2 and Figure 4-3. Note that all figures, tables, and results shown below are taken from the master thesis work in [2].

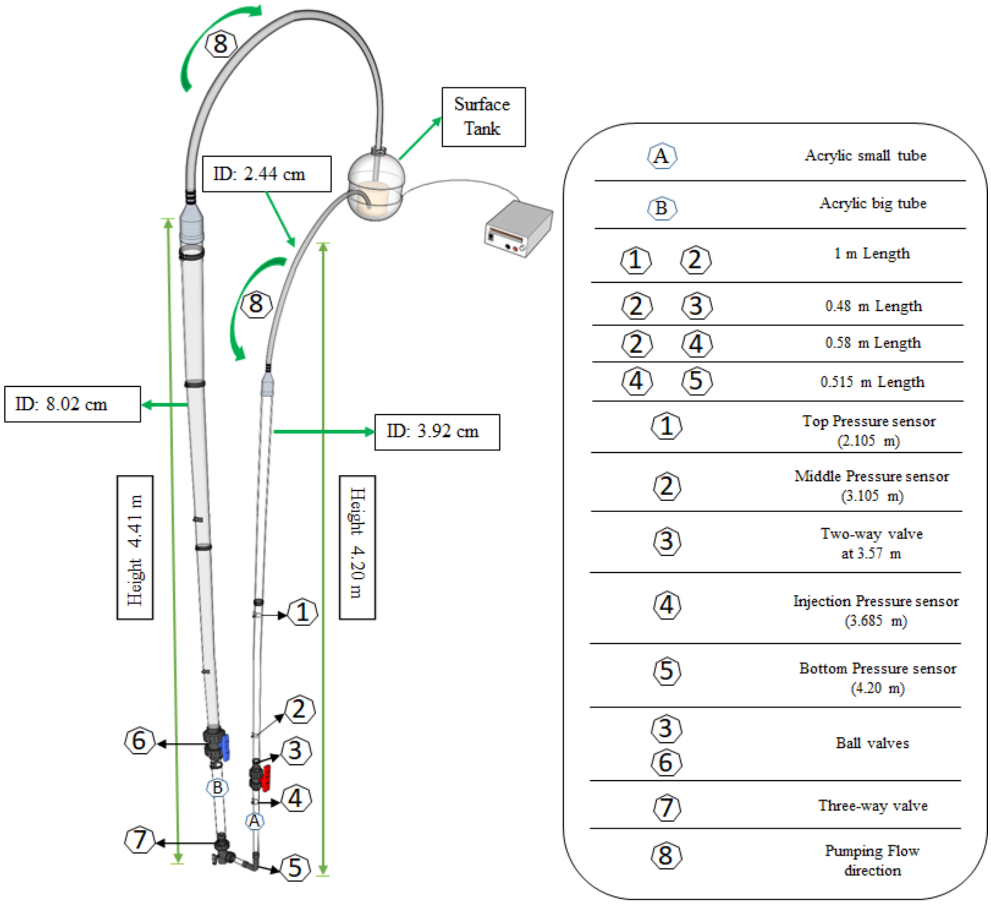


Figure 4-1: Experimental Equipment Set-Up, Taken from [2]

1. PASCO Capstone is configured. Pressure recording begins.

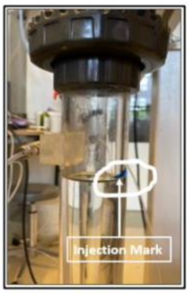
2. Both ball valves **(3)** and **(6)**; must be open to allow fluid to pass through the system, as well the Three-way valve **(7)** must be in this position ⊥ to avoid drainage of the system.

3. The surface tank must be filled with water preferably 10 cm above the intake of the pump to avoid suction of bubbles directly coming from the return hose. Circulate liquid through the acrylic tube **(A)**, and receives the fluid coming out from the tube **(B)**, to purge any trapped air bubbles and ensure that the system is filled with water. Then stop circulation.

4. Calibrate pressure sensors with the atmospheric pressure under laboratory conditions and the calculated hydrostatic pressure using the data from **Table 3.1**.

5. The next step is to close the red valve **(3)**.

6. Open the air inlet valve around the 10 sec. and let the air displace the liquid column bellow **(4)** until the *Injection Mark*. The amount of displaced liquid is supported with a pressure difference visualized in the software pressure recording.



7. Wait until second 40 to open the valve and allow the pressure to stabilize.

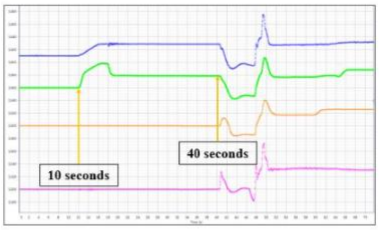
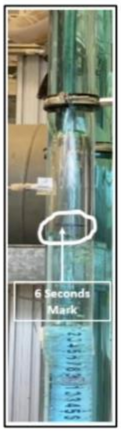


Figure 4-2: Experimental Procedure Steps 1-7, Taken from [2]

8. Open red valve **(3)**. The air bubble migration effect can be seen through the translucid acrylic tube. Taylor bubble migration begins.

9. The bubble is allowed to migrate to a second mark previously measured which gives the migration reference for 6 seconds.



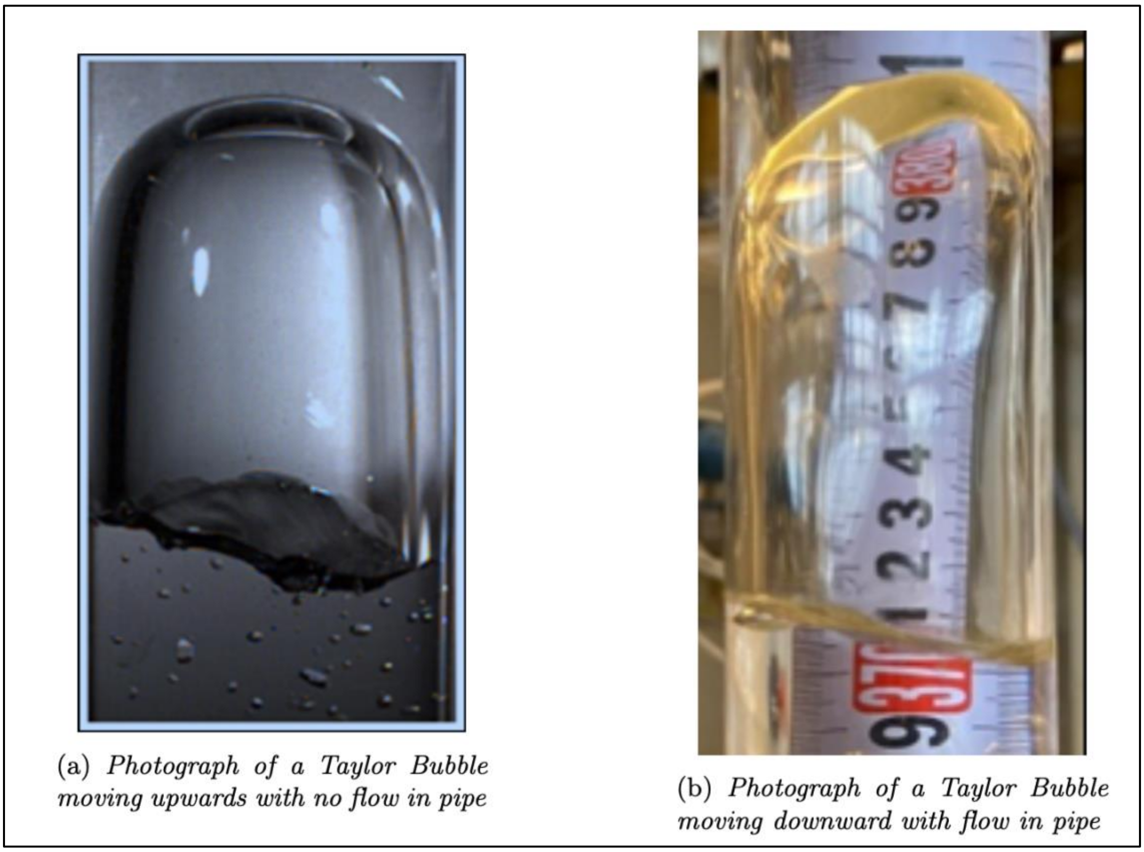
10. The pump is turned on at the voltage determined for the test being evaluated. It pumps liquid downward through the tube **(A)**. If the flow rate is low, the gas will still migrate upwards creating a countercurrent flow situation. If the flow rate is sufficiently large the Taylor bubble will start moving downwards.

11. Bullheading is performed until the bubble completely passes the injection sensor.
The experimental attempts should be repeated several times varying voltage supply to study the results with different flow rates.

12. Stop pump and sensor recording.

Figure 4-3: Experimental Procedure Steps 8 – 12, Taken from [2]

Figure 4-4 shows the shape of the Taylor bubble observed during the upward and downward movement. During the experiment, a slug flow was obtained with some dispersed bubbles within the flow.



(a) *Photograph of a Taylor Bubble moving upwards with no flow in pipe*

(b) *Photograph of a Taylor Bubble moving downward with flow in pipe*

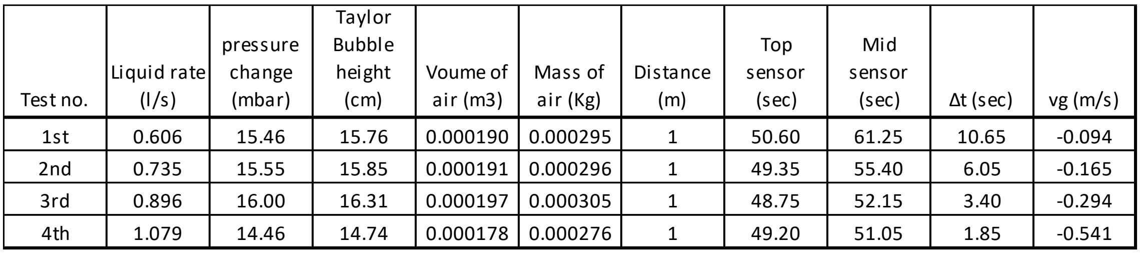
Figure 4-4: Bubble Shape during Upward and Downward Movement, Taken from [2]

The pressure at the mid sensor was measured when the tube above it was filled with water only and when the tube column was a mixture of water and all the gas bubbles. The drop in hydrostatic pressure can estimate the height of the gas bubble if we assume it occupies the whole cross-sectional area. This height was used in calculating the volume of gas injected during each test. The gas mass was found by multiplying the volume by the gas density calculated from equation 10 using the sensor pressure.

The average gas velocity during bullheading was found by measuring the Taylor bubble's time to move from the Top Sensor to the Mid Sensor. Knowing that the distance between the two sensors was 1m, the average gas velocity was calculated by dividing the distance of 1m by the time in seconds between the sensors during each test.

Table 4-1 summarizes the average results obtained from the experiments. For detailed calculations and analysis, the reader is referred to reference [2]

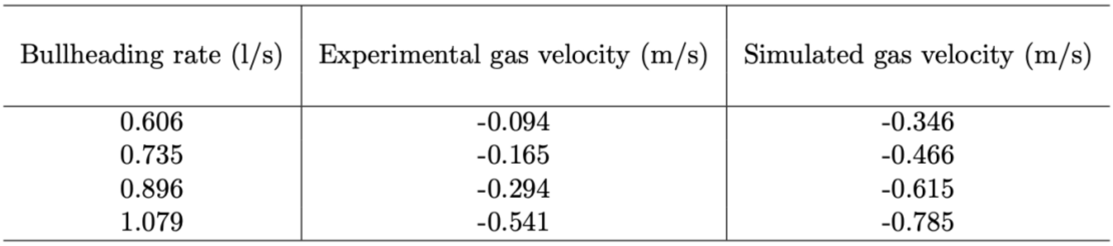
Table 4-1: Average Experimental Results, Taken from [2]



Test no.	Liquid rate (l/s)	pressure change (mbar)	Taylor Bubble height (cm)	Voume of air (m3)	Mass of air (Kg)	Distance (m)	Top sensor (sec)	Mid sensor (sec)	Δt (sec)	vg (m/s)
1st	0.606	15.46	15.76	0.000190	0.000295	1	50.60	61.25	10.65	-0.094
2nd	0.735	15.55	15.85	0.000191	0.000296	1	49.35	55.40	6.05	-0.165
3rd	0.896	16.00	16.31	0.000197	0.000305	1	48.75	52.15	3.40	-0.294
4th	1.079	14.46	14.74	0.000178	0.000276	1	49.20	51.05	1.85	-0.541

Finally, Patino Oscar and Alfonso Manuel [2] simulated the small-scale experiment and compared the experimental gas velocities with the simulated ones, as shown in Table 4-2. It is worth noting that the simulated gas velocity is the instantaneous gas velocity taken at 10 seconds.

Table 4-2: Gas Velocity (m/s), Experimental vs Simulated, Taken from [2]



Bullheading rate (l/s)	Experimental gas velocity (m/s)	Simulated gas velocity (m/s)
0.606	-0.094	-0.346
0.735	-0.165	-0.466
0.896	-0.294	-0.615
1.079	-0.541	-0.785

There is a significant difference between the simulation and the experiment. For a specific bullheading rate, the gas bubble moved at a more significant downward velocity in the simulation than in the experiment. The main conclusion drawn by the authors [2] was that the gas might have been more concentrated in the experiment than that in the simulation. They recommended re-running the simulation by using a more concentrated kick and reducing the numerical diffusion

4.3 Replication of Previous Simulation

The small-scale experiment introduced in section 4.2 was re-simulated using The Drift Flux Model and the AUSMV explicit scheme discussed in chapter 3. This work replicates the simulation part of the experimental study [2]. We obtained the same results, and the complete code for the simulation replication is provided in Appendix A. In this section, the main inputs and results of the simulation will be presented.

4.3.1 Gas Slip Model

The gas slip model discussed in section 3.1.3 is used. In addition, the equations are recalled in equations 46 and 47:

$$v_g = Kv_{mix} + S \qquad 46$$

$$v_{mix} = (\alpha_l v_l + \alpha_g v_g) \qquad 47$$

As mentioned in section 4.2, slug flow with Taylor bubble was observed in the experimental tests. Therefore, the following values were selected for K from section 3.1.3.4.

	Upward concurrent (Gas migration during shut-in)	Downward (Bullheading)
K	1.2	1.12

Since the experiment equipment set-up was for a vertical flow, the Taylor Bubble velocity in equation 48 was used in calculating the gas-rise velocity (S).

$$S = 0.35\left(gD\frac{(\rho_l - \rho_g)}{\rho_l}\right)^{\frac{1}{2}} \qquad 48$$

Where: D is the tubing inside diameter (0.0392m), g is the gravitational constant (9.81m/s^2), ρ_l and $\rho_g$are densities of liquid (water = 1000 kg/ m^3) and gas (air = 1.2 kg/ m^3) respectively. If we substitute the values in equation 48, we obtain S=0.2169 m/s, which is the value used in the simulation.

4.3.2 Density Models

In simulating the small-scale experiment, density models that include the effect of the temperature and pressure were implemented as in equations 49 and 50.

$$\rho_w = \rho_o + \frac{\rho_o}{\beta}(P - P_o) - \rho_o\alpha(T - T_o) \qquad 49$$

$$\rho_g = \frac{P}{RT} \qquad 50$$

Where: P$_o$ - reference pressure (100000 P$_a$), T$_o$ –water reference temperature (293.15 °K). β - the reciprocal of water compressibility factor (2.2 x 10^9 Pa), α - water thermal expansion factor = 207x10^{-6} K^{-1}. ρ_o - water reference density of water (1000 kg/ m^3). R

- ideal gas parameter for air = 286.9 J/Kg. K. The temperature T was considered constant and equal to T_o). The pressure P is the absolute pressure in Pa

4.3.3 Number of Boxes, CFL

The number of boxes for the replication case is 25. The CFL used in the simulation is 0.1018.

Other inputs can be obtained from the Matlab code provided in Appendix A.

4.3.4 The simulation Results

Figure 4-5 shows the gas volume versus time for the four experimental tests. As can be seen, as the bullheading rate increases, the time required to push the gas downwards and out of the pipe decreases.

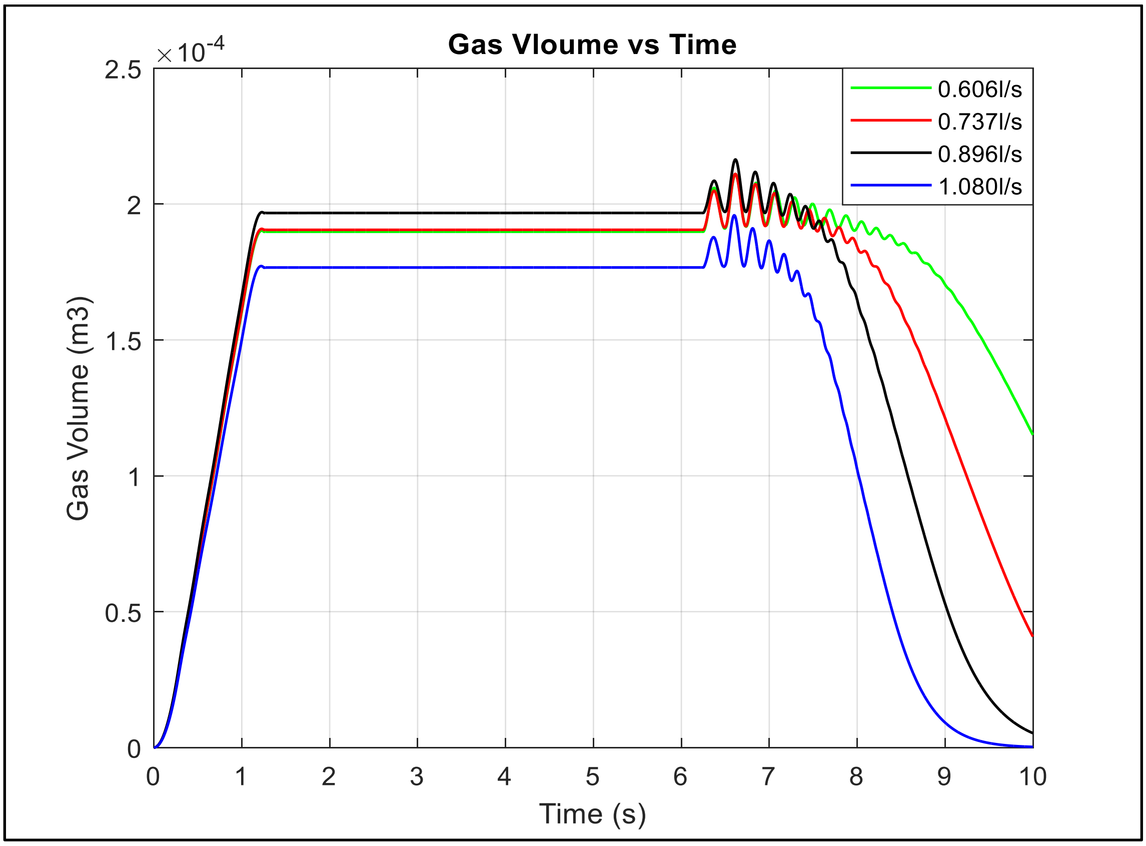


Figure 4-5: Gas Volume vs Time

Figure 4-6 shows the simulated instantaneous gas velocities at time 10s during the bullheading process. Note that this plot shows the gas velocity even though there is no gas present.

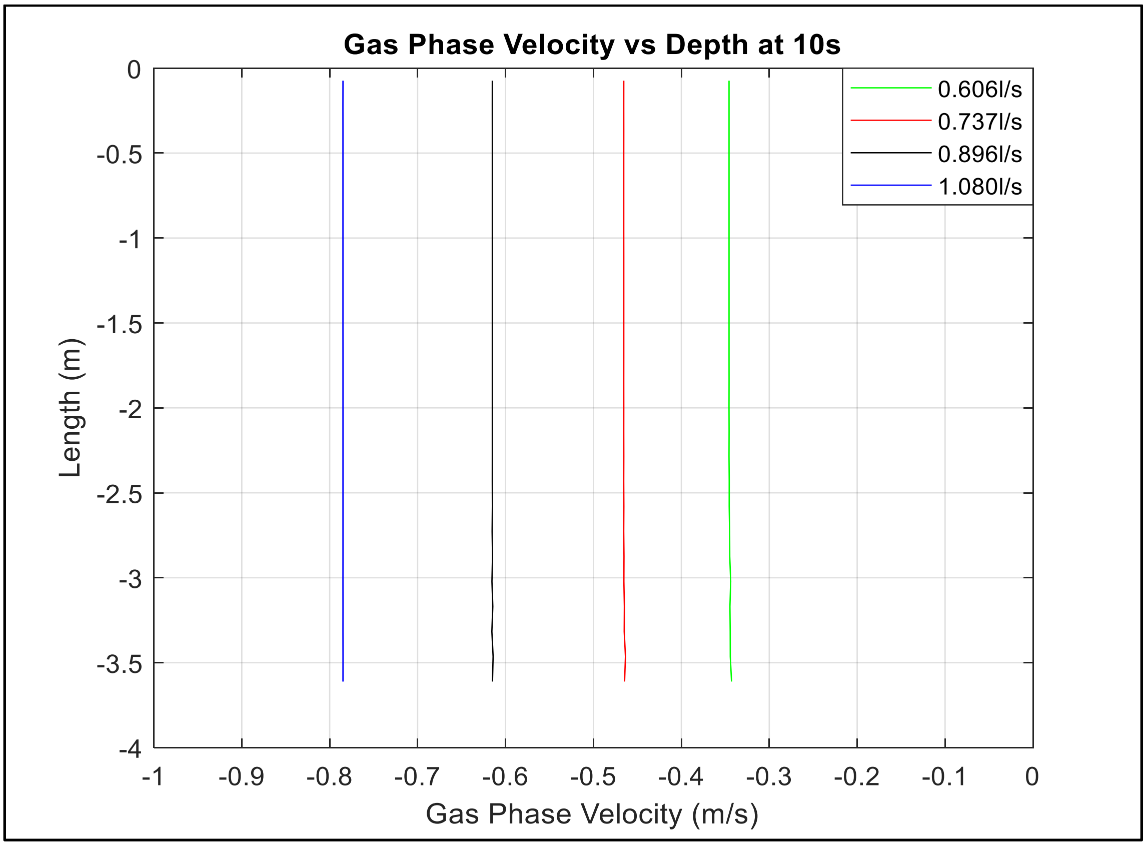


Figure 4-6: Gas Velocity vs Depth at 10s

The simulation gas velocity is compared to the experimental average velocity, as shown in Table 4-3. There is a significant difference between the experiment and the simulation results confirming what was seen in the previous work [2].

Table 4-3: Gas Velocity Experimental vs Simulated

Bullheading rate (l/s)	**Experimental vg (m/s)**	**Simulated vg (m/s)**
0.606	-0.094	-0.346
0.735	-0.165	-0.466
0.896	-0.294	-0.615
1.079	-0.541	-0.785

In the following sections, attempts will be made to improve the simulation results by testing two techniques: reducing the numerical diffusion by increasing the number of boxes and increasing the kick concentration

4.4 Effect of Reducing Numerical Diffusion

This section will study the effect of reducing the numerical diffusion by increasing the number of boxes (nobox) to 50 and 100 and compare the results with the base case of 25 boxes in section 4.3. Test 4 in the experiment was selected for the study, and the simulation results are discussed below. However, similar observations were obtained from investigating the numerical diffusion effects on the other three tests.

Figure 4-7 and Figure 4-8 show the total gas mass and the gas volume in the pipe versus time. Figure 4-9 and Figure 4-10 show the gas volume fraction versus depth at the start of bullheading and after 1.25s of bullheading pumping.

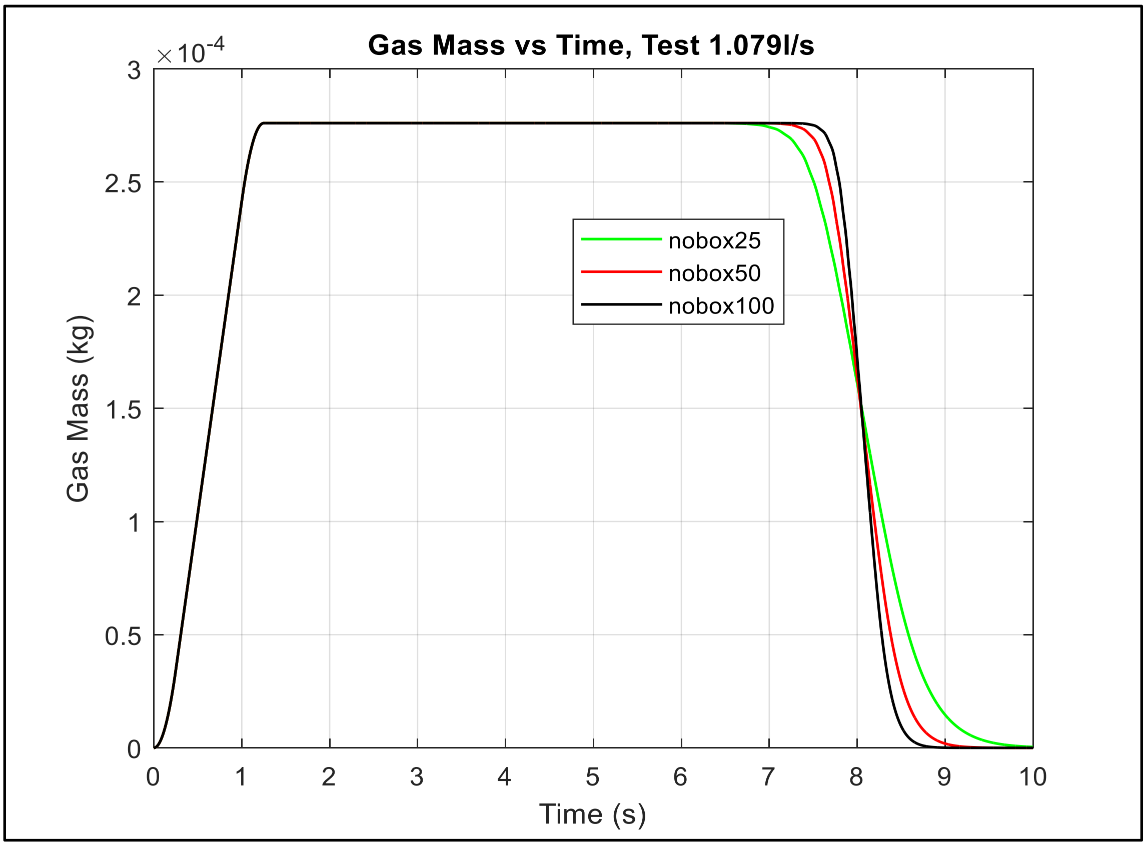


Figure 4-7: Gas Mass vs Time - Effect of nobox

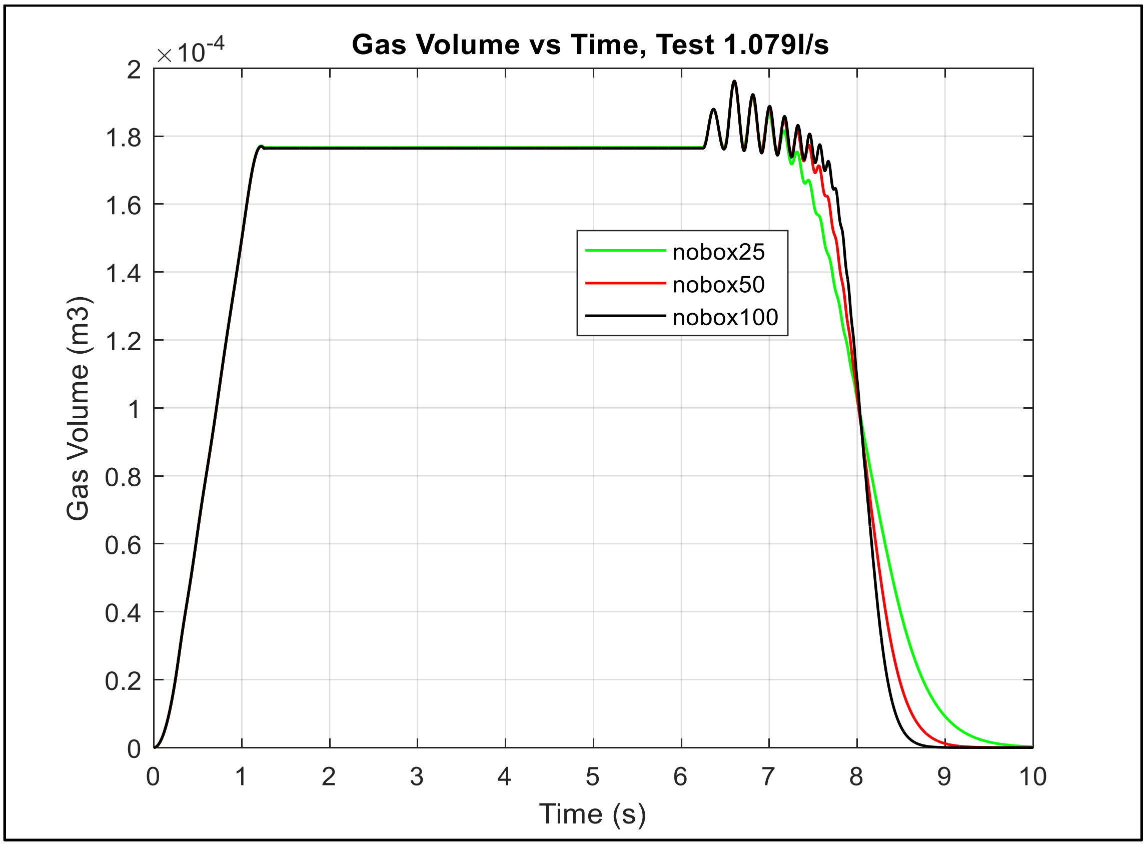


Figure 4-8: Gas Volume vs Time - Effect of nobox

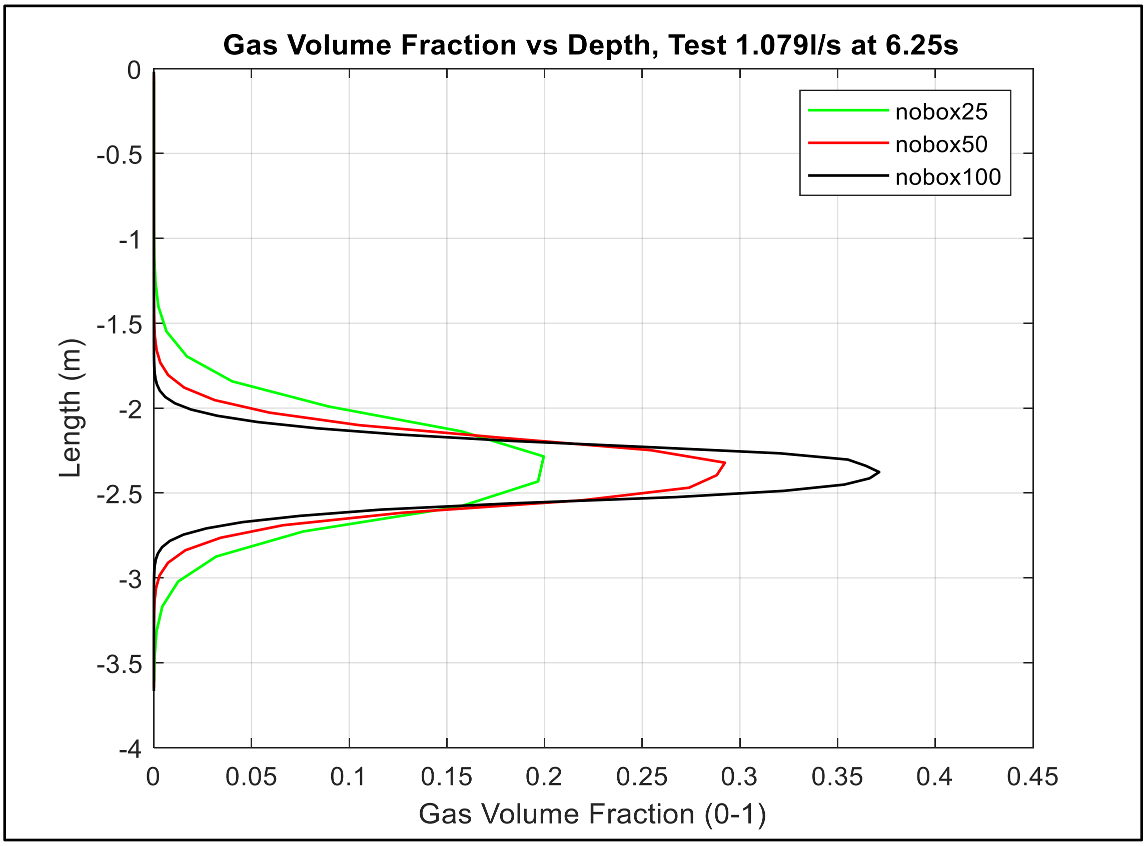


Figure 4-9: Gas Volume Fraction 6.25s

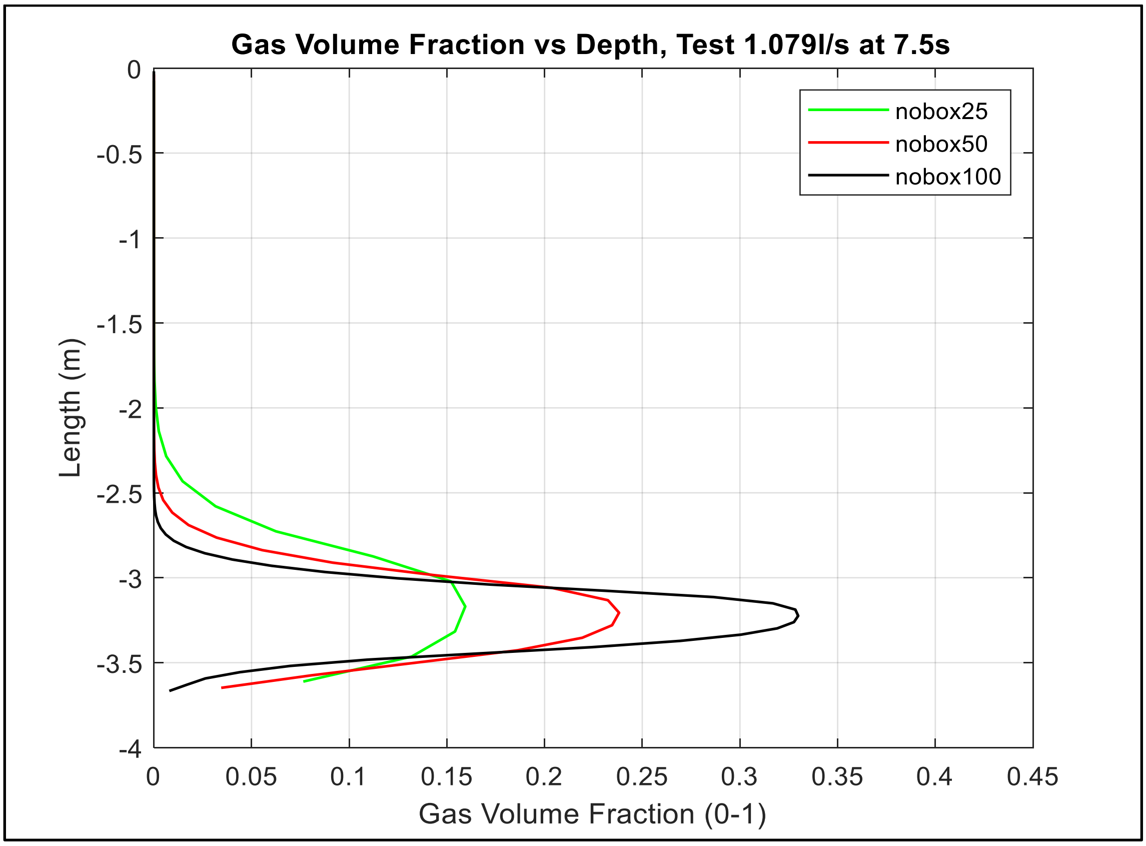


Figure 4-10: Gas Volume Fraction 7.5s

As shown in Figure 4-7 for the three nobox cases, the gas mass increases steadily during the injection period. It remains constant throughout the closed-in period and bullheading until the gas, pushed by the liquid, starts to leave the pipe. Then it is reduced to zero by the end of the bullheading process. The time at which the gas starts to leave the well and the duration of the bullheading process is different in the three cases. Although the kick starts to leave the pipe earlier for a lower nobox case, it takes longer to complete the bullheading process than for a higher number of boxes case. This is because increasing the number of boxes reduces the numerical diffusion, which makes the front and tail of the Taylor bubble sharper, as can be depicted in Figure 4-9 and Figure 4-10. As a result, removing the gas bubble in the low diffusion case (100 nobox) is easier than in the high diffusion case (25 nobox).

Similar observations can be obtained from the gas volume versus time graphs in Figure 4-8. However, the gas volume starts to decrease before the gas mass decreases. In other words, the gas volume in the well starts to decrease slightly even before the gas starts to leave the pipe. This is mainly due to the gas compressibility behavior. As the gas is pushed down the tube, the pressure increases, and its volume decreases. Another observation is the oscillations in the gas volume graphs. The main reason for these oscillations is the pressure pulses propagation caused by the pump rate change when the bullheading process is started. The pump starts from zero at 6.25 s and reaches its final flow rate at 6.50s. This will be discussed further in section 4.6.2.

In summary, reducing the numerical diffusion results in that the kick starts to leave the pipe later, but it will be removed from the pipe completely in a shorter time. Therefore, to obtain a realistic estimate of the duration of the bullheading process and the required liquid volumes to be pumped, it is crucial to reduce the numerical diffusion to an acceptable level.

4.5 Effect of Increasing the Kick Concentration

We continue with the 100-nobox case to study the effect of increasing the kick concentration. The same amount of gas will be injected but for varying injection duration. xp is the time when the gas injection will start to reduce. This will take the values 0.2, 0.5, and 1.0 seconds. The latter value will correspond to the base case used in the thesis [2] and the simulations presented in section 4.4. Finally, the injection is simulated, as shown in Figure 4-11. One can note that when the injection time interval is reduced, the gas mass rate is increased to maintain the same amount of gas termed mass in the figure. The mass is the total mass injected in the experiment (0.000276 kg) from Table 4-1.

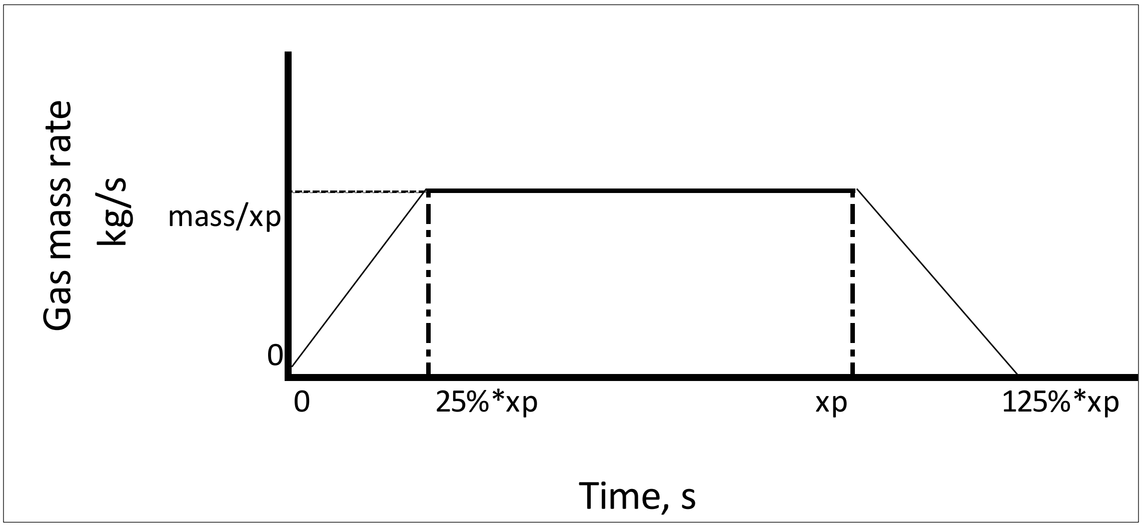


Figure 4-11: Injection of Gas in the Simulation

Figure 4-12 shows the gas mass vs time for different kick concentrations. As can be observed, the more concentrated kick starts to leave the pipe earlier and is completely removed from the pipe in a shorter time.

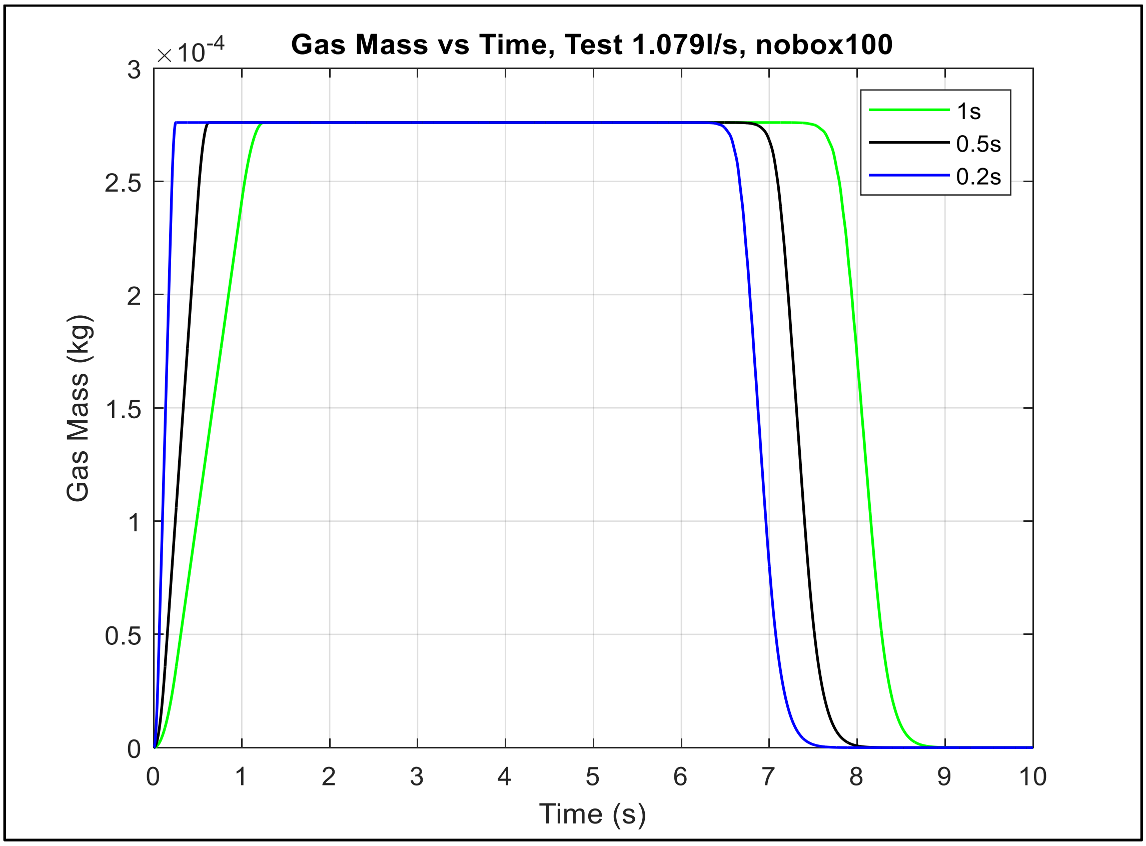


Figure 4-12: Gas Mass vs Time - Different Concentrations

4.6 Estimation of Gas Velocity for a More Concentrated Kick

This section will estimate the instantaneous and average gas velocities for the 100 nobox and xp = 0.2s concentration case. The simulation process is shown in Figure 4-13. First, the air is injected at 0.0kg/s at 0.0s and increased linearly to a 100% gas mass rate at 0.05s. Then the injection is maintained constant until 0.2s, and after that, it is decreased gradually to 0.0kg/s at 0.25s. At the end of injection, the well is closed in for 6s before the pump is started at 6.25s at 0.0 pumping rate and linearly ramped up to 100% capacity at 6.75s. After that, pumping is continued at the same rate until 10s.

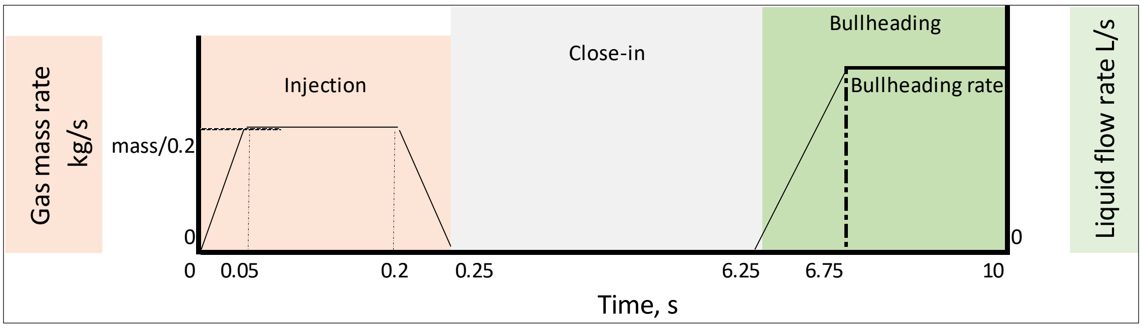


Figure 4-13:Injection of Gas, Close in, and Bullheading Periods used in the Simulation

The velocity of the Taylor bubble is estimated after the pump is fully started (after 6.75s) and before the gas leaves the well. Therefore, we will use time points of 7.5s and 8.0s in the calculations of the gas velocities.

We used a restart code and save workspace procedure to save time and avoid simulating from the start every time. First, the simulation is run for 7.5s, and the results in the workspace are saved with a specific file name. After that, the file is loaded using a restart script where only parts of the original MATLAB code are used. Then, the simulation is resumed to 8.0s, and the workspace results are saved again under a different file name. The workspace saved file can be loaded as many times as needed and simulation extended to any more extended time. A complete restart file code is provided in Appendix B. The advantage of this approach can, for instance, be seen when one shall perform several simulations where the only difference between the simulations will be, e.g., the bullheading rate. Then there will be no need to repeat the simulation process until the bullheading pump is started. Another advantage is that one does not need to start from time zero when storing depth data.

4.6.1 Initial Results

The gas velocities, liquid velocities, and gas volume fractions at 7.5s and 8.0s are plotted in Figure 4-14, Figure 4-15, and Figure 4-16. Experiment test 1 (0.606 l/s liquid rate) is used in the discussion. Two observations can be drawn from the graphs: the gas velocities change with depth, and the liquid velocities below and above the gas bubble are different, which does not look physically correct. The pipe has a uniform area, and when there is no gas, the liquid velocity should be the same when a constant liquid flow rate has been established.

The main reason for these variations in the velocities is the oscillations resulting from the pressure pulses propagating in the system, as mentioned in 4.4. These pulses will compress and decompress the fluids, making the velocities fluctuate again. Hence, they make it difficult to deduce the correct velocity. Therefore, we will try to dampen them in section 4.6.2 to predict the velocities better.

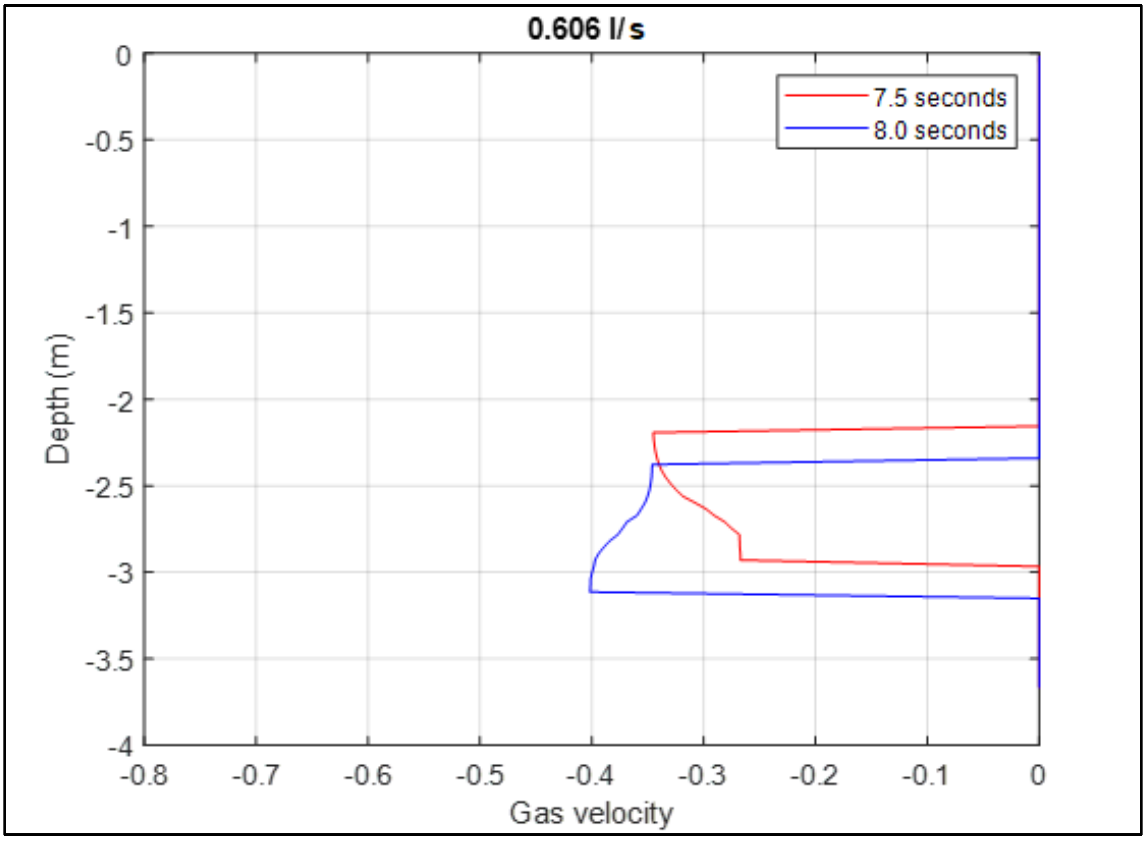


Figure 4-14: Gas Velocities vs Depth 7.5s and 8.0s

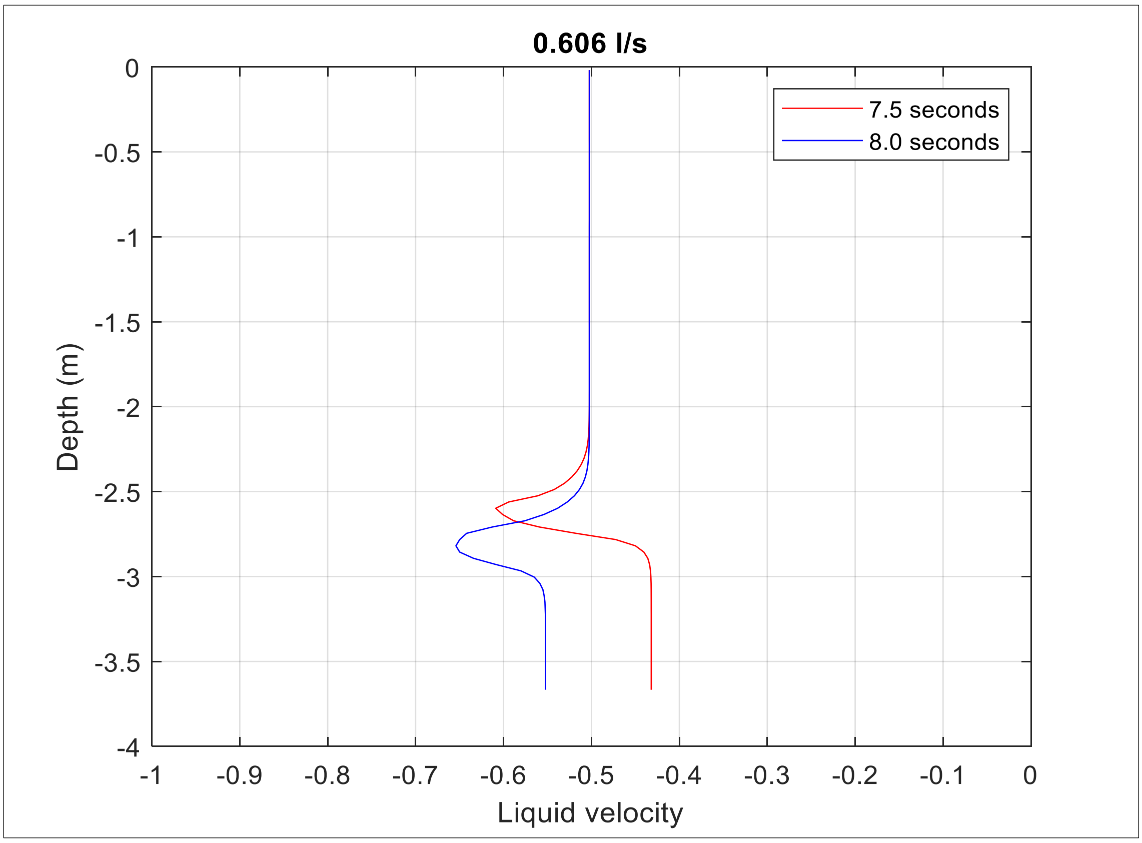


Figure 4-15: Liquid Velocities vs Depth 7.5s and 8.0s

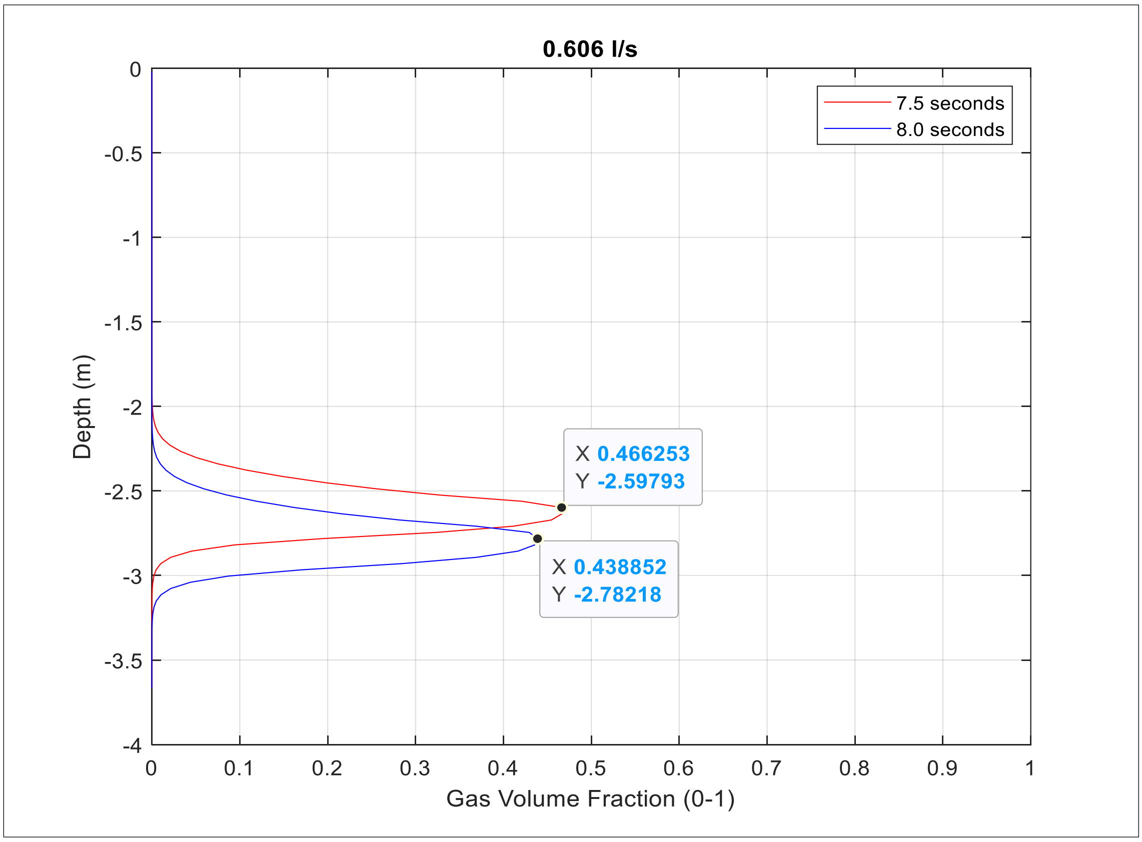


Figure 4-16: Gas Volume Fraction vs Depth 7.5s and 8.0s

4.6.2 Pressure Pulses Propagation in the Well

As seen in Figure 4-8, the oscillations in the gas volume versus time graphs were caused by the pressure pulse propagation. Pressure pulses are generated when the flow condition in the well is changed, e.g., the pump stopped or started. In the small-scale experiment, there are three situations where pressure pulses are generated: when the gas injection is started and stopped and when the pump is started for bullheading. Pressure pulses are damped as time passes due to the friction in the system. They also become less severe if the changes in pump rate are smooth.

Figure 4-17 shows the bottom-hole pressure plotted versus time. First, the pressure increases due to added friction when we force the gas into the pipe. The pressure then decreases after the gas injection due to the reduction in the hydrostatic column. Here one can note some oscillations, and the injection stop of gas causes these. Then the pressure increases steadily during the close-in period due to gas migration. Finally, during the bullheading, the bottom-hole pressure is constant and equals the pressure boundary condition of 1.35bar, specified in the simulation.

Consequently, the bottom-hole pressure vs time graph shows the pulses when the gas injection stopped only. If we zoom in and re-plot the pressure up to 2s as in Figure 4 18, these pressure pulses can be seen clearly.

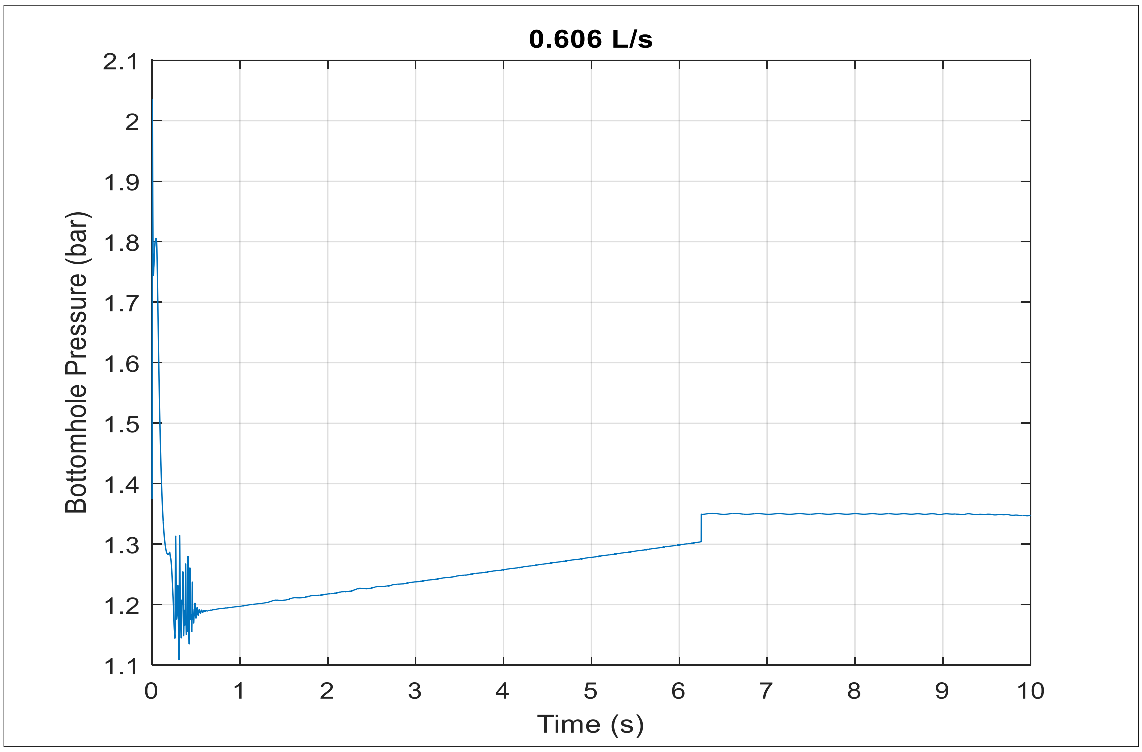


Figure 4-17: Bottomhole Pressure vs Time

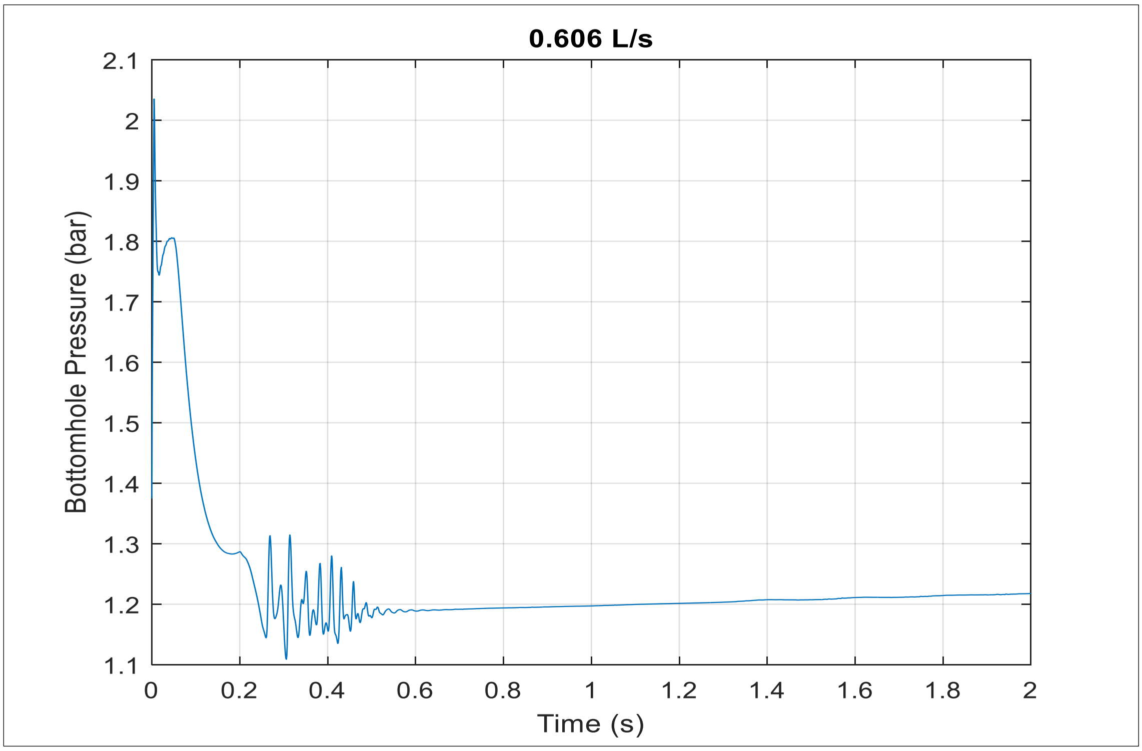


Figure 4-18: Bottomhole Pressure vs Time -End of Injection Pressure Pulses

The gas volume- time plot shows evidence of the pressure pulses due to the pump start for bullheading. This can be seen in the gas volume - time plot as in Figure 4-19. These fluctuations in the gas volume are caused by the gas being compressed and decompressed as the pulses migrate in the system. Although they decay with time, they affect the gas and liquid velocities, causing some variations that make it difficult to deduce the correct instantaneous velocities.

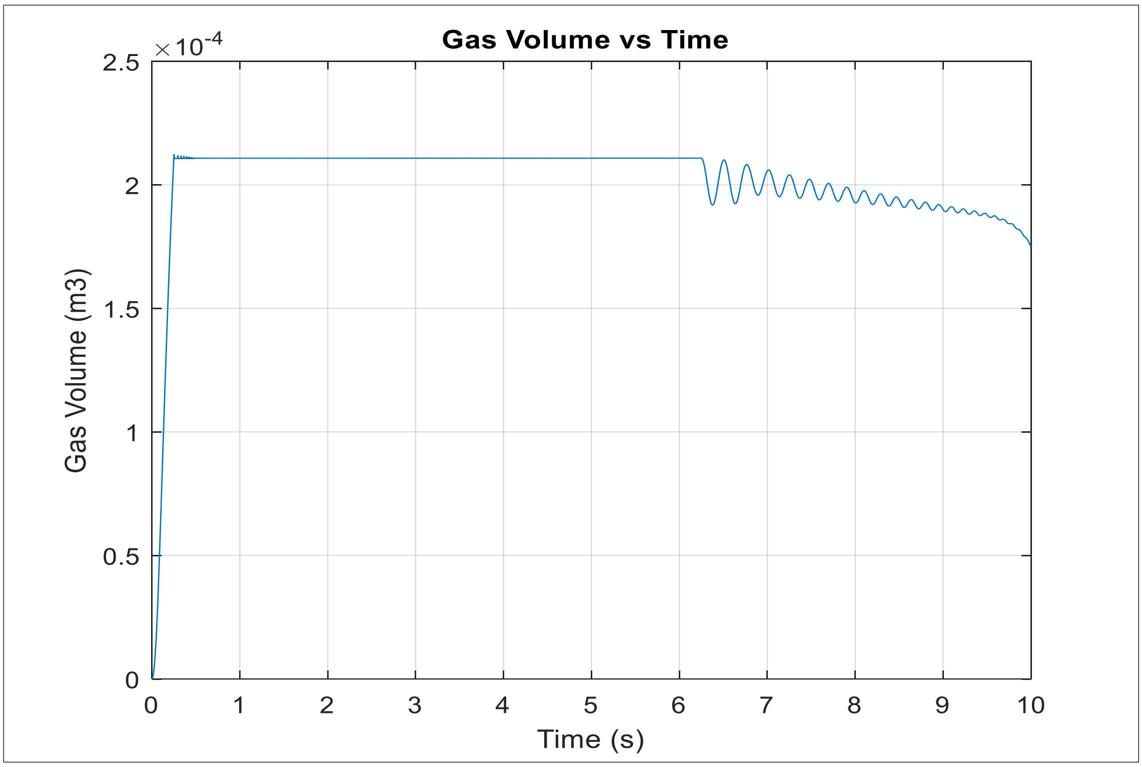


Figure 4-19: Gas Volume vs Time - Pump Start Pressure Pulses

4.6.2.1 Dampening the pressure pulses

To dampen the pressure pulses, we first extended the period for the pump to ramp up from 0.25s to 0.50s, but this proved not to improve the results. Then we used artificially larger viscosities of the liquid (0.200 Pa. S) and gas (0.020 Pa.S) in the restart file. This removed the pressure pulses completely and gave consistent velocity values. The increased viscosities will increase the friction, and the pulses will be dampened out more quickly. It is important to note that artificial viscosities are only recommended when evaluating the instantaneous gas velocity. One will then get rid of the pressure fluctuation that will compress/decompress the fluids, which leads to fluctuations in the velocities. Increasing the friction in the system will not impact the movement of the gas bubble, but it will lead to increased pressures in the pipe, which will impact the gas volume fraction. The results are summarised in the next section.

4.6.3 Estimation of the Gas Velocity after Dampening the Pulses

Figure 4-20 and Figure 4-21 show the gas and liquid velocities after removing the effect of the pressure pulsations plotted for Test 1 (0.606 l/s). We have a constant gas velocity vs depth and the same liquid velocity above and below the gas bubble. Figure 4-22 shows the gas volume fraction versus depth. Note that the front position is the same as previously reported, but the gas volume fraction values differ slightly from those with pulses in Figure 4-16. Therefore, the technique is suitable for estimating the gas velocity only.

As can be seen from Figure 4-22, as the gas bubble gets deeper down the well, the gas fraction is reduced, and this is because the pressure has increased slightly since the hydrostatic pressure of water above has increased. From the liquid velocity plot, one can also see that the liquid velocity becomes more negative around the gas bubble, which is natural since there is less space for the liquid to pass.

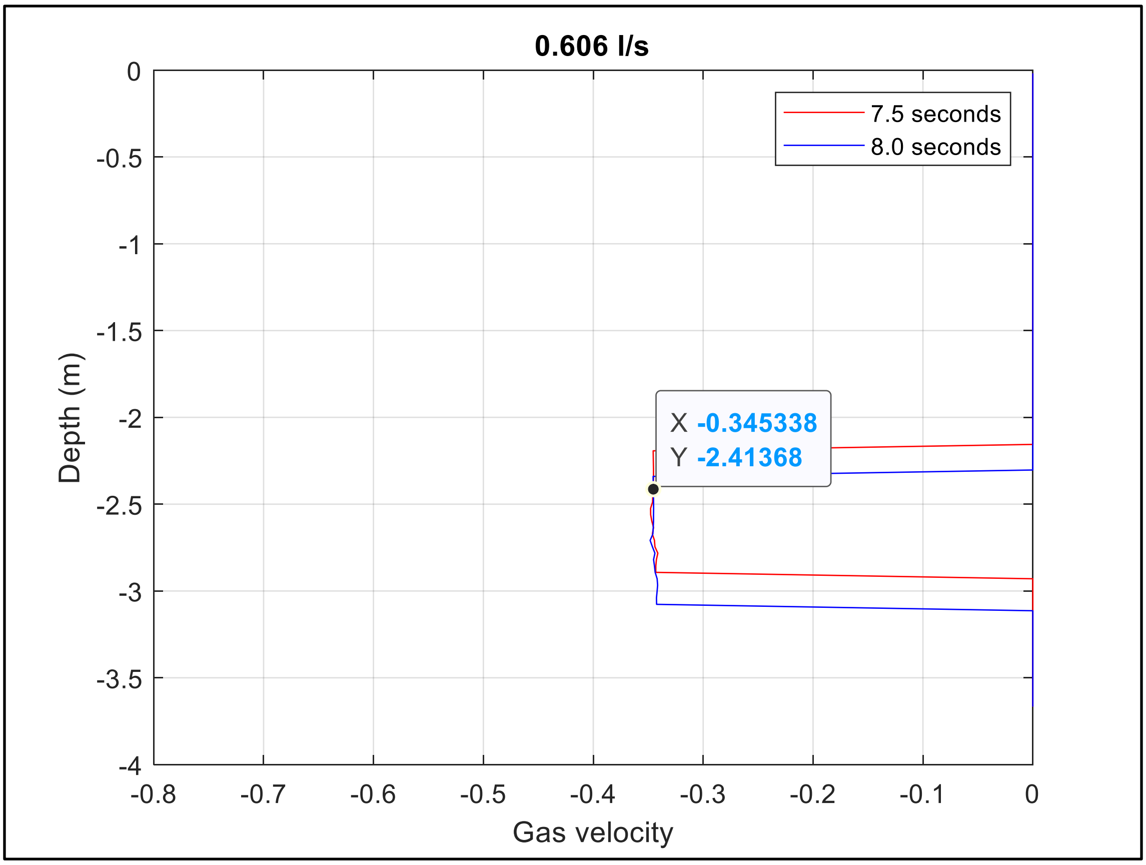


Figure 4-20: Gas Velocity vs Depth - Pulses Dampened

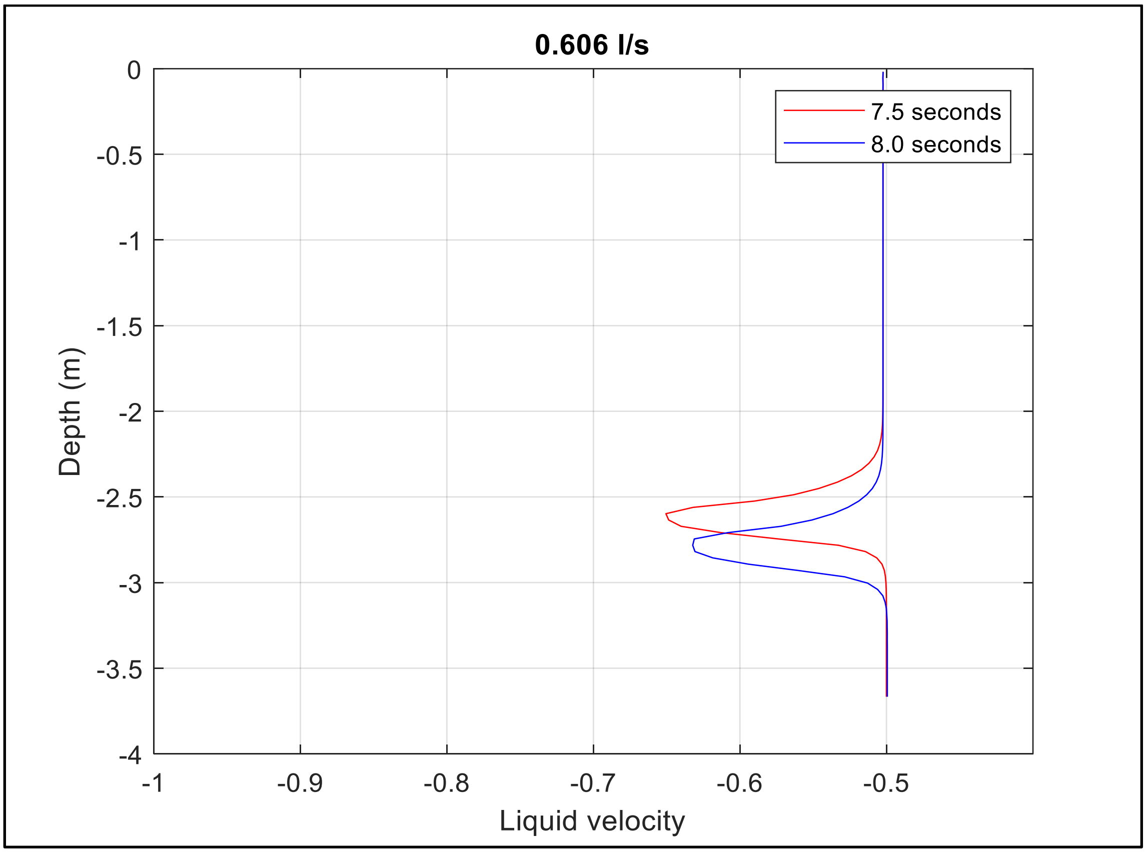


Figure 4-21: Liquid Velocity vs Depth - Pulses Dampened

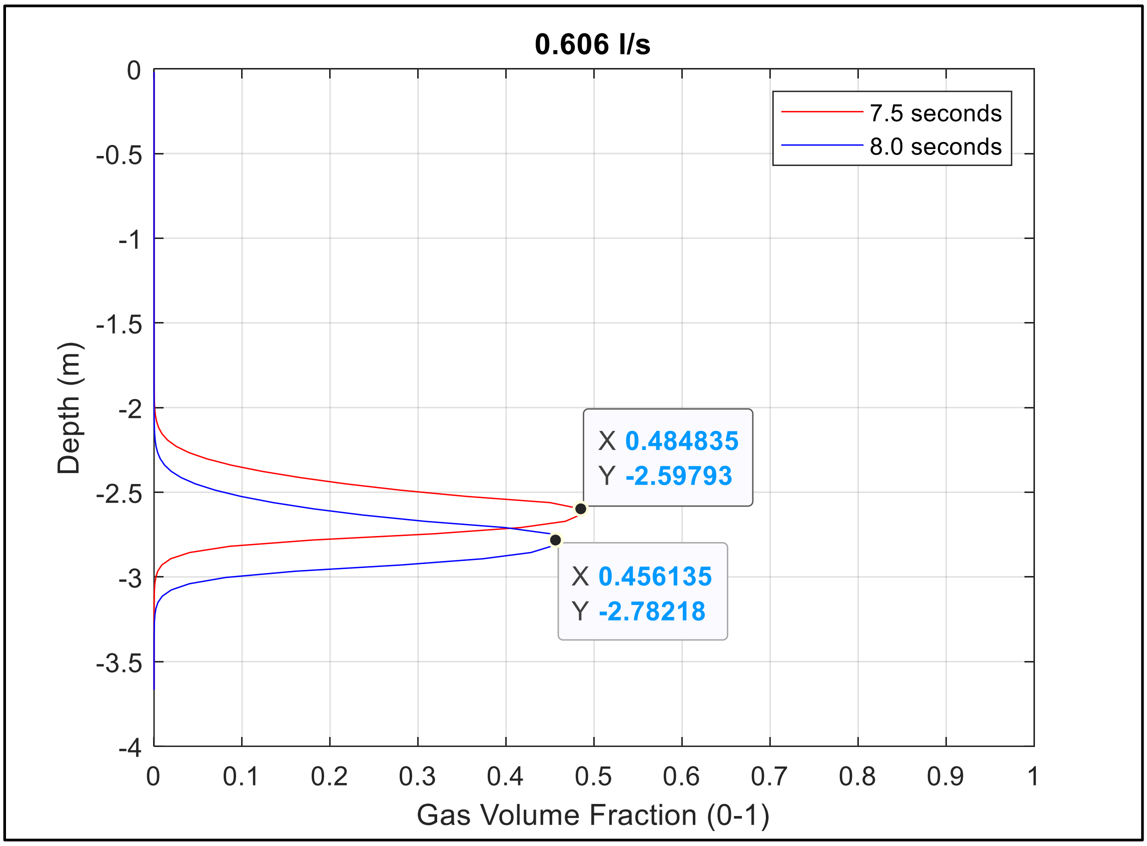


Figure 4-22: Gas Volume Fraction vs Depth - Pulses Dampened

The front positions at 7.5s and 8.0s are used in calculating the average gas velocity during the bullheading process.

$$vg = \frac{distance}{time} = \frac{-2.78218 - (-2.59793)}{8.0 - 7.5} = -0.369\ m/s \qquad 51$$

The procedure was repeated for the other three tests in the experiment, and the results are summarised in Table 4-4.

Table 4-4: Average and Instantaneous Gas Velocity Results

Liquid rate, l/s	(7.5s) Y, m	(8.0s) Y, m	vg, m/s average	vg, m/s instantaneous
0.606	-2.5979	-2.7822	-0.369	-0.340
0.735	-2.7453	-2.9664	-0.442	-0.460
0.896	-2.8927	-3.1875	-0.590	-0.610
1.079	-3.0401	-3.4455	-0.811	-0.780

Comparing these results with the results obtained from the experiment and the corresponding simulations as reported in Table 4-3, we conclude that making the kick more concentrated and reducing the numerical diffusion had no impact on how fast the Taylor bubble moves downward.

4.7 Critical Bullheading Rate

4.7.1 Calculation of Critical Bullheading Rate

The critical bullheading rate is the pumping rate at which the gas bubble remains stationary (vg = 0). It is an important parameter to estimate in the bullheading job design. One needs to overcome the critical flow rate to push the kick downward and back into the formation. It can be calculated by setting vg equal to zero in equation 20.

$$v_g = Kv_{mix} + S \qquad 52$$

$$v_{mix} = \alpha_l v_l + \alpha_g v_g \qquad 53$$

$$v_g = K(\alpha_l v_l + v_g \alpha_g) + S \qquad 54$$

$$0 = K\alpha_l v_l + S \qquad 55$$

$$Q_{liq} = A\alpha_l v_l = -S\frac{A}{K} \tag{56}$$

By substituting S = 0.2169 m/s, K = 1.12 and $A = \frac{\pi}{4}(0.0392)^2 = 1.2069\ l/meter$

$$Q_{liq} = 0.236\frac{l}{s}$$

4.7.2 Estimation of the Critical Bullheading Rate from the Simulation

We will now try to find the critical bullheading rate from the simulation and check if the formula in equation 56 can be used to estimate this rate. We will vary the bullheading rate parameter until we obtain a value that gives $v_g = 0$. Our upper point is 0.606 l/s used in the simulation as we are confident it is above the critical pumping rate. Also, the gas mass and gas volume curves should be flat during pumping at a critical pump rate as the amount of gas in the pipe remains unchanged.

Figure 4-23 and Figure 4-24 show the gas mass and gas volume versus time for various pumping rates. As can be observed, the mass curve is flat for pumping rates 0.200, 0.236, and 0.240 l/s, which means no gas has left the pipe during bullheading. However, looking closer at the gas volume graphs, we notice that the 0.240 l/s case shows a slight decrease in the gas volume with time. This is probably due to the gas being pushed down some distance but not far away, and its volume decreases very slightly with the slightly higher pressure at that depth.

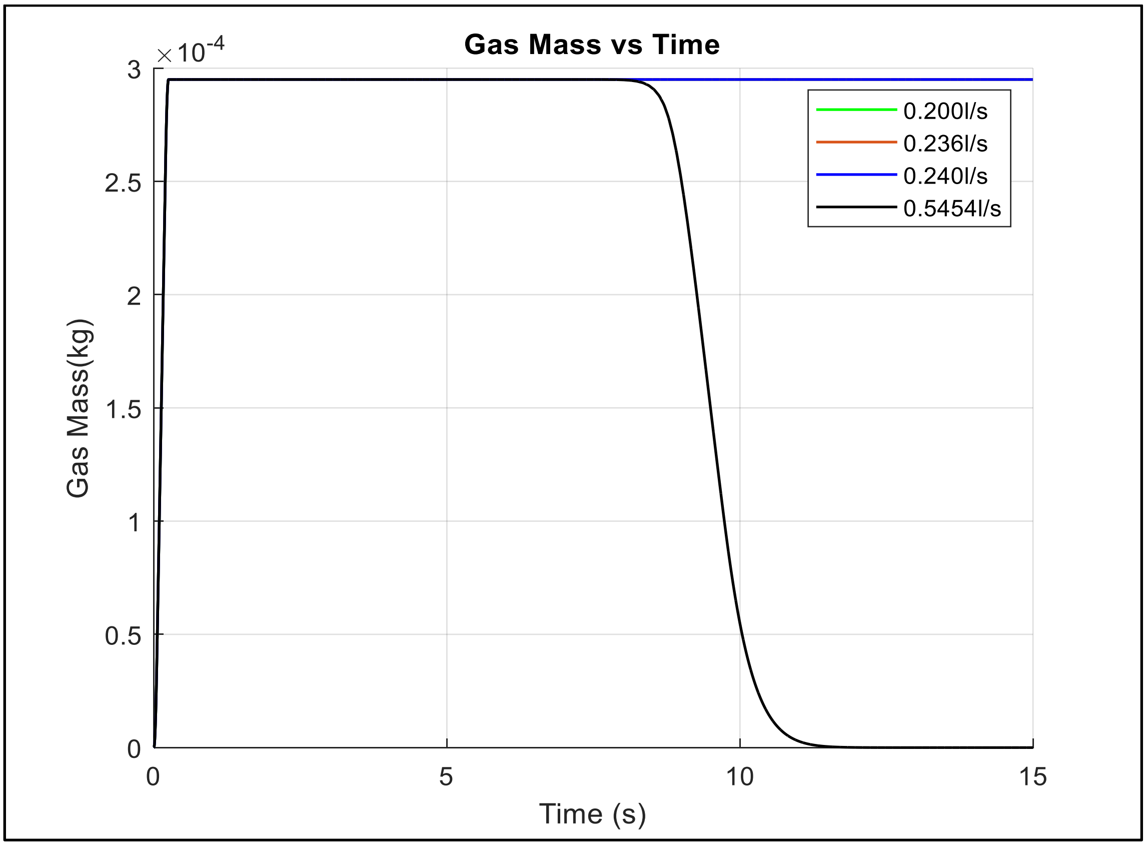


Figure 4-23: Gas Mass vs Time - Investigating the Critical Rate

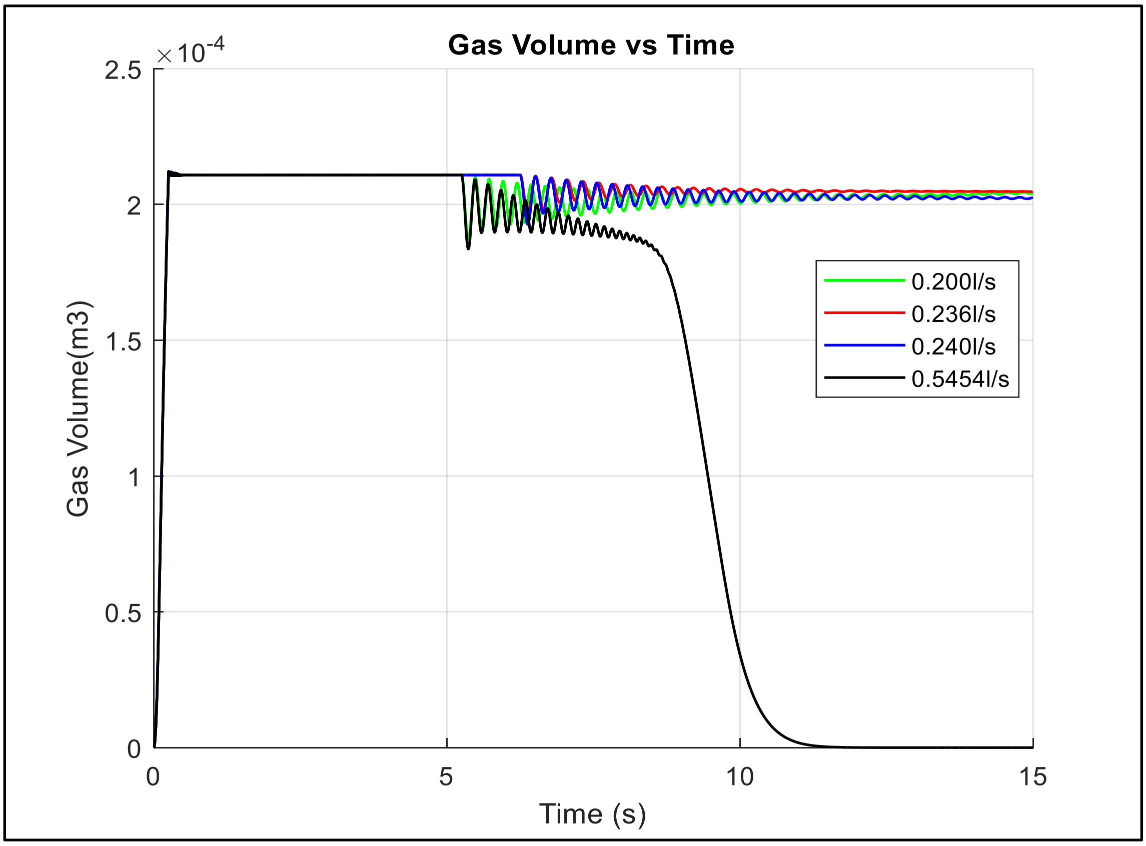


Figure 4-24: Gas Volume vs Time - Investigating the Critical Rate

Finally, we plot the gas velocity versus depth in Figure 4-25. Figure 4-25 confirms the downward movement for the gas in the 0.240 l/s case as it shows a negative velocity although it is small. The 0.236 l/s case shows almost zero velocity. The 0.200 l/s shows a more positive gas velocity which means the bubble continues to move upward during bullheading. It is a small positive velocity because the pumping rate of 0.200 l/s is close to the critical bullheading rate. Therefore, we accept that the critical bullheading rate is 0.236 l/s agreeing with the calculations in section 4.7.1.

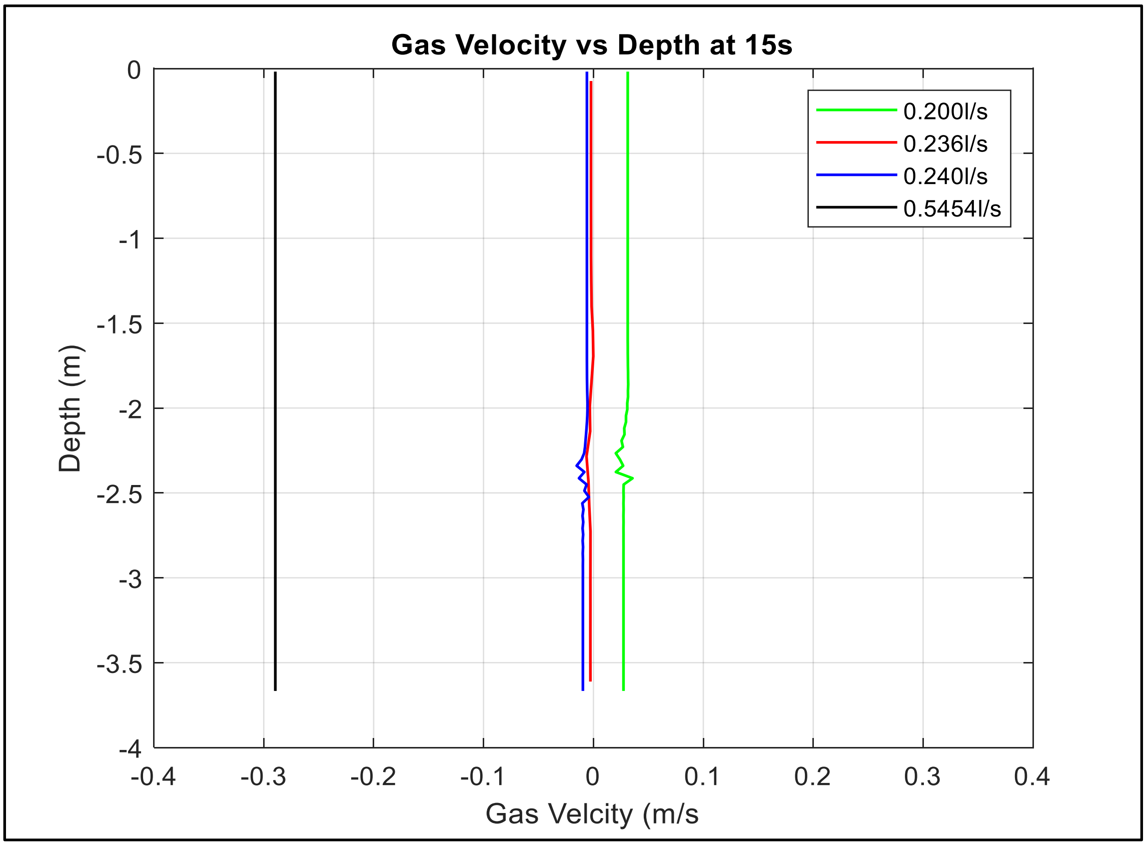


Figure 4-25: Gas Velocity vs Depth - Investigating the Critical Rate

4.7.3 Critical Bullheading Rate from the Experiment

As mentioned in section 4.2, the experiment started with a liquid rate of 0.47 l/s which was insufficient for bullheading the gas bubble. Then the liquid rate was increased to 0.606 l/s in the first test, and it was able to bullhead the air bubble entirely out of the tube. This means the experimental critical bullheading rate lies between 0.47 and 0.606 l/s.

If we put these two liquid rates in equation 56, we can evaluate the corresponding slip parameter S:

$$Q_{liq} = -S\frac{A}{K} \qquad 57$$

$$0.47\ l/s = -S\frac{1.2069\ l/m}{1.12} \qquad \text{S = 0.4361} \qquad 58$$

$$0.606\ l/s = -S\frac{1.2069\ l/m}{1.12} \qquad \text{S = 0.5623} \qquad 59$$

According to the Taylor bubble velocity (equation 23), such high S values require a pipe diameter of 158 mm and 263 mm, respectively, compared to the 39.2 mm we have in the experiment. The model in equation 23 is widely accepted in predicting the slip velocity for slug flow.

The critical bullheading rate seems to be overestimated in the experiment compared to what we should expect from a widely accepted model for the Taylor bubble velocity. Nevertheless, our simulation shows that the proposed critical bullheading rate can easily bullhead the gas-induced bubble. The simulation limits the critical bullheading rate to 0.236 l/s, and its results agree with the simple calculations in 4.7.1.

4.8 Changing the S Value to fit Experiments Better

The simulation was run for slip velocity S of 0.5 m/s. The gas velocity results agree with the experiments. Figure 4-26 and Figure 4-27 show the gas velocity versus depth for Test 1 and Test 4 of the experiment. One can observe that the gas velocities of 0.07 m/s and 0.50 m/s agree much more with the experimental measurements of 0.09 m/s and 0.54 m/s in the two tests. Although this does not seem very realistic, should the widely used formula for S be correct (equation 23).

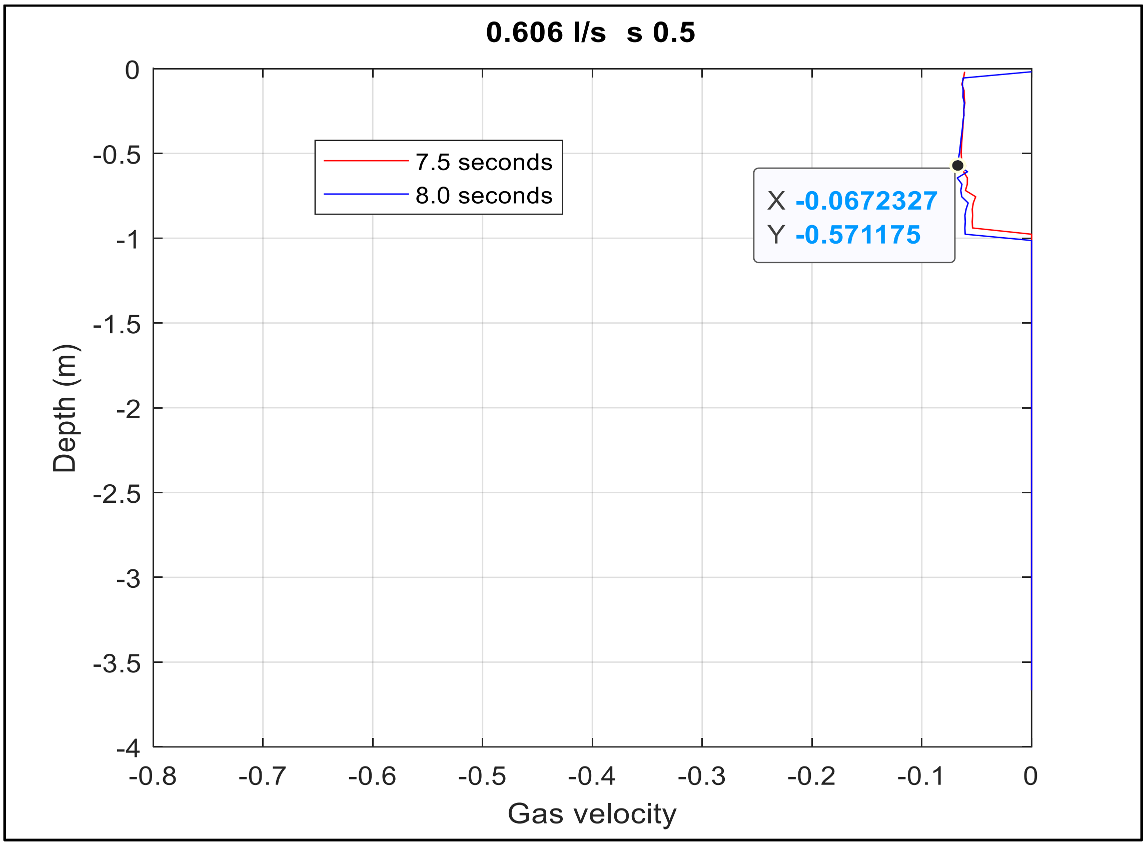


Figure 4-26: Gas Velocity vs Depth for S = 0.5 M/S – Test 1 (0.606 l/m)

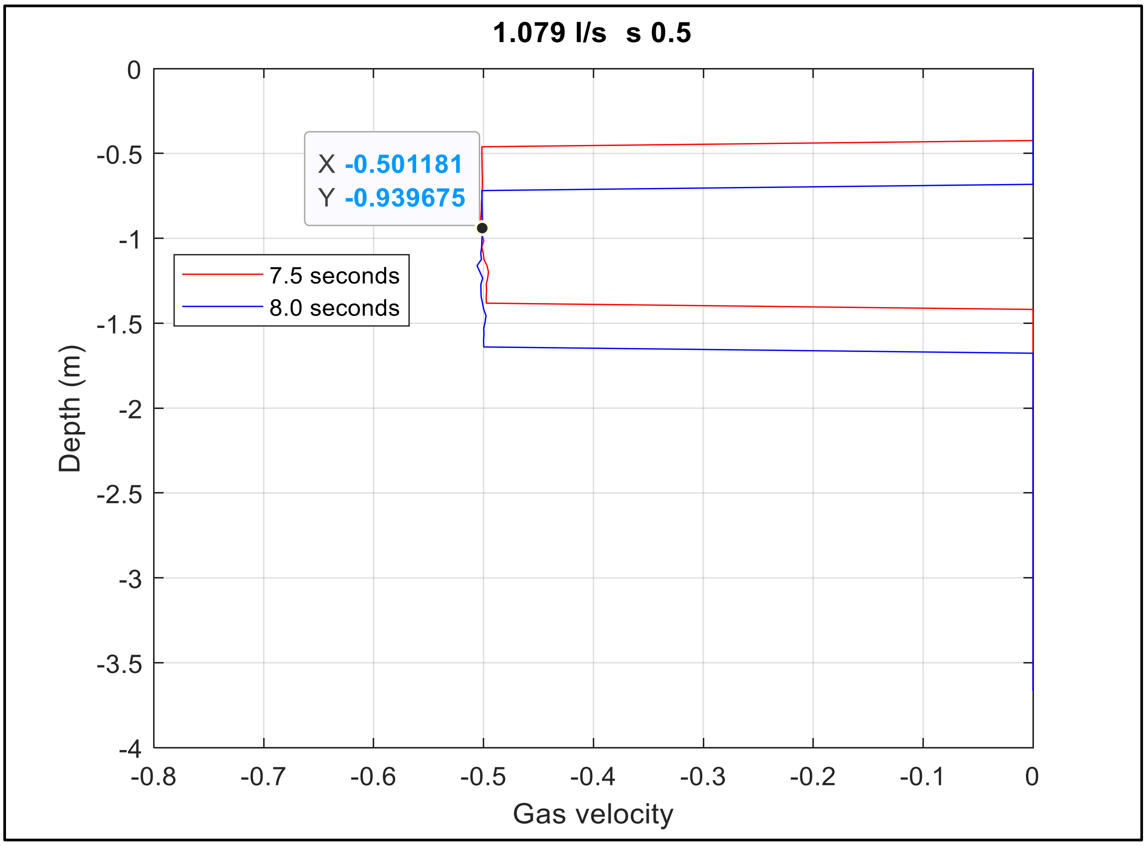


Figure 4-27: Gas Velocity vs Depth for S = 0.5 M/S – Test 4 (1.079 l/S)

4.9 Discussion of the Results and Recommendations

The simulation of the small-scale experiment was replicated using the experimental data. The complete code of the main script is provided in Appendix A. In addition, the gas slip and density models used were briefly discussed in 4.3.1 and 4.3.2.

For the simulation of the small-scale experiment, a single gas slip model was used. This was sufficient for the small loop experiment as the gas bubble did not change flow pattern as it migrated up the short tube.

The simulation results were compared with the experiments in terms of the gas's downward velocity during the bullheading process. There was a significant difference in the gas velocity, as summarized in Table 4-2 and Table 4-3, also seen in [2].

Attempts to improve the simulation results were made using two techniques: reducing the numerical diffusion (see section 4.4) and making the kick more concentrated (see section 4.5). The numerical diffusion was reduced by increasing the number of cells used in the discretization. The kick concentration was increased by injecting the same amount of gas mass but in a shorter time.

The numerical diffusion reduction and the kick concentration increase made the tail and front of the bubble sharper. As a result, the gas was bullheaded in a shorter time. Therefore, reducing the numerical diffusion to an acceptable level is vital in estimating the duration of the bullheading process and the required volume of fluids to be pumped.

However, the numerical diffusion and the kick concentration did not impact the gas velocity much. Therefore, we obtained close results to the original simulation. Hence the large discrepancy between the experiment and the simulation remains.

The average downward gas velocity during the bullheading process was estimated from the front position at two different time points in the simulation. The time interval was 0.5s, and the time points (7.5s and 8.0s) were selected after the pump was fully operating and before the gas bubble started to leave the pipe. In addition, the instantaneous gas velocity versus depth was plotted from the simulation directly. Here we excluded values where the gas volume fraction was less than 0.01. The average and instantaneous showed a good agreement with each other. The minor differences are mainly caused by some inaccuracies when reading from graphs.

In applying the Drift Flux Model to a small-scale experiment, one needs to realize that oscillations will be generated by the pressure pulses propagation in the system when the pump is started or when the air injection is stopped. The system is small and sensitive to any changes. These pressure oscillations in our simulation were observed in the bottomhole pressure versus time graph and the volume versus time graph (see section 4.6.2). The gas volume fluctuations were caused by the gas being compressed and decompressed as the pulses migrated in the system.

Although the pressure pulses decay with time, they cause variations in the gas and liquid velocities making it difficult to deduce the correct values (see section 4.6.1). For example, when we plotted the gas and liquid velocities versus depth at two times, the gas velocity graph showed variation in the downward velocity with depth. Likewise, the liquid velocity graph showed different values above and below the bubble.

To determine the instantaneous gas velocity, we tried two methods to get rid of the pressure pulses (see section 4.6.2.1). We first doubled the time for the pump to ramp up from 0.25s to 0.50s, but this proved not to help much. Then we introduced artificial higher viscosities for the liquid and gas to increase the friction force. This removed the pressure pulses entirely, and we could see consistency in the velocity results. The increasing friction will not have any impact on mass transport. It was also shown that the gas volume fraction changed slightly though (see 4.6.3). Therefore, we recommend only using the artificial viscosities technique when the instantaneous gas velocity shall be estimated. The average gas velocity would not, in any case, be affected by the presence of pulses since neither presence of pulses nor an increase in friction will have any significant impact on the transport process.

The critical bullheading rate is an important parameter to determine, and one needs to overcome it to bullhead the gas bubble. In this study, it was estimated by setting the gas velocity (vg) equal to zero in the gas slip model shown in 4.3.1. Then it was estimated from the simulation by varying the liquid rate and running the simulation (see 4.7.2). The results were shown in the Mass Gas vs Time, Volume Gas vs Time, and Gas Velocity vs Depth graphs. At the critical liquid rate, the mass gas–time curve is flat, the gas volume–time curve is flat, and the gas velocity is zero. There were cases where the gas mass–time curve was flat but was excluded from being critical as the corresponding volume gas–time curve showed a slight decrease. This is because the gas was pushed down some distance in the system, and its volume decreases at higher pressure. Both the simulation value and the calculated value for the critical rate agree.

The simulated critical bullheading rate is much lower than the one determined by the experiment. The simulation estimates the critical rate as 0.236 l/s, and the experiment estimated it between 0.47 l/s and 0.606 l/s. The experiment predicts a much larger critical flow rate than what was derived from the theoretical slip model. The measured negative gas velocities are much lower in absolute value than what was seen in the simulations. The gas bubble will move downward faster in the simulation than in the experiment for a fixed bullheading rate.

We tested the simulation for S = 0.5 m/s for slug flow. The results show a better fit for the experimental results. However, this does not seem very realistic should the widely used formula for S be correct (see section 4.8)

Therefore, we recommend to re-do the experiment and re-measure the liquid rates. We also recommend building the simulation side by side with the experiment setup. In this way, they check each other and minimize the error. It is difficult to conclude the main reason for the discrepancy at this stage. The theoretical gas slip model is widely accepted, but it is still a model. However, the reduction in numerical errors like reducing numerical diffusion and making the gas bubble more concentrated did not reduce the discrepancies. However, reducing numerical should be done anything since it will significantly improve the prediction of the duration needed for bullheading a kick entirely out of the well, which again will impact recommendations for how large fluid volumes shall be pumped.

5 Implementation of a Complete Gas Slip Model

5.1 Summary

This chapter starts by replicating a simulation case presented in [7]. First, we replicate the simulation and investigate the effect of reducing the numerical diffusion by using 25 cells, 50 cells, and 100 cells, similar to what was discussed in section 4.4. From here, we determine the optimum number of cells for the simulation. Then we test the critical bullheading rate formula derived in section 4.7.1. After that, we will implement the complete gas slip relation introduced in section 3.1.3.4. Finally, we will use the complete slip model to investigate the effect of kick size and the effect of suspension separately on bullheading.

5.2 Replication of Previous Simulation

5.2.1 Case Definition

The case used throughout this chapter was presented in [7], where the simulation was performed using the simple slip model for fixed slug flow regime parameters (K = 1.2 upward and K =1.12 for bullheading, S = 0.55). Therefore, we shall replicate the simulation in this section to test our code and show the old model's limitations.

The simulation case is for a vertical well with a total depth of 2000m. Bullheading is assumed to be performed through the annulus between 8 ½" casing/open hole and 5" drillpipe string from surface to total depth. The kick is taken at 2000m with an injection rate over a specific time, as shown in Figure 5-1. In the graph, XX is the gas mass rate which is 8.0 kg/s for the case in [7]. This chapter will test two more kick sizes with injection rates of 1.0 and 6.0 kg/s.

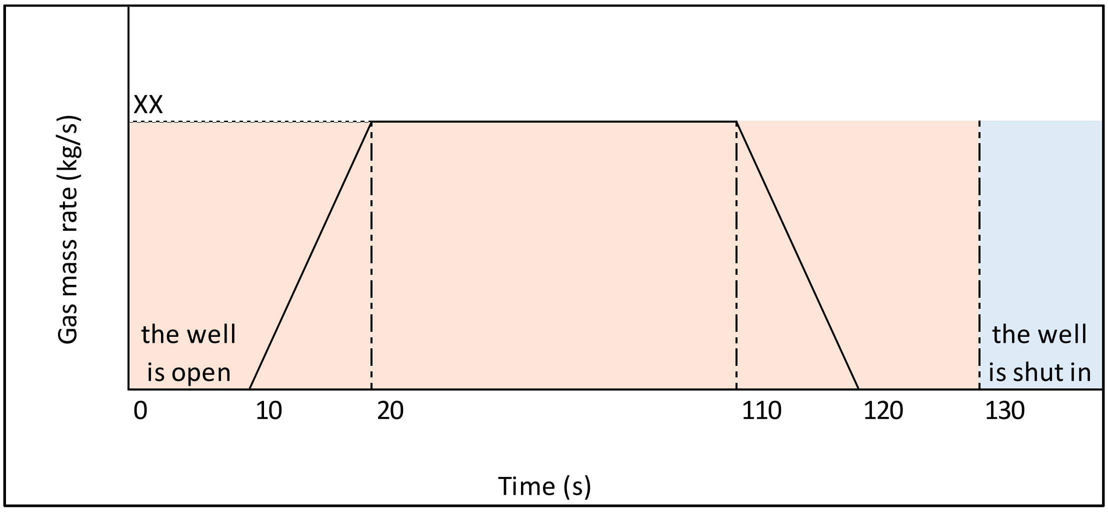


Figure 5-1: Injection of Gas in The Simulation

The well is shut in until 1500s before it is opened, and the pump is linearly ramped up from 0.00% at 1500s to 100% pumping rate at 1560s. Four pumping rates are simulated as summarized in Table 5-1. The CFL was maintained constant at 0.1875, and the number of cells and time steps were adjusted accordingly.

Table 5-1: Pumping Rates, Taken from [7] With Some Modification

Pumping Rate, kg/s	**Pumping Rate l/min**
33.33	= 33.33(kg/s)/1000(kg/ m^3) x 1000 x 60 = 2000
25	= 1500
13	= 780
8.33	= 500

We shall use the simple gas and the simple liquid density models discussed in sections 3.1.2.2 and 3.1.2.3. One can introduce the kick size in cubic meters by dividing the kick mass by the density. Assume that the bottomhole pressure is 9.81(m/s^2) x 1000(kg/ m^3) x 2000(mTVD) = 196.2bar, the gas density at this pressure is 196.2 x 10^5/316^2 = 196.5 kg/m^3 from equation 10. Therefore, the three cases that will be used in this chapter can be introduced as follows:

For 8.0 kg/s – total gas mass = 800 kg and 4.07 m^3 (we show it as 4.0 m^3 case)

For 1.0 kg/s – total gas mass = 100 kg and 0.50 m^3

For 6.0 kg/s – total gas mass = 600 kg and 3.05 m^3 (we show it as 3.0 m^3 case)

Figure 5-2 shows the gas volume versus time for the four pumping rates. One can observe that the critical bullheading rate for this case lies between 500 l/min and 780 l/min. The two largest pumping rates were able to completely push the kick out of the well within the simulation time with only 2800s for 2000l/min vs 3750s for 1500 l/min.

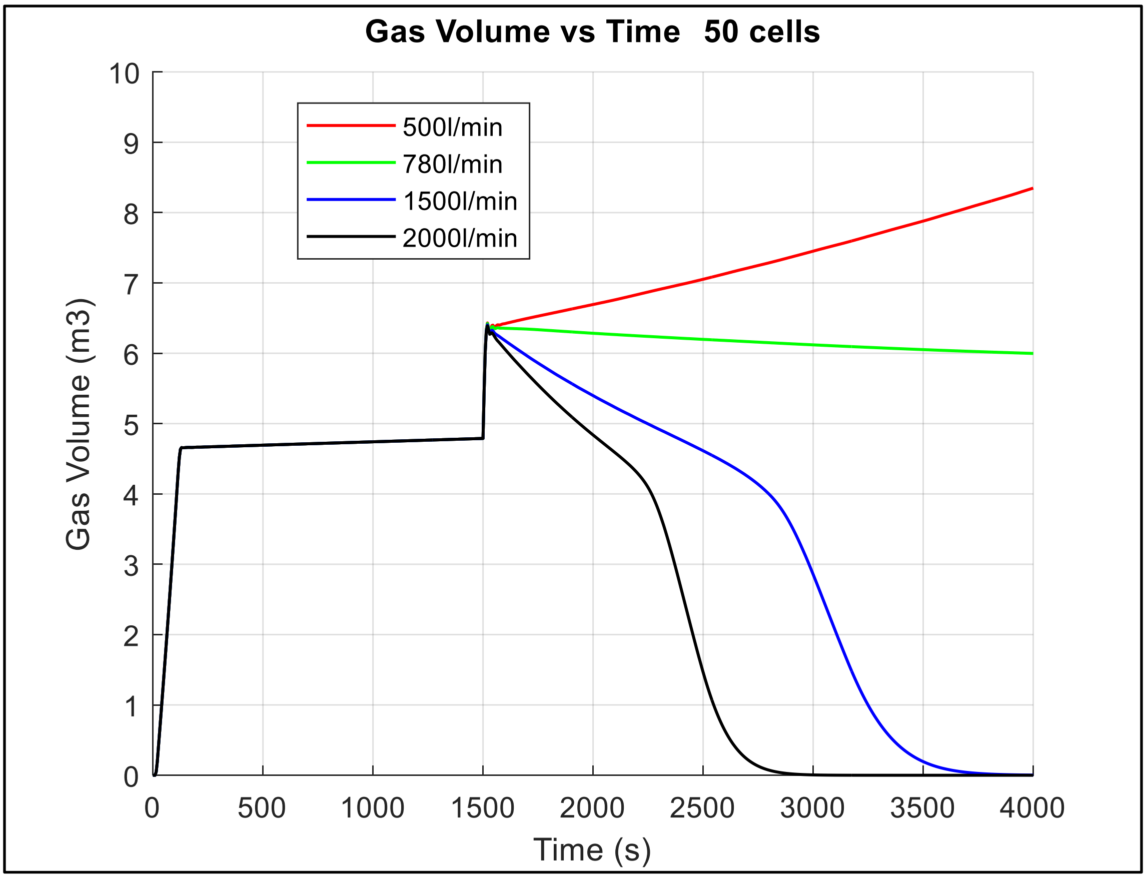


Figure 5-2: Gas Volume vs Time - Replication Case

This volume–time graph is important for selecting the optimum bullheading rate and determining the duration of the job.

5.2.2 Effect of Reducing the Numerical Diffusion

We continue with a bullheading rate 1500 l/min to study the effect of reducing the numerical diffusion. We shall vary the number of cells between 25, 50, and 100 but maintain a constant CFL of 0.1875.

Figure 5-3 through Figure 5-5 show the effect of varying the number of cells between 25, 50, and 100 cells. As shown in Figure 5-3, reducing the numerical diffusion makes the front and tail of the kick sharper. As a result, the kick started to leave the well later but was pushed out of the well completely at an earlier time in the less diffusive case than in the high diffusive one as can be seen from Figure 5-4 and Figure 5-5.

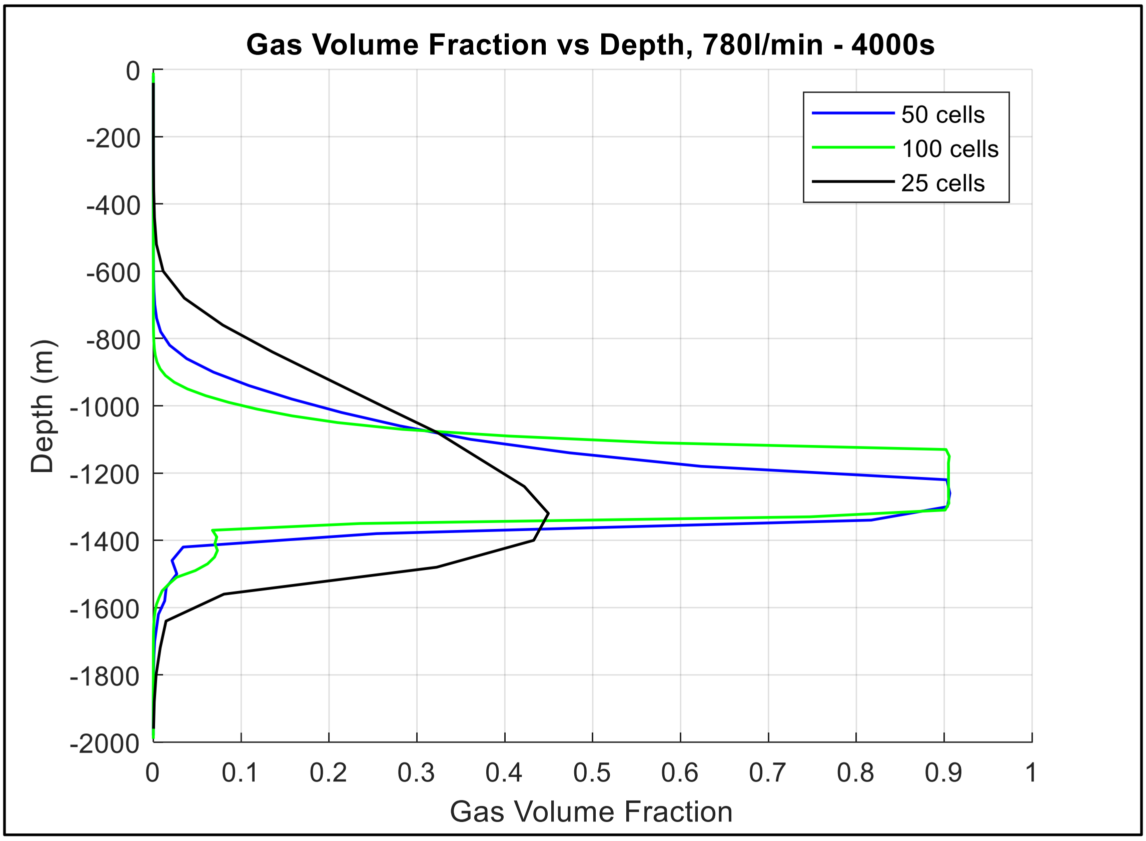


Figure 5-3: Gas Volume Fraction vs Depth, Effect of Grid Refinement

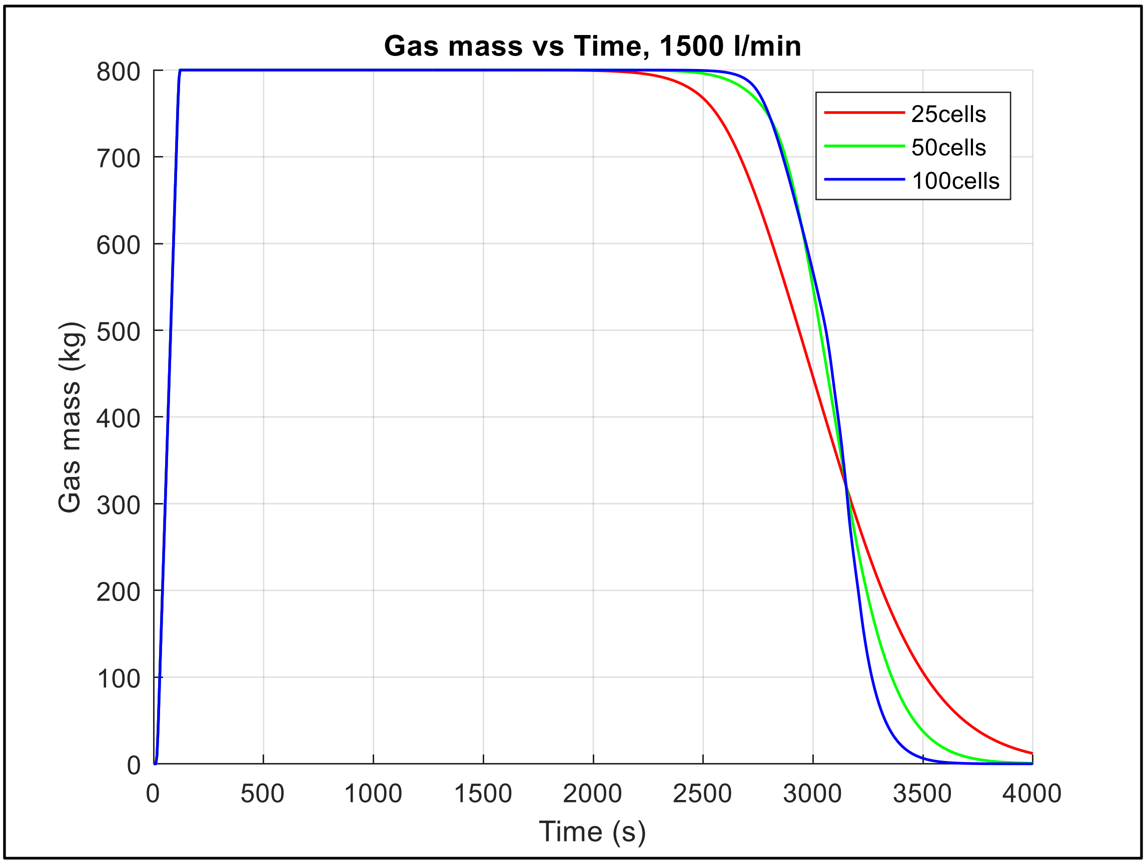


Figure 5-4: Gas Mass vs Time, Effect of Grid Refinement

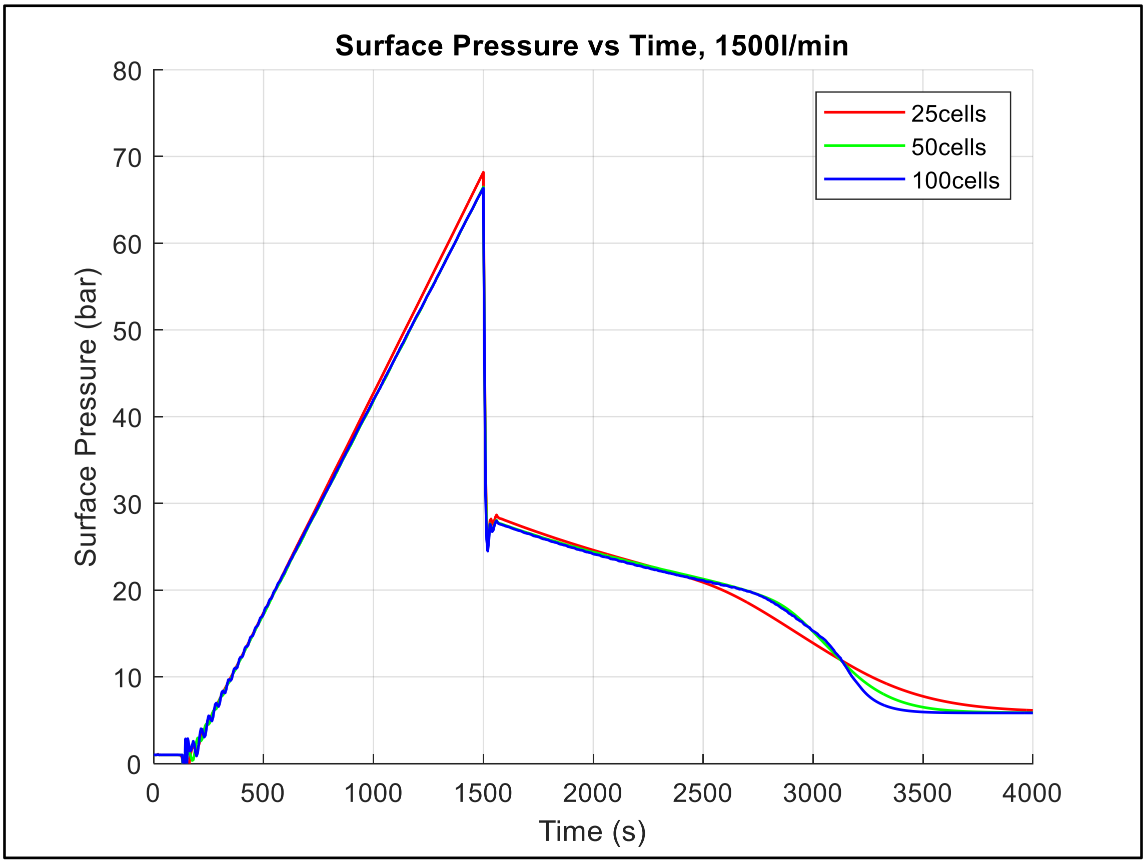


Figure 5-5: Surface Pressure vs Time, Effect of Grid Refinement

One can observe that the improvement is more minor when going from 50 to 100 cells than when going from 25 to 50 cells. Further refinement from 100 to 200 cells will improve the results, but the improvement will become even less. Moreover, the computation time is becoming 4 times longer each time the number of boxes is doubled. Therefore, we have considered 50 cells as sufficient for the simulation of this case and used it in the remaining simulations.

5.2.3 Critical Bullheading Rate

As discussed in section 4.7.1, the critical bullheading rate can be estimated from equation 60. One needs to overcome this rate to have a successful displacement. We use S = 0.55 m/s for slug flow.

$$Q_{liq} = -S\frac{A}{K} = -0.55\frac{0.02393}{K} \qquad 60$$

$$Q_{liq} = 705\ lpm\ for\ K = 1.12$$

This pumping rate was used in the simulator and plotted beside the simulation results in section 5.2.1. Figure 5-6 through Figure 5-9 show the results.

As shown in Figure 5-6, the total mass of the gas in the well remains unchanged for all the cases except the two with the largest pumping rate. In the 1500 l/m and 2000 l/min cases, the total gas mass is reduced to zero at about 2800s and 3750s, respectively, meaning the kick was completely removed from the well.

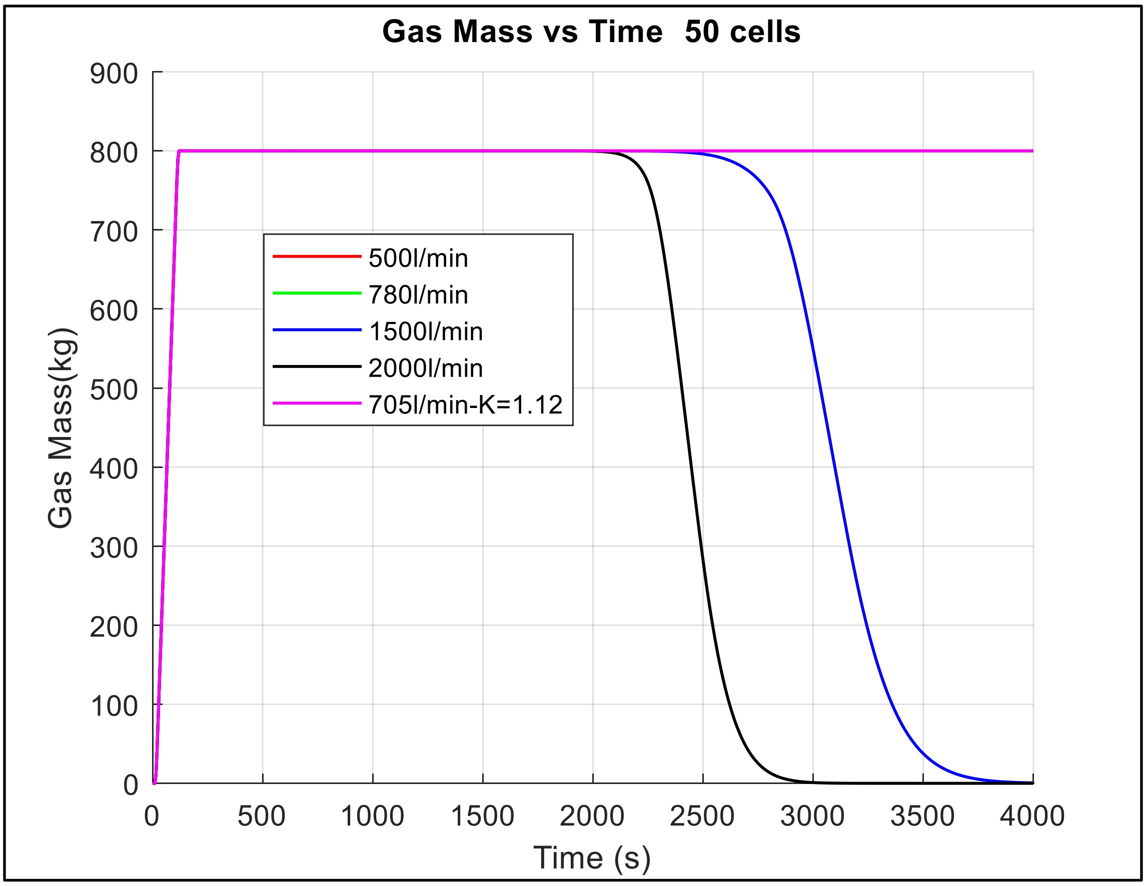


Figure 5-6: Gas Mass vs Time, Investigating Critical Bullheading Rate

From Figure 5-7, one observes that with a 500 l/min pumping rate, the gas volume increases with time after the pump starts, which indicates that the gas continues to move up the well while bullheading. This upward movement is confirmed with a positive velocity in Figure 5-8. The 780 l/min – case shows a slight decrease in the total gas volume with time, indicating a successful bullheading process to some degree where one expects to see a complete removal of the kick after a very long time.

The 705 l/min pumping rate is close to the critical bullheading rate. The kick mass and volume remain constant, and the kick is stationary with almost zero velocity from Figure 5-8. Unlike the other cases, the kick position does not change with time, as shown in Figure 5-9. In conclusion, the formula derived in section 4.7.1 and provided in equation 60 accurately predicts the critical bullheading rate.

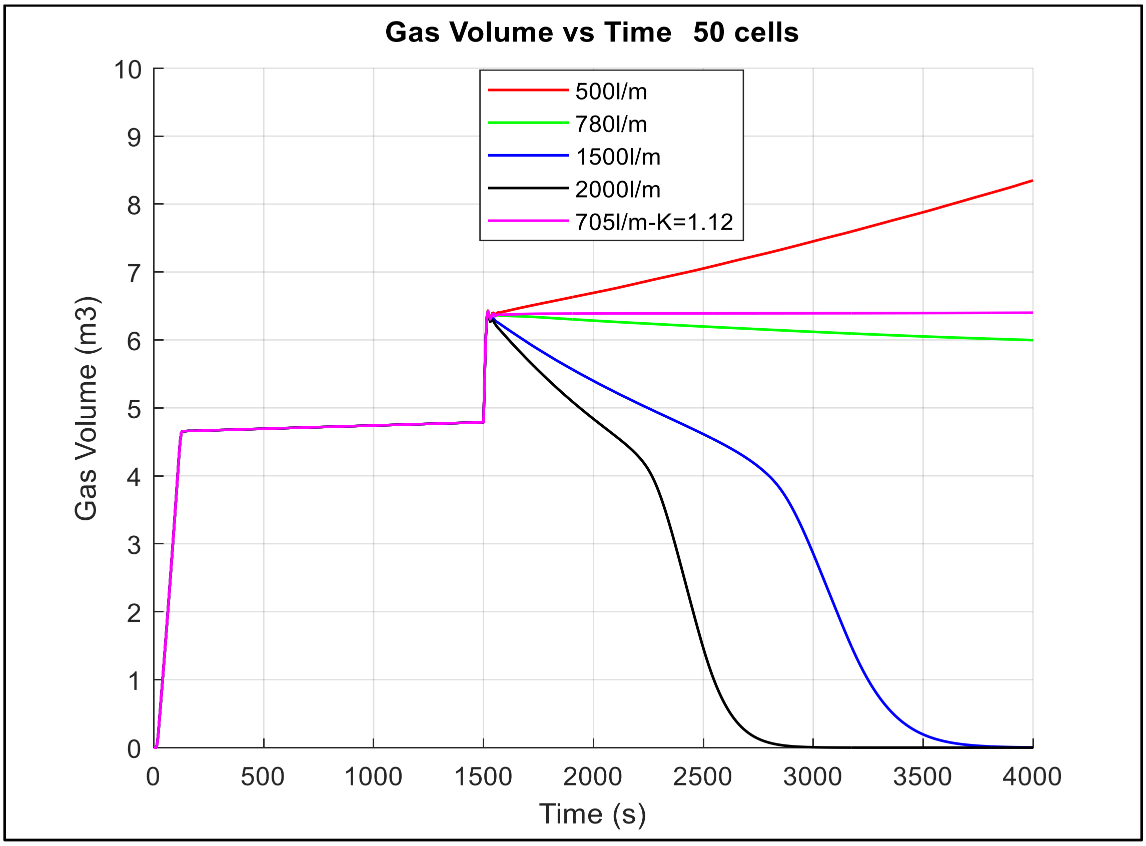


Figure 5-7: Gas Volume vs Time, Investigating Critical Bullheading Rate

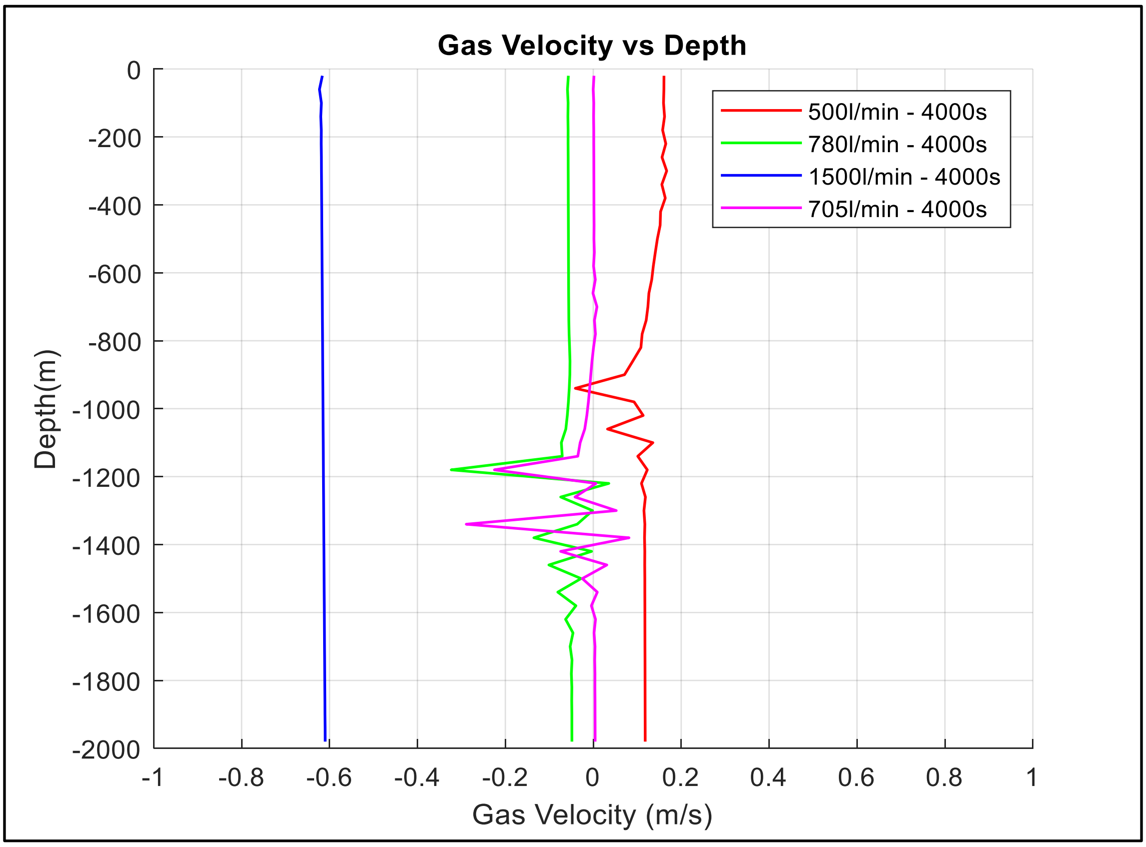


Figure 5-8: Gas Velocity vs Depth, Investigating Critical Bullheading Rate

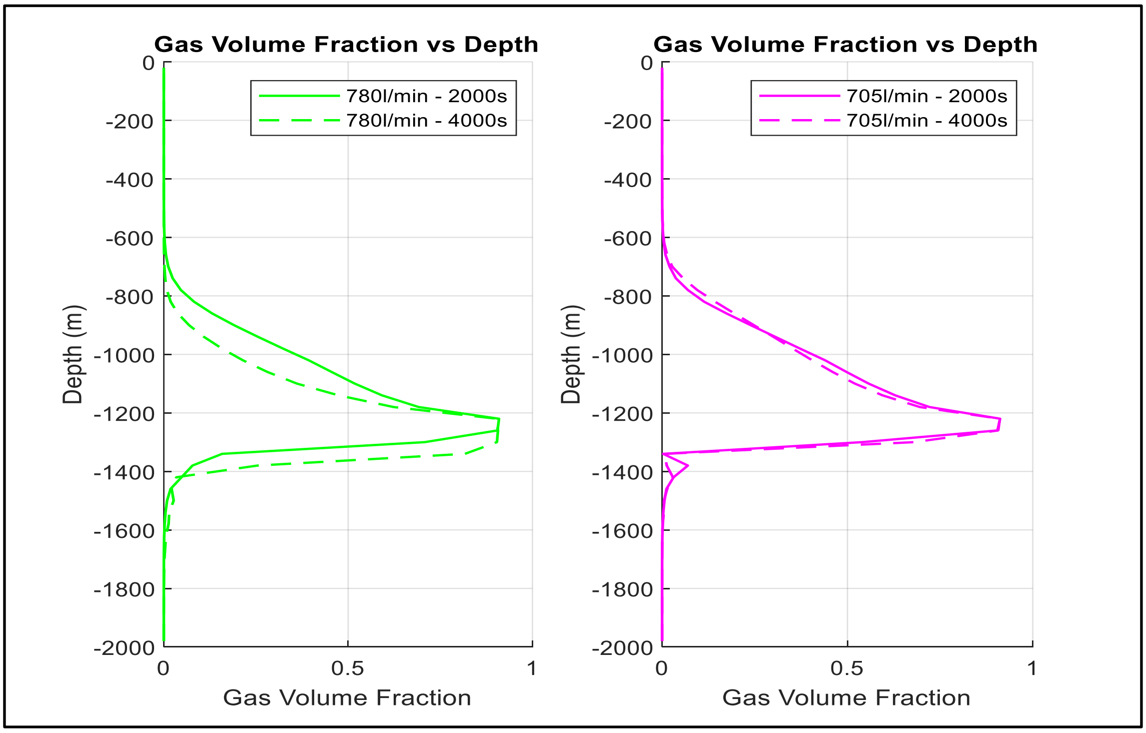


Figure 5-9: Gas Volume Fraction vs Depth, Investigating Critical Bullheading Rate

Figure 5-10 shows the accumulated liquid pumped during bullheading for each case. One can observe that for the cases where bullheading was successful, more than 1 annulus volume has been pumped. The large pumping rates complete the process at shorter time.

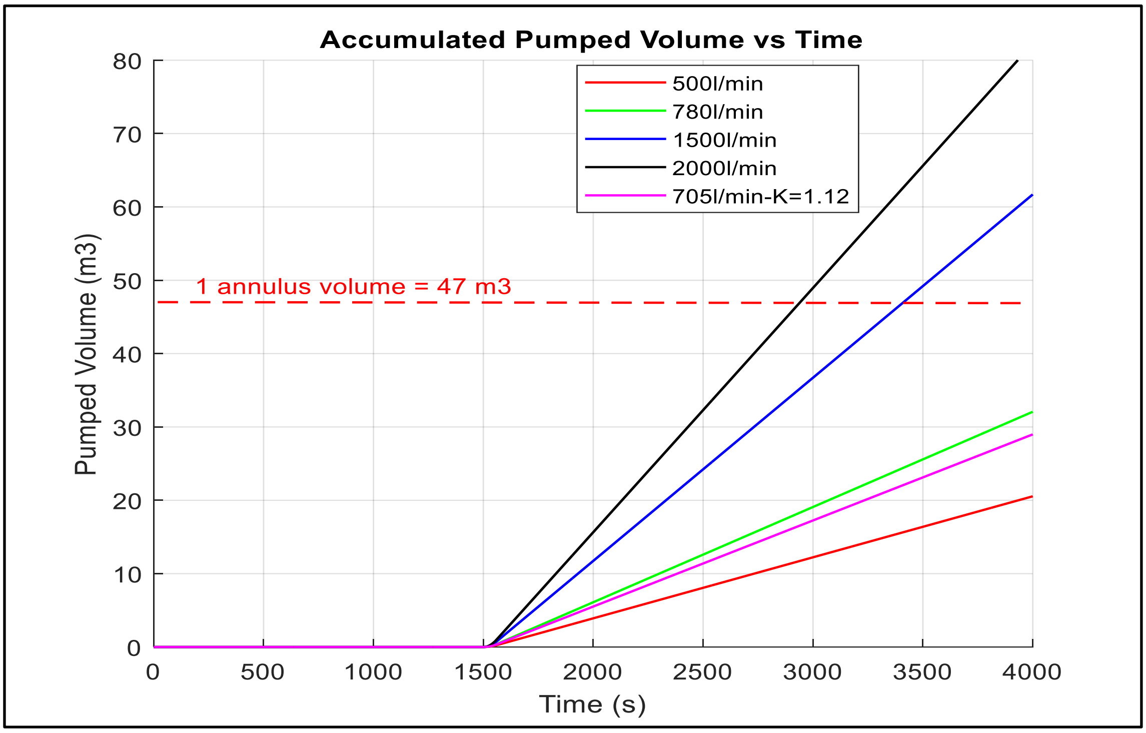


Figure 5-10: Pumped Volume vs Time, Investigating Critical Bullheading Rate

Finally, we used the critical bullheading rate and simulated the cases discussed in section 5.2.2. Figure 5-11 shows the liquid velocity versus depth at 3500s for 3 different number of cells cases. Since reducing the numerical diffusion increases the gas volume for the bulk of the kick, the liquid velocity becomes more negative around the bubble for the less diffusive cases. Furthermore, one can note a significant reduction in numerical diffusion when moving from 25 to 50 cells. Therefore, it is clear that 25 cells will smear out the gas region quite extensively, and the negative liquid velocity around the gas bubble during the bullheading process is far less in terms of value compared to what is seen for the 50 and 100 cell cases.

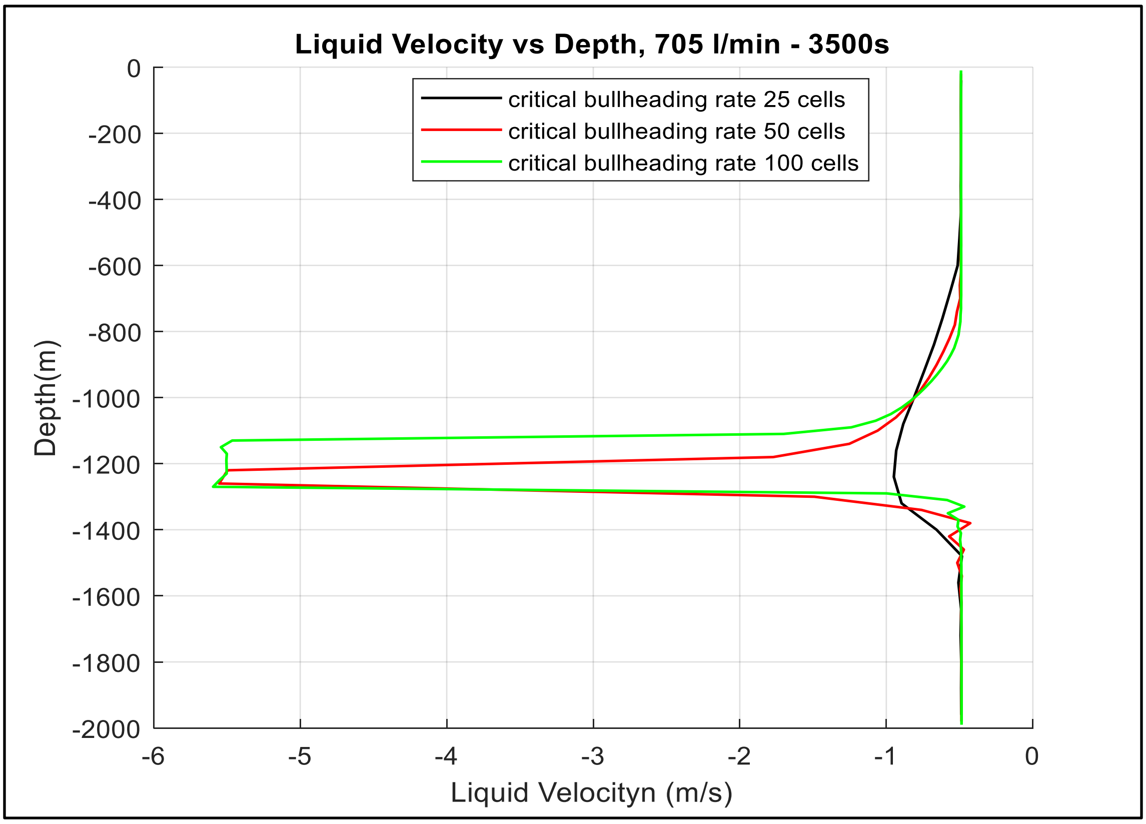


Figure 5-11: Liquid Velocity vs Depth for Different Number of Cells

5.2.4 Effect of Kick Size on Bullheading

Continuing with 50 cells and a bullheading rate of 1500 l/min, the effect of bullheading a small kick (0.5 m^3) and large kick (4.0 m^3) will be studied using the simple slip model.

Figure 5-12 shows the total gas mass versus time for the two kick sizes. As can be observed, both kicks start to leave the well at around 2505s. This observation is expected as the simple slip model is a fixed flow regime model, slug flow for our case. This results in that the gas migrates with the Taylor bubble velocity and being displaced with the same displacement velocity regardless of the kick size. Figure 5-13 shows that the displacement velocity of the two gas bubbles is the same (around (-0.6m/s)). The gas volume fraction graphs in Figure 5-13 show that the lower gas kick tail for the two cases is identical as the gas bubbles move downward.

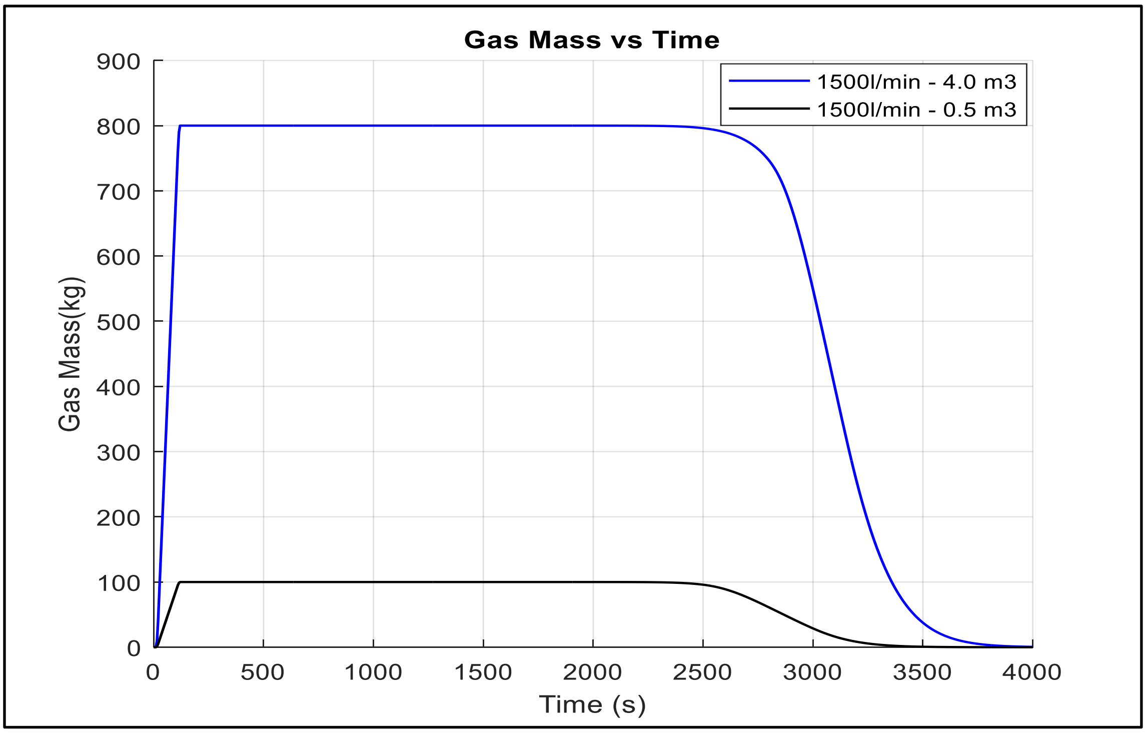


Figure 5-12: Gas Mass vs Time, Effect of Kick Size

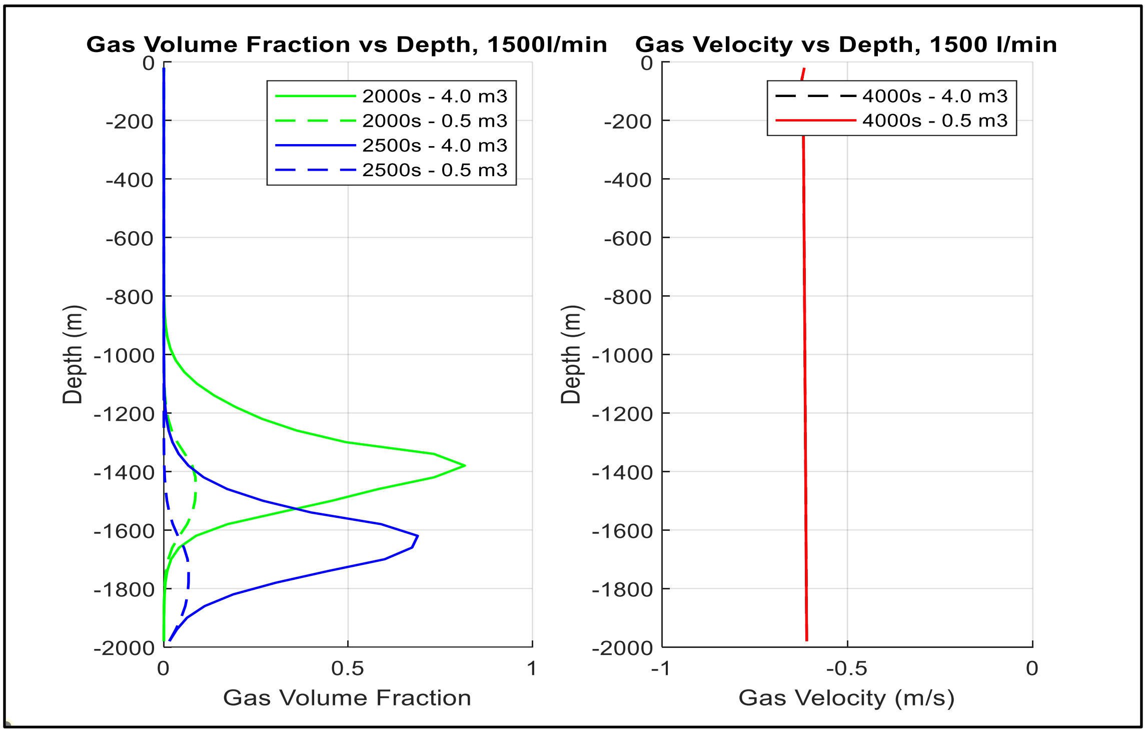


Figure 5-13: Gas Volume Fraction and Gas Velocity vs Depth, Effect of Kick Size

5.3 Implementation of Advanced Slip Model

In this section, we will implement a complete gas slip relation in the model as discussed in section 3.1.3.4 and use the model to investigate the effect of kick size and suspension on bullheading. The complete code is provided in Appendix C

5.3.1 Comparing Bullheading of two Different Kick Sizes

We will re-run the cases in section 5.2.4 using the model with the complete slip relation. We do not consider suspension here, but we would like to demonstrate the impact of having different flow patterns on the bullheading process in the well.

Figure 5-14 shows the gas volume fraction versus depth at 300s and 800s during the shut-in period. As can be seen, the small kick migrates with a slip velocity given by the bubble flow model. However, the large kick can be divided into two parts: the bulk on the top part migrates with a slug flow model, and the bottom part, which comprises trails of small, distributed bubbles, migrates with a bubble flow model. One can also observe that the large kick has been stretched out due to these velocity differences.

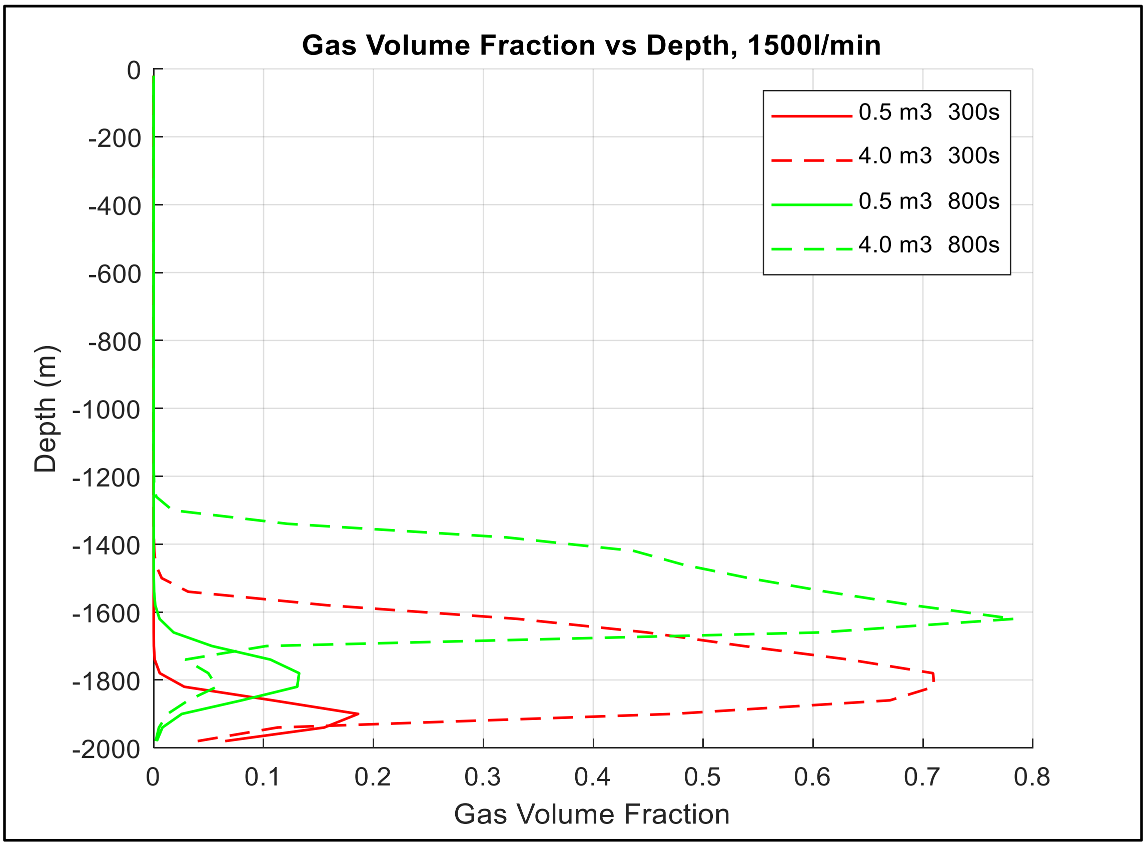


Figure 5-14: Gas Volume Fraction vs Depth, Comparing Two Kick Sizes

We use the front position of the two kicks to calculate average migration velocities, as shown in Table 5-2.

Table 5-2: Calculations of Average Kick Migration Velocities

	Depth, m		**Time, s**		**Depth, m**		**Time, s**
x1	-1780	t1	800	x1	-1620	t1	800
x2	-1900	t2	300	x2	-1820	t2	300
	120		500		200		500
0.5 m^3	vg	0.24	m/s	4.0 m^3	vg	0.4	m/s

These average velocities can be directly compared with the instantaneous gas velocities from the simulation, as shown in Figure 5-19. The large kick migrates with almost double the small kick migration velocity.

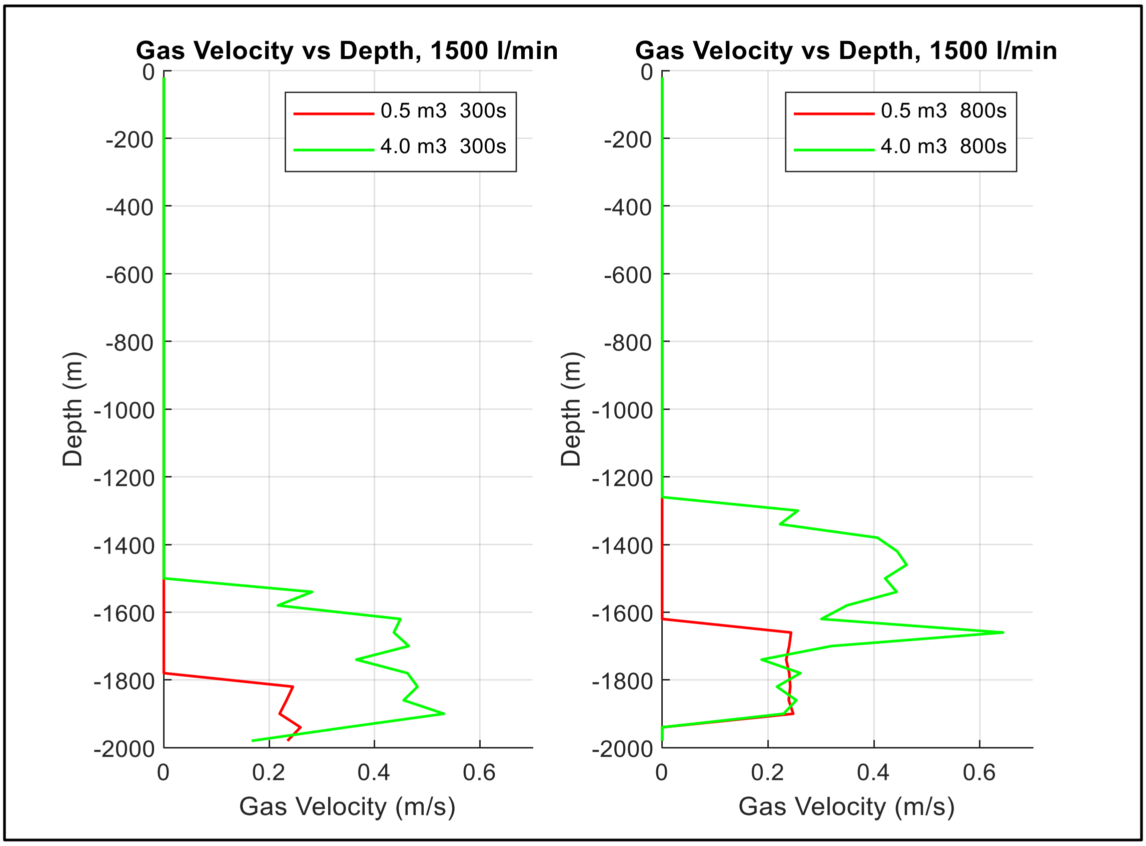


Figure 5-15: Gas Velocity vs Depth, Comparing Two Kick Sizes

Since this section aims to demonstrate the impact of having different flow patterns in the well on bullheading, we plan to have the front of both kicks at the same depth before the pump is started for bullheading.

The close-in period for the 4.0 m^3 kick was set at 2500s, and the gas volume fraction versus depth at this time was plotted in Figure 5-16. The simulation was run for the 0.5

m^3 case to 2500s, and the gas volume fraction was plotted versus depth on the same figure. We notice the difference in the fronts' depths at 2500s as (-820m) for the large kick against (-1400m) for the small kick. We use the calculated average velocity to bring the front of the small kick 580m higher. This requires shut-in period of a bout 4920s (580/0.24 = 2417 + 2500 = 4917s) for the small kick. The simulation of the small kick was extended to 4920s, and the gas volume fraction was plotted as shown in Figure 5-16. One can notice that main bulk of both kicks are now at approximately the same depth (around - 820m).

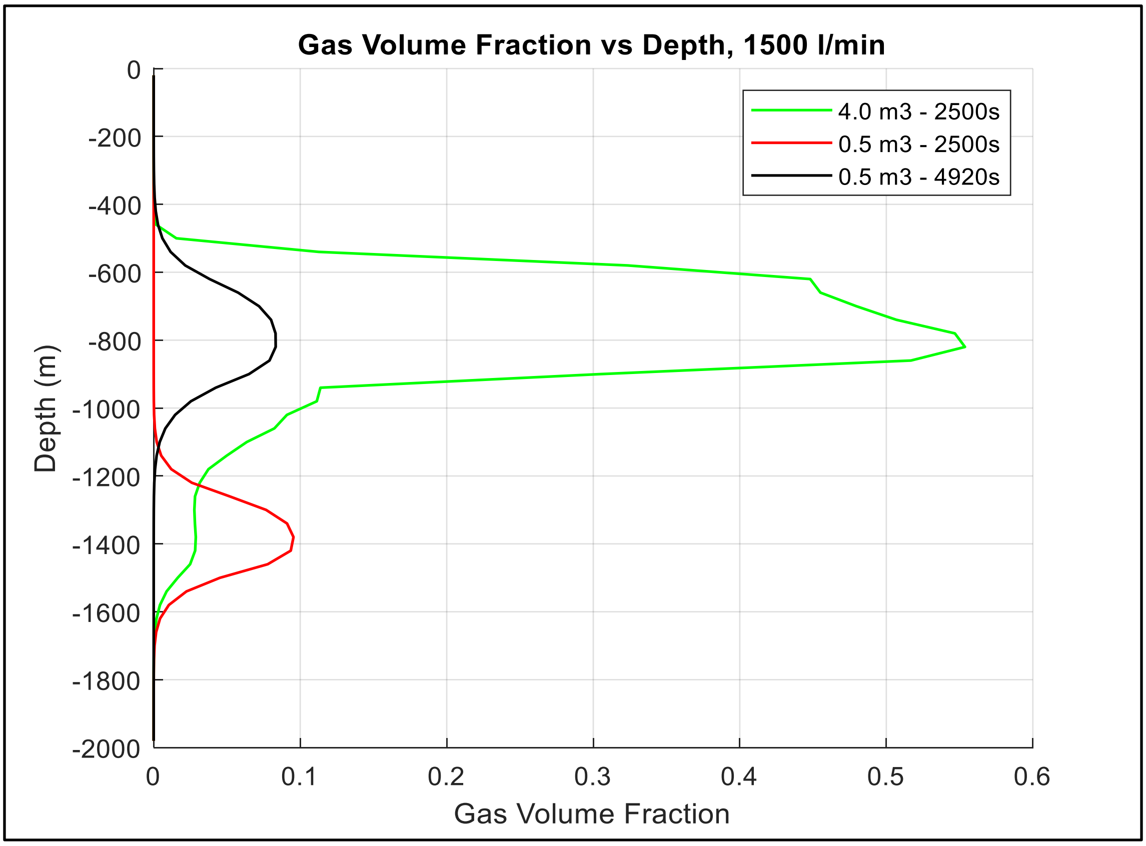


Figure 5-16: Gas Volume Fraction vs Depth, Comparing Two Kick Sizes

Figure 5-17 shows the plotted wellhead and bottomhole pressures versus shut-in time. One can notice that the bottomhole pressure has decreased during the injection period of the kick because of the reduction in the hydrostatic column with the well open to the atmosphere. Then the well is closed in, and the kick starts to migrate up the well. As the gas is not allowed to expand very much at low-pressure depths, the wellhead and bottomhole pressures build up in both cases but at different rates. The pressure build-up rate is higher for the large kick than for the small kick because of its higher migration velocity.

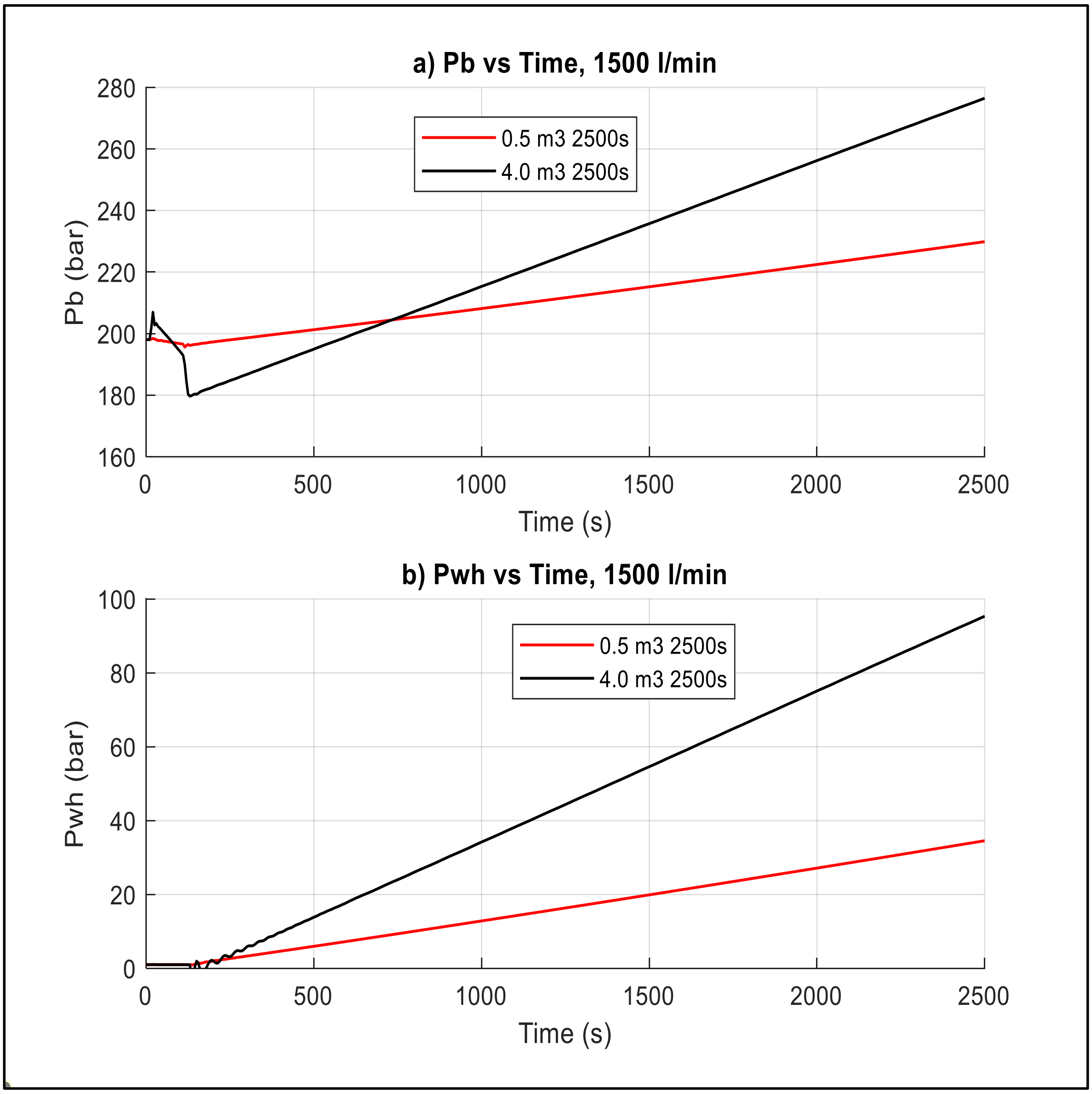


Figure 5-17: Pressure vs Depth A) Bottomhole Pressure, B) Wellhead Pressure

After that, the pump was started, and the workspace was saved at 150s, 250s, and 1000s of bullheading. Figure 5-18 shows the gas volume fraction versus depth at these times. As can be seen, the kick shrinks as it is pushed down because the pressure is higher at deeper depths. One can also observe that the gas bubble volume decreases as it moves downward because the pressure applied to it increases due to the increased hydrostatic pressure.

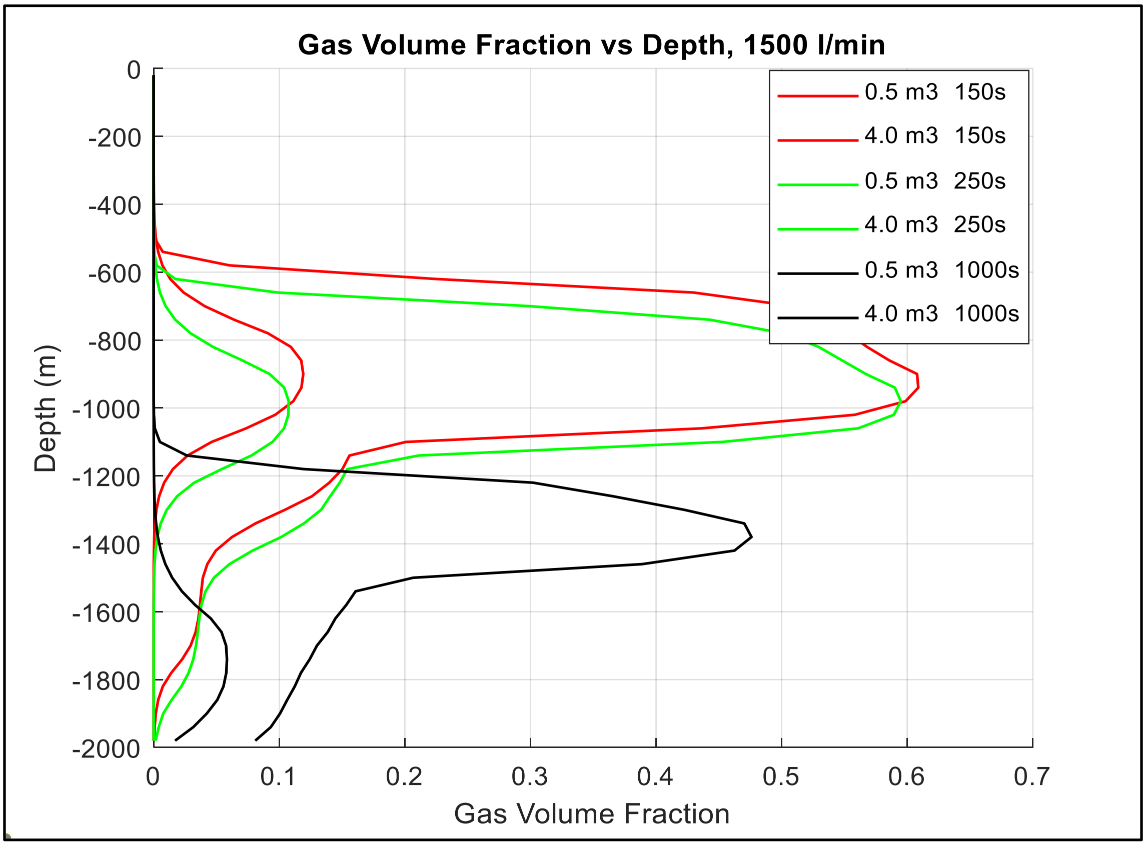


Figure 5-18: Gas Volume Fraction vs Depth, Comparing Two Kick Sizes

We use the bulk of the kick position at each time and calculate the average displacement velocities as shown in Table 5-3.

Table 5-3: Calculations of Average Gas Displacement Velocities

	Depth, m		Time, s		Depth, m		Time, s
x1	-900	t1	150	x1	-920	t1	150
x2	-1000	t2	250	x2	-980	t2	250
	100		-100		60		-100
4.0 m^3	vg	-1.0	m/s	0.5 m^3	vg	-0.6	m/s

These average velocities can be directly compared with the instantaneous velocities of gas versus depth at 150s and 250s, as shown in Figure 5-19. Here, the 0.5 m^3 kick and the bottom part of the 4.0 m^3 kick are displaced with a higher downward velocity since they are in the bubble flow regime. On the other hand, the bulk of the 4.0 m^3 kick is moved with the slowest downward velocity since it is in the slug flow regime.

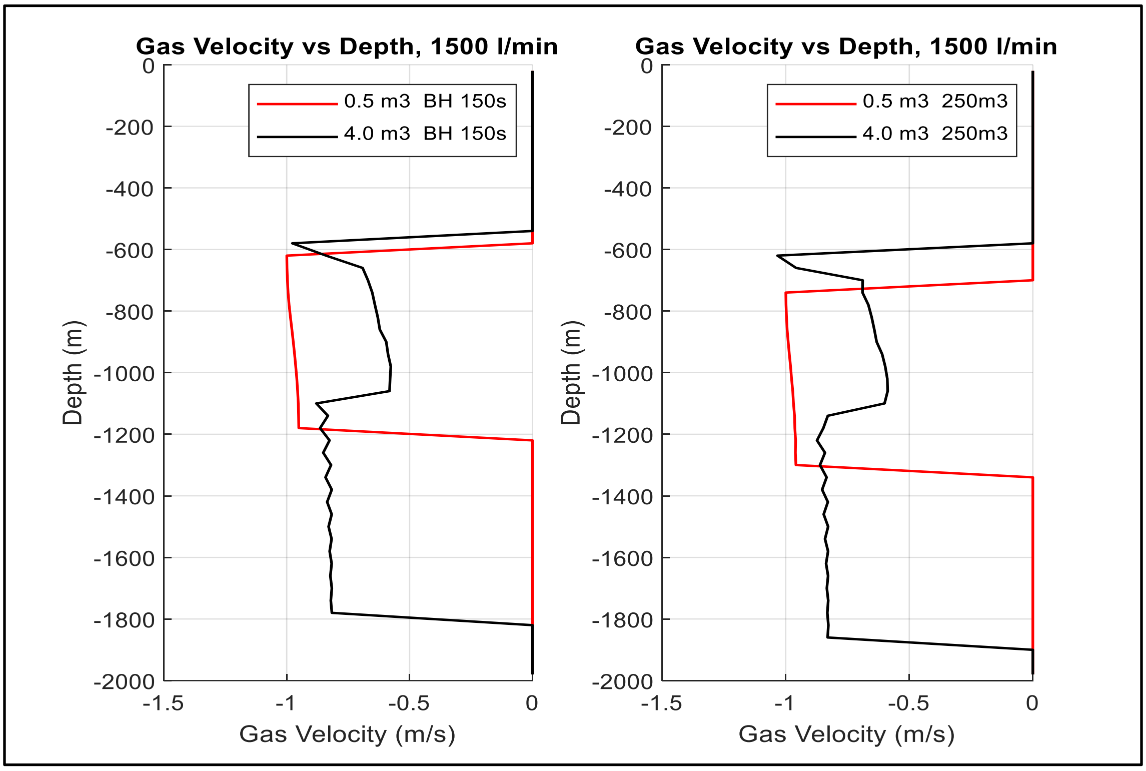


Figure 5-19: Gas Velocity vs Depth, Comparing Two Kick Sizes

Figure 5-20 shows the liquid velocities versus depth at 150s and 250s after bullheading is started. As can be seen, the liquid velocities are much larger around the large bubble because of the smaller area available for the liquid in the two-phase region.

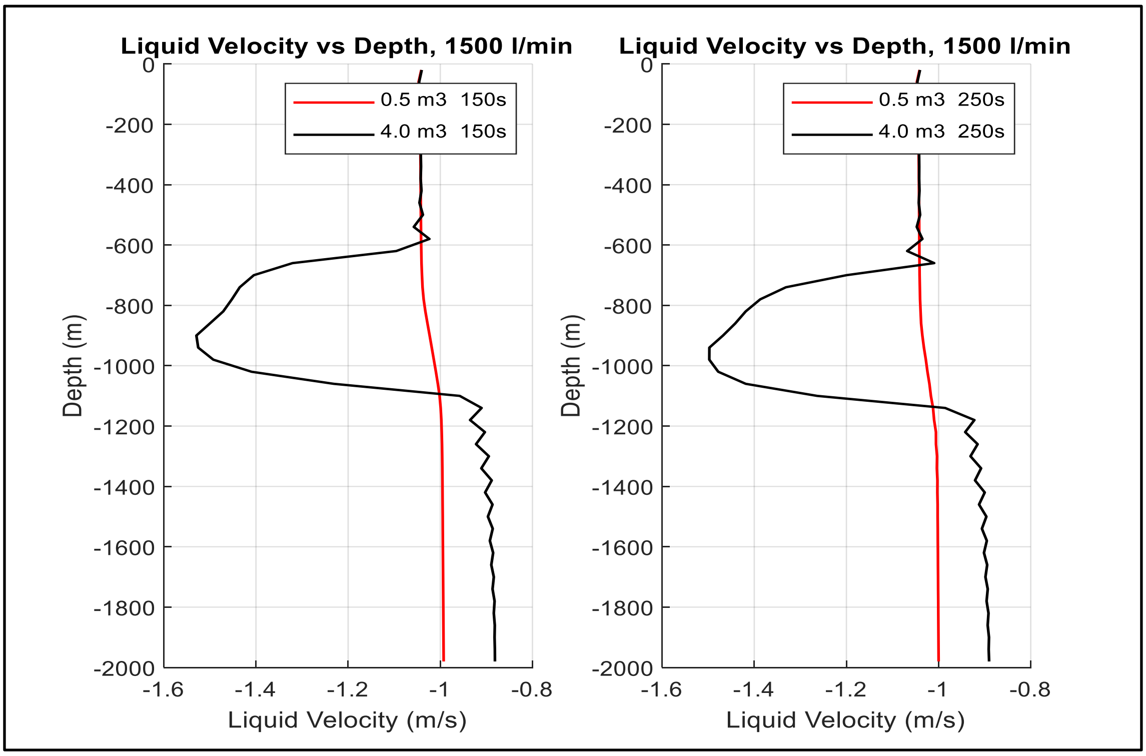


Figure 5-20: Liquid Velocity vs Depth, Comparing Two Kick Sizes

Interestingly, the liquid velocity below the kick during bullheading is lower than the liquid above it. It also turns out that the liquid velocity below the kick is lower for the larger kick than the small kick. If we inspect the liquid velocity formula ($v_l = \frac{1-\alpha_g k_o}{\alpha_l} v_{mix} - \frac{\alpha_g S_o}{\alpha_l}$) we notice that the liquid velocity in the pure liquid region below the kick is reduced to $v_l = v_{mix}$ when α_g becomes zero. Hence the reduction in vl must be due to a reduction in vmix when moving from the upper part of the well towards the lower part of the well. We plotted vmix as a function of depth in Figure 5-21 to check this, and a reduction in vmix was seen across the two-phase flow region. This reduction was larger for the large size kick.

It is believed that this reduction is due to the compression effect when the gas is moved downwards. In the opposite case, when a kick migrates in an open well with no pumping, one can observe that the mixture velocity above the kick is larger than the liquid velocity below the kick. This is due to an expansion effect. So, it may be logical that the opposite will be the case when we are bullheading gas downwards

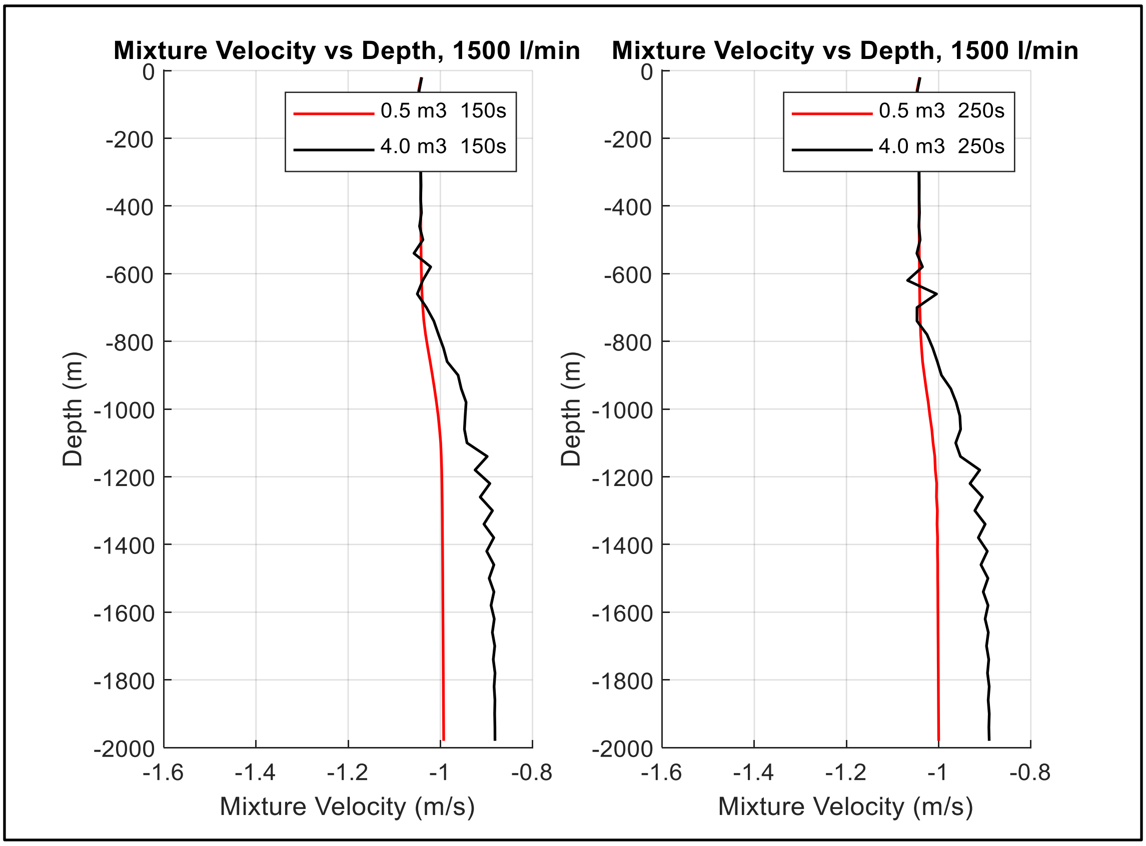


Figure 5-21: Mixture Velocity vs Depth, Comparing Two Kick Sizes

Figure 5-22 and Figure 5-23 show the total gas mass and the total gas volume in the well versus time, respectively. As can be seen, the small is bullheaded much faster than the large kick with 1500s versus 2250s.

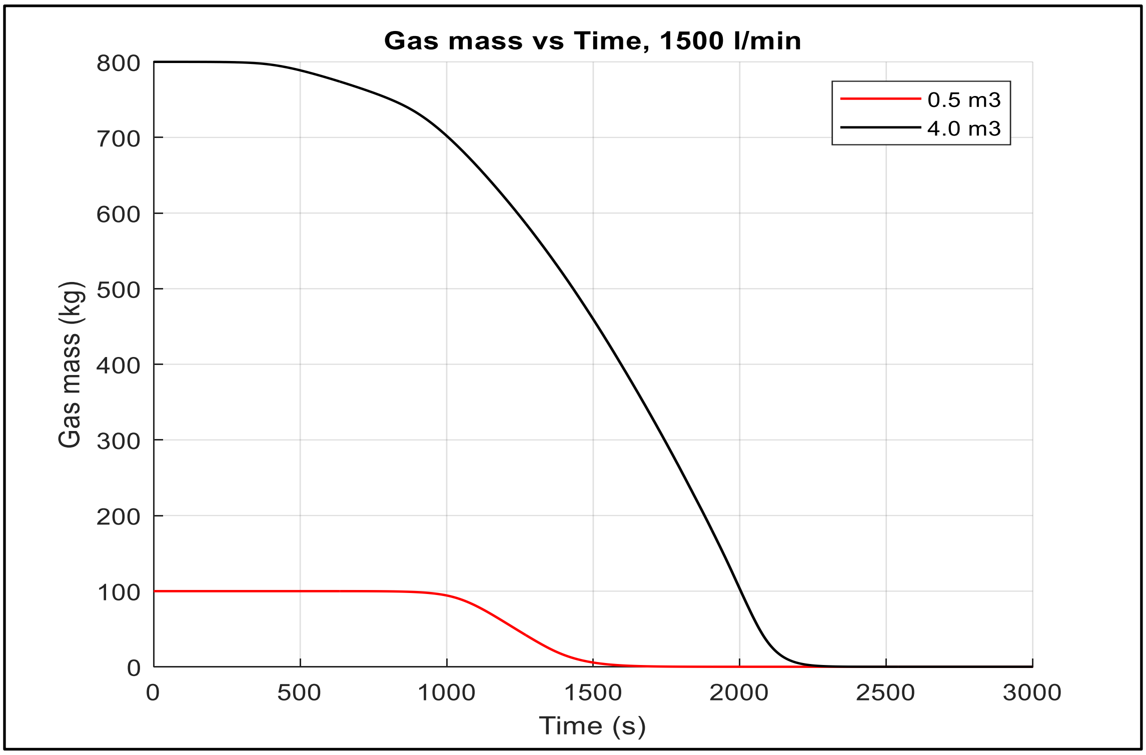


Figure 5-22: Gas Mass vs Time, Comparing Two Kick Sizes

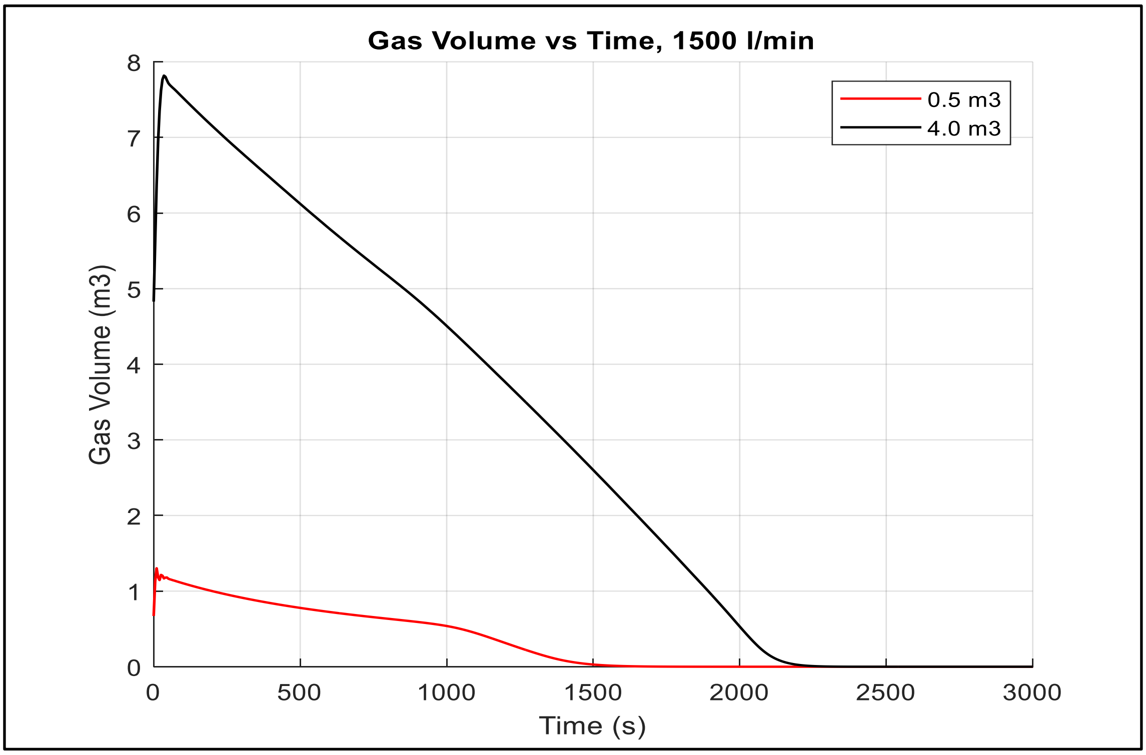


Figure 5-23: Gas Volume vs Time, Comparing Two Kick Sizes

Figure 5-24 shows the wellhead pressure versus time for two kick sizes. After that, the wellhead pressure decreases with time in the two cases because the hydrostatic component increases indicating a successful bullheading operation. When all the gas is removed from the well, the pressure stabilizes at the pump pressure. When the pump stops, one expects the final pressure to become zero.

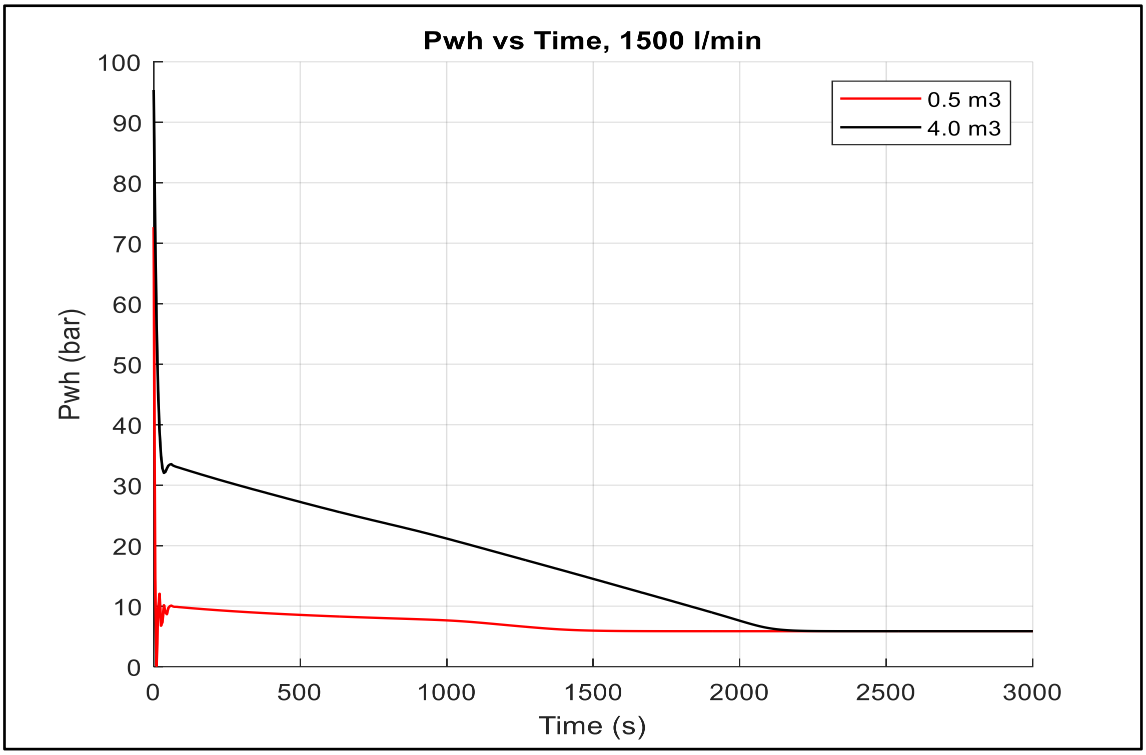


Figure 5-24: Wellhead Pressure vs Time, Comparing Two Kick Sizes

In conclusion, it is easier to bullhead a small kick than a large one. The small kick is displaced with a larger downward velocity than the large size kick. The wellhead pressure is much lower in the small kick case.

5.3.2 Effect of Suspension on Bulleading

In this section, we study the effect of suspension on bullheading a 6.0 m^3 kick. We use two cases: one with 0.00% suspension and another with 10% suspension. The shut-in period is 3500s, and the bullheading rate is 1500 l/min.

We have assumed a high suspension percentage to demonstrate what can happen with respect to the flow regime transitions. The assumed percentage is probably higher than what would be seen in practice, but at the same time, a deep well would have given the same effect for a lower suspension effect. The amount of gas left behind depends on how severe the suspension effect is but also how far the kick has been allowed to migrate.

Figure 5-25 shows the gas volume fraction versus depth at different times during the shut-in period. As can be seen, both cases start with a slug flow regime. As the gas migrates up the well, the trail of the gas behind the bulk of the kick increases. As a result, the gas volume fraction in the bulk of the kick reduces, leading to a transition in the flow pattern. In the 10% suspension case, the bulk of the kick transitioned to a bubble flow regime at 3500s. The 0% suspension case migrates with the slug flow model all the time. From Figure 5-26 which shows the gas velocities versus depth at different times, one can observe that the 10% case has a kick migration velocity of only 0.25 m/s at 3500s versus 0.49 m/s for the 0% suspension case.

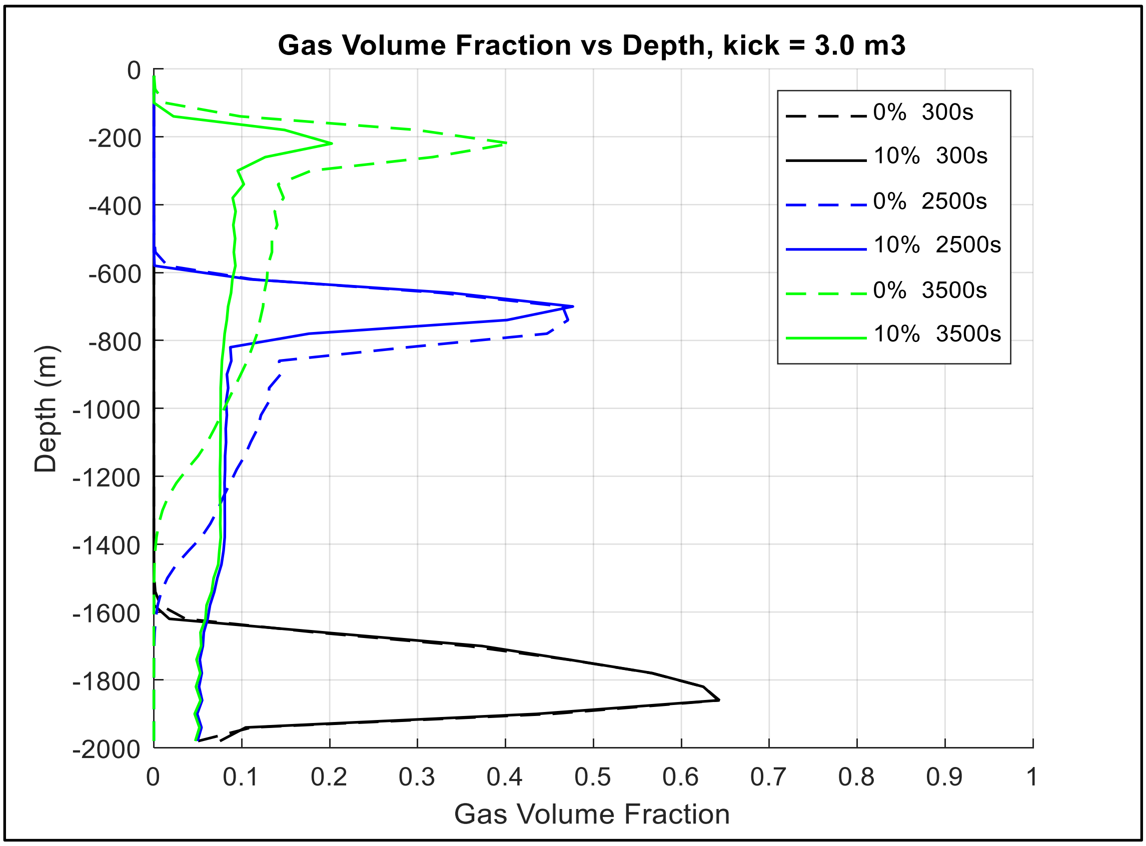


Figure 5-25: Gas Volume Fraction vs Depth, Effect of Suspension

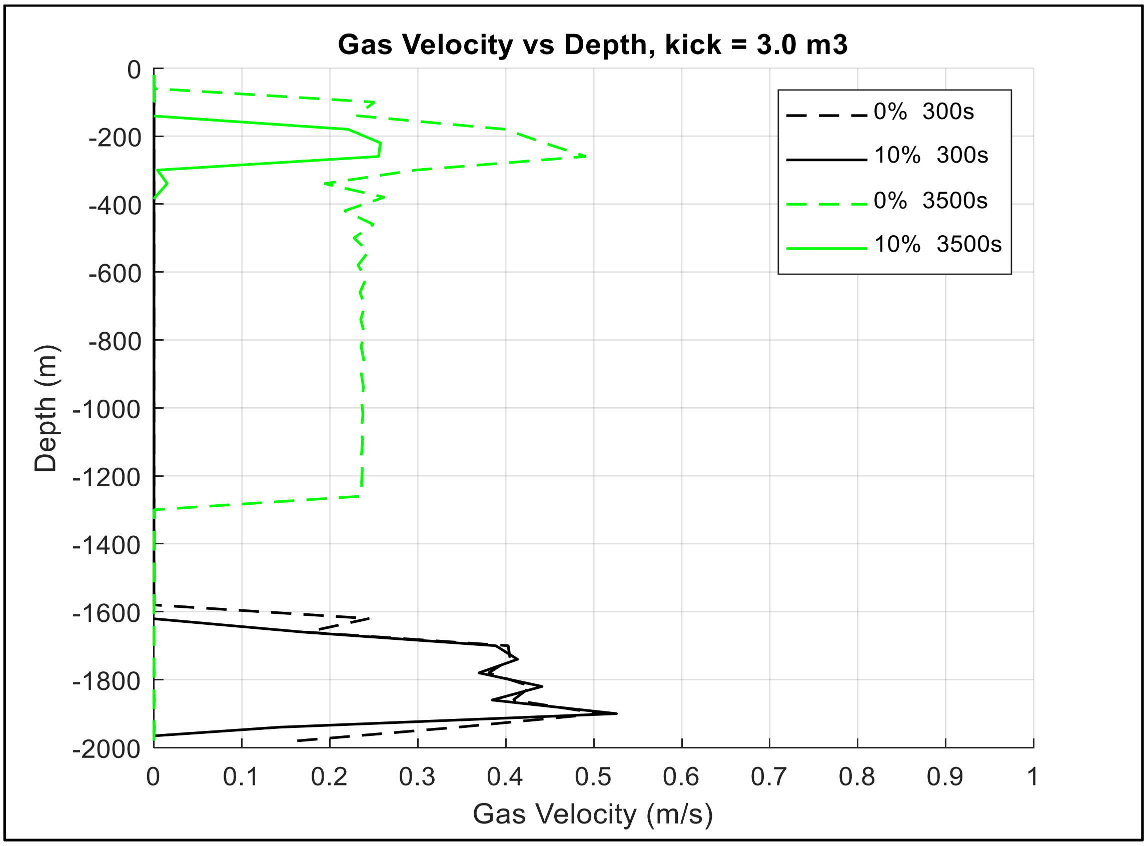


Figure 5-26: Gas Velocity vs Depth, Effect of Suspension

Figure 5-27 and Figure 5-28 show the gas volume fraction and gas velocity versus depth at different times during bullheading. As can be observed, the 10% suspension case is pushed downward with a larger displacement velocity than the 0% suspension case. This is because for the 10% case, there is a tail of gas all the way to the bottom; meanwhile, the front of the kick for the 0% suspension is in the slug flow regime.

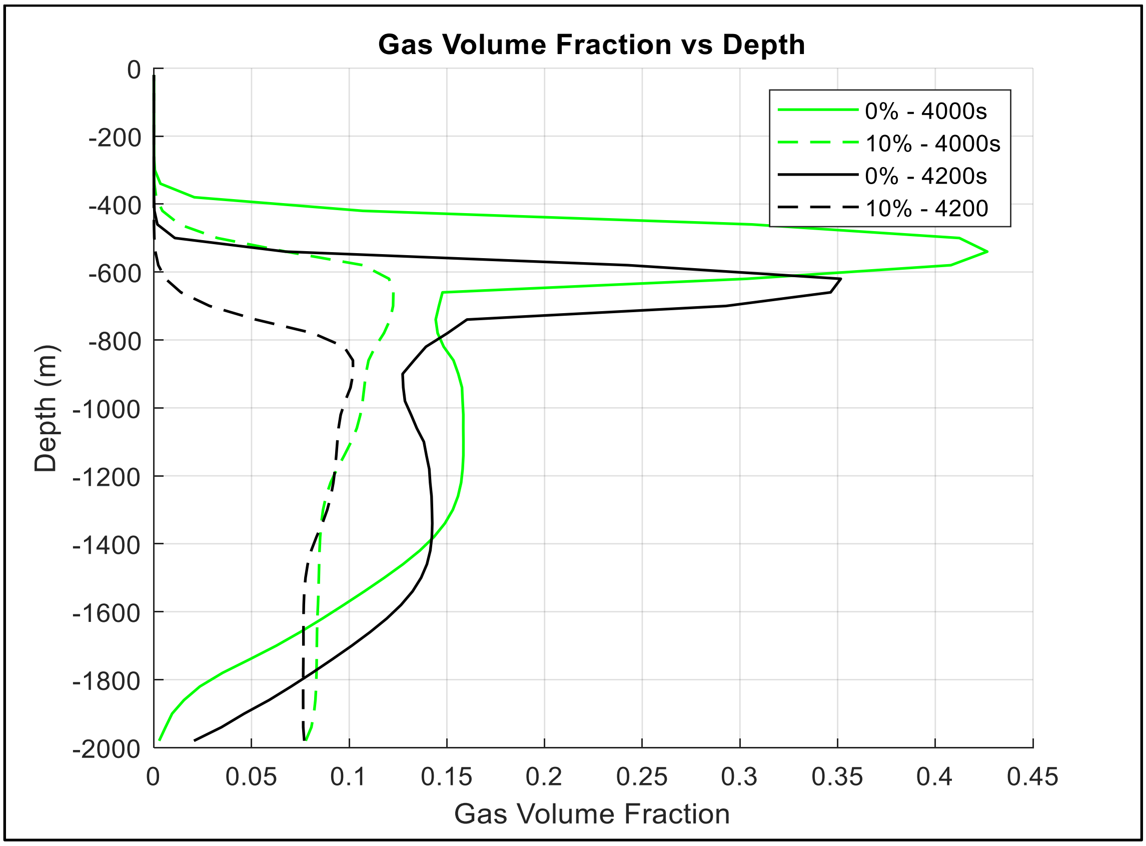


Figure 5-27: Gas Volume Fraction vs Depth During Bullheading, Effect of Suspension

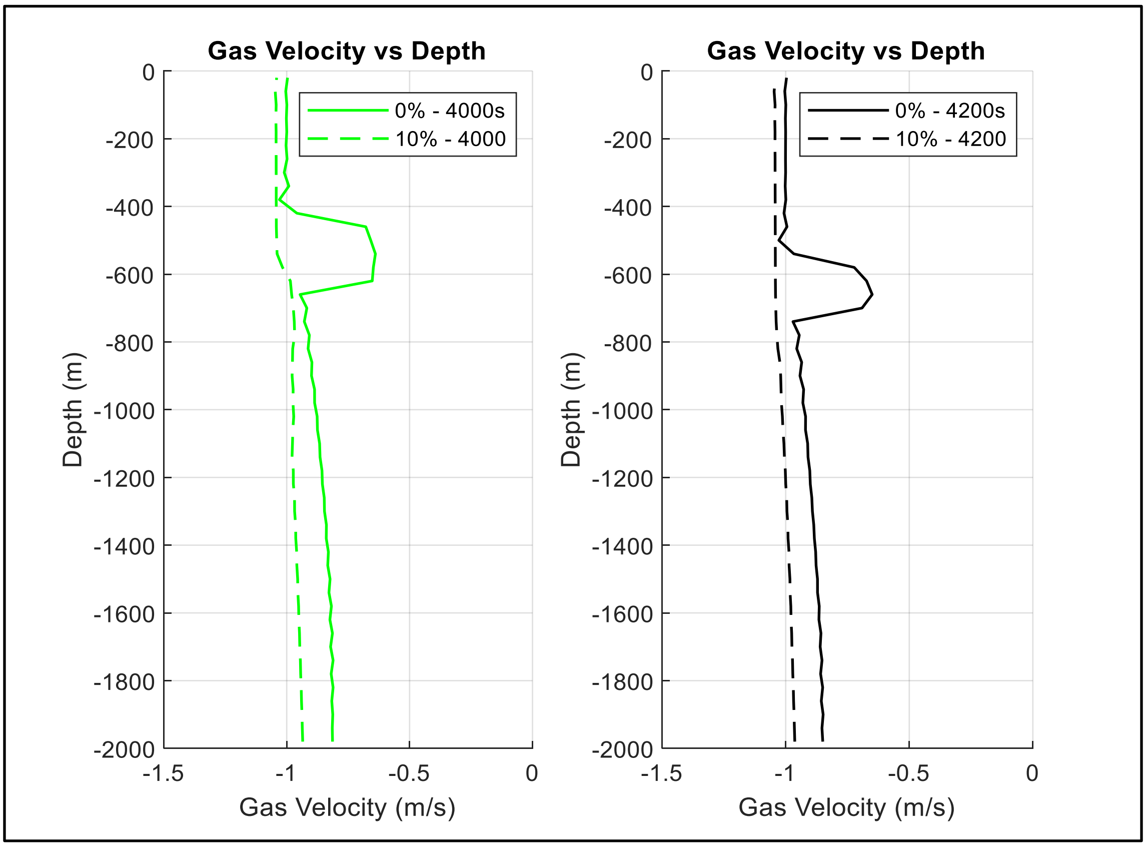


Figure 5-28: Gas Velocity vs Depth During Bullheading, Effect of Suspension

Figure 5-29 and Figure 5-30 show the wellhead and bottomhole pressures versus time for the two cases. As can be seen, the wellhead pressure build-up during kick migration for closed-in conditions is largest for the case where no suspension takes place. When having suspension, a trail of gas bubbles will be left behind, which will change the overall liquid compressibility, resulting in consequences for the pressure build-up. The result is that the pressure build-up will be lower when having suspension. The suspension effect will be that the volume of gas that still migrates reduces with time. The kick stops migrating and becomes entirely trapped in the well in some cases. In these cases, the pressure build-up will stop earlier than expected. During the bullheading period, the wellhead pressure decreases during bullheading for the two cases indicating an increase in the hydrostatic column and a successful bullheading operation; the bottomhole pressure (the injection pressure) is maintained constant at 200bar.

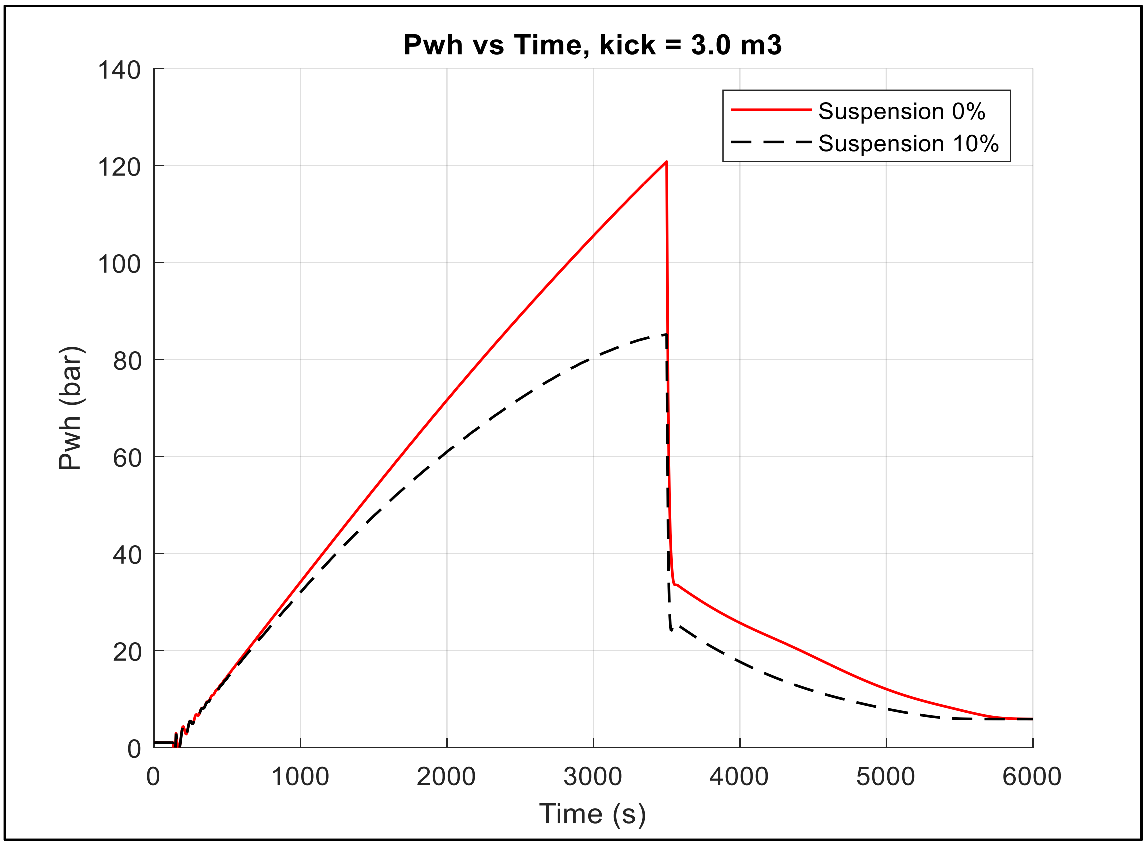


Figure 5-29: Wellhead Pressure vs Time, Effect of Suspension

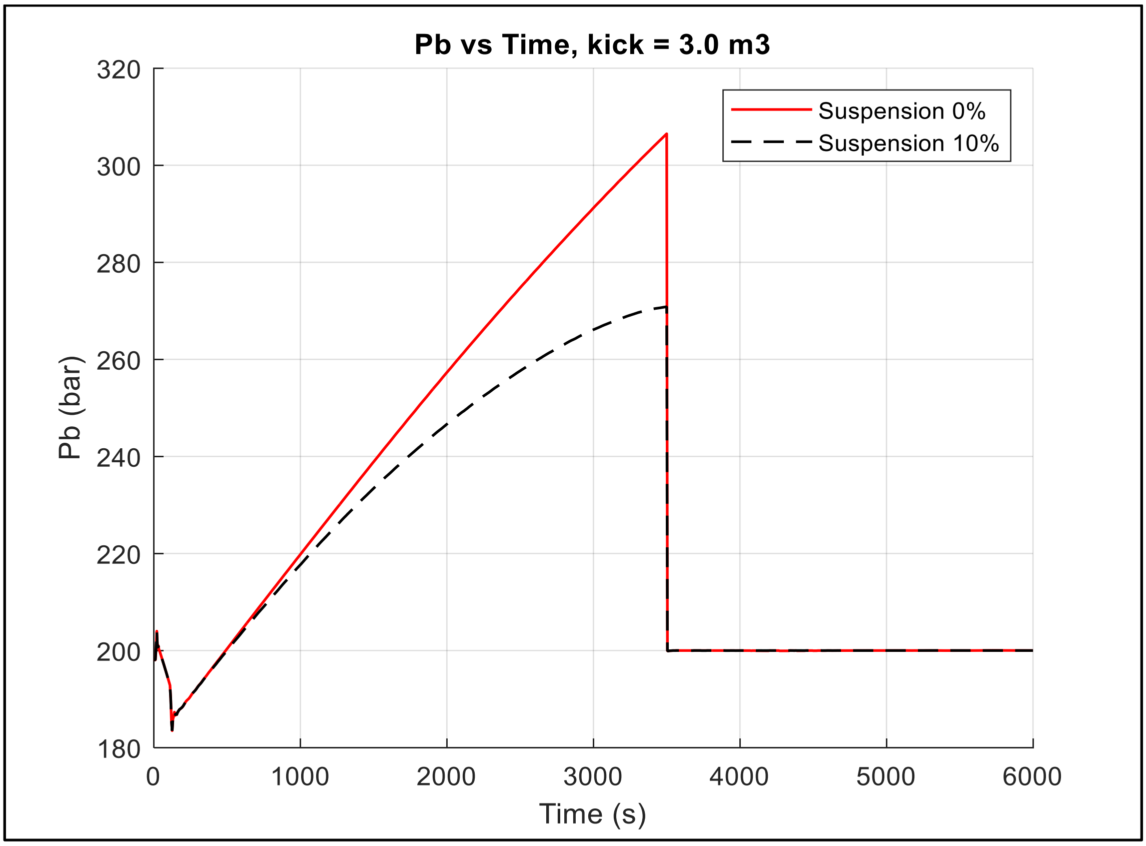


Figure 5-30: Bottomhole Pressure vs Time, Effect of Suspension

Figure 5-31 and Figure 5-32 show the total gas mass and the total gas volume plotted versus time. As can be seen, the 10% suspension case starts to leave the well much earlier than the 0% case with 3500s versus 4100s from Figure 5-31. This is because the 10% case has suspended gas at the bottom. As the 10% is displaced with a higher downward velocity, the bullheading operation is complete at a shorter time than for the 0% suspension case with 5500s versus 6000s.

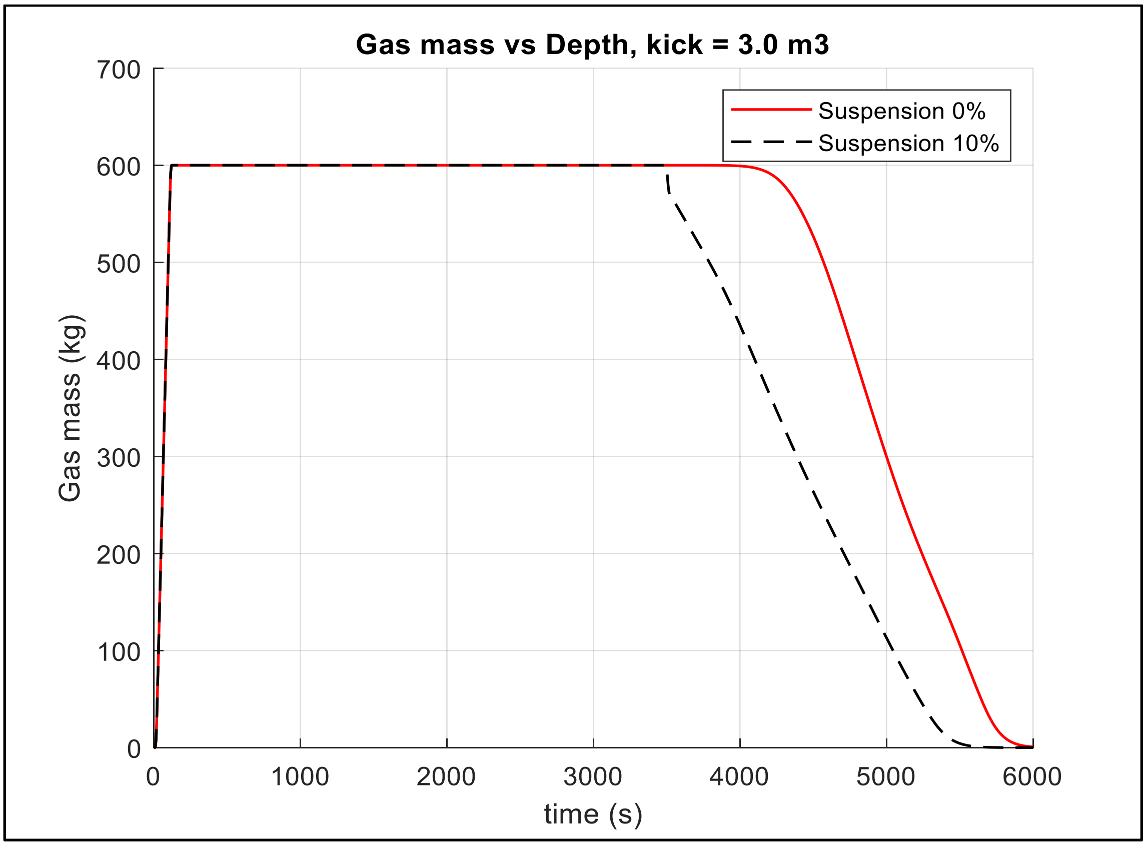


Figure 5-31: Gas Mass vs Time, Effect of Suspension

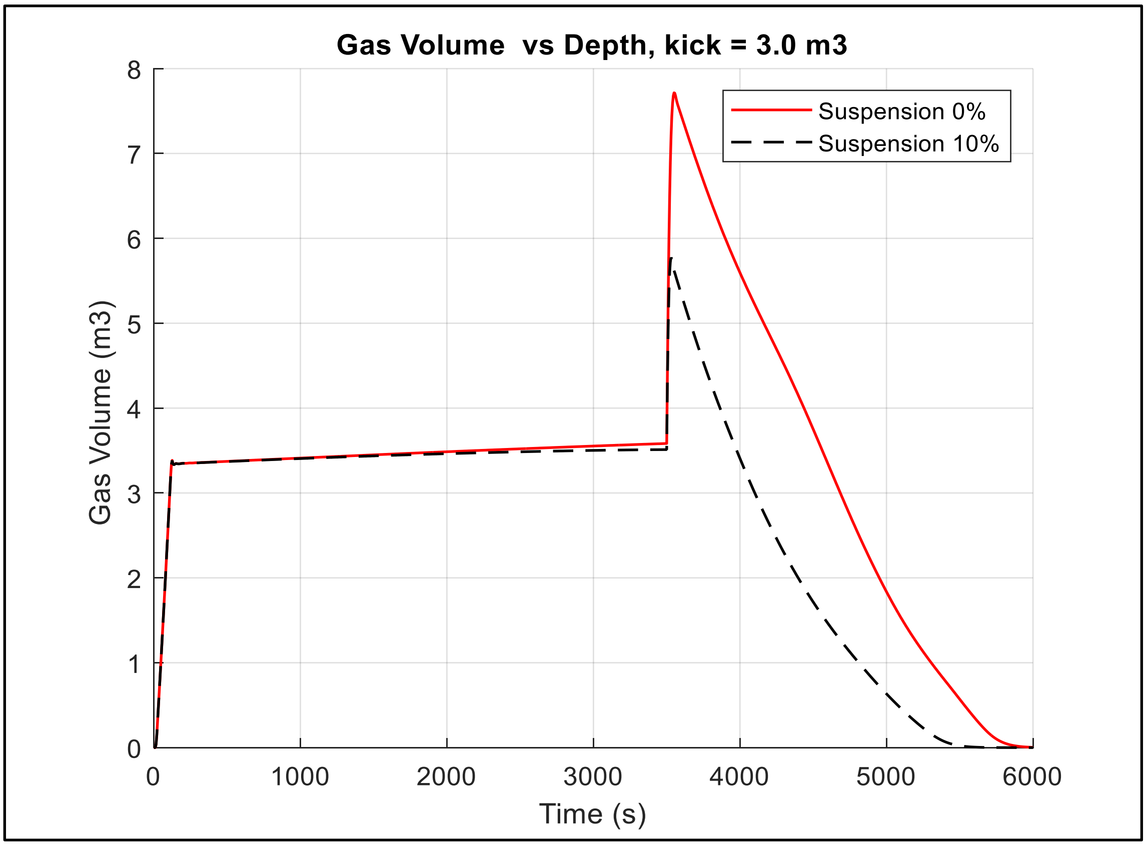


Figure 5-32: Gas Volume vs Time, Effect of Suspension

In conclusion, the suspended gas case is easier to bullhead with lower pressures to handle at the surface. The bullheading operation is complete in a shorter time than the non-suspension case. This will be true when the suspension effect results in a flow regime transition.

5.4 Discussion of the Results and Recommendations

In this chapter, a hypothetical well was simulated using the transient model described in chapter 3 using a simple and complete gas slip models separately. We varied the number of cells to test the effect of reducing the numerical diffusion on the solution. As the number of cells increased, the tail and front of the bubble became sharper, resulting in the kick being removed from the well in a shorter time. Therefore, to obtain a realistic estimate of the duration of the bullheading process and the required liquid volumes, it is crucial to reduce the numerical diffusion to an acceptable level. This will, however, require more simulation time. It was also observed that as the number of cells gets higher, the improvement in the results gets lower. In the simulated case, increasing the number of cells from 50 to 100 showed minor improvement in the results.

The critical bullheading rate formula derived in section 4.7.1 was tested on the case with 50 cells and proved to predict the critical bullheading rate accurately.

The simple slip relation model showed limitations in handling different kick sizes as it assumes one flow pattern throughout the simulation. Therefore, a complete slip model was implemented and used to study the effect of bullheading two kick sizes and the effect of suspension in non-Newtonian fluids. The simulation showed that it is easier to bullhead smaller kicks and the kicks with suspension because the small kick migrates with a bubble flow regime, and suspended gas kicks transition to bubble flow much faster.

6 Conclusions and Recommendations

This study simulated a bullheading process using a fully transient two-phase flow model based on the drift flux model.

A simple slip relation model replicated the simulation of a previous small-scale bullheading experiment. As with the original simulation, the difference between the simulation and the experimental results was significant. Attempts to improve the simulation by reducing the numerical diffusion and increasing the kick concentration did not improve the results. A simple analytical formula for the critical rate was derived. However, recalculating the critical bullheading rate from equation 56 gave a much lower value than predicted by the experiment. Furthermore, the calculated critical bullheading rate was tested with the simulation and showed a good agreement suggesting that the reported pumping rates in the experiments are probably higher than the actual. Therefore, we recommend to re-do the experiment and re-measure the liquid rates. We also recommend building the simulation side by side with the experiment setup. In this way, they check each other and minimize the error. It is difficult to conclude the main reason for the discrepancy at this stage.

This thesis showed the limitations of the simple slip model in handling two different kick sizes. The simulation results gave the same displacement velocity for the two kicks. This limitation in the simple slip model is because it uses one flow regime throughout the simulation. Therefore, one would expect to have one displacement velocity independent of the kick size. This was different when the complete gas slip relation was implemented. Then the small kick was displaced faster than the large kick because the first was in a bubble flow regime and the latter was in a slug flow regime.

We concluded that it is much easier to bullhead a small kick than a large one. Therefore, it is recommended to shut in the well as soon as the kick is detected to reduce the kick size in the well

We tested the suspension effect on bullheading using the model with the complete slip relation. One kick size was simulated with and without suspension. The simulation results showed that the gas bubbles tend to transition from slug to bubble flow as they migrate upward in the well, with the suspended gas case transitioning much faster. The suspended gas case experienced flow pattern transition to pure bubble flow at the end of the shut-in period. One would expect a similar effect even without suspension for deep wells. As a result, the suspended gas case was displaced with a higher downward velocity, and the bullheading operation was completed in a shorter time.

In this work, it was found effective to introduce the option of reloading saved workspaces to make it possible to continue a previously saved simulation without having to simulate it again from the beginning. This is very beneficial when one, e.g., shall try to investigate several scenarios with the same simulation pre-history

6.1 Future Work

In this study the simulation was performed by assuming a constant injection pressure, therefore, a future work could be to introduce an injectivity model to make the model more realistic. Also, the effect of suspension in non-Newtonian fluid was simulated by assuming a constant suspension ratio. Therefore, it can be an idea to implement an advanced non-Newtonian model that base the suspension amount on the fluid yield strength.

References

[1] A. Johnson, I. Rezmer-Cooper, T. Bailey, and D. McCann, "Gas Migration: Fast, Slow or Stopped," in *SPE/IADC Drilling Conference*, 1995, vol. All Days, SPE-29342-MS, doi: 10.2118/29342-ms. [Online]. Available: https://doi.org/10.2118/29342-MS

[2] M. Alfonso and O. Patino, "Experimental Investigation and Transient Flow Modelling of the Bullheading Process, Master's Thesis," Master of Secience in Petroleum Engineering Master's thesis in Petroleum engineering, Petroleum Engineering, University of Stavanger, 2020. [Online]. Available: https://hdl.handle.net/11250/2683701

[3] H. Crumpton, "Chapter Seven - Well Kill, Kick Detection, and Well Shut-In," in *Well Control for Completions and Interventions*, H. Crumpton Ed. Boston: Gulf Professional Publishing, 2018, pp. 235-360.

[4] T. Adam, L. William, and B. Hank, "An Experimental Study of Bullheading Operations for Control of Underground Blowouts," *Craft and Hawkins. Department of Petroleum Engineering,* 2001.

[5] W. L. Koederitz and A. T. Bourgoyne, "Gas Kick Behavior During Bullheading Operations."

[6] G. Raabe and S. Jortner, "Chapter Two - Routine well control methods," in *Universal Well Control*, G. Raabe and S. Jortner Eds.: Gulf Professional Publishing, 2022, pp. 79-165.

[7] A. A. Ghauri, K. K. Fjelde, and J. Frøyen, "A Transient Model for Hydraulic Simulation of Bullheading and Pressurized Mud Cap Drilling," in *ASME 2016 35th International Conference on Ocean, Offshore and Arctic Engineering*, 2016, vol. Volume 8: Polar and Arctic Sciences and Technology; Petroleum Technology, V008T11A029, doi: 10.1115/omae2016-54293. [Online]. Available: https://doi.org/10.1115/OMAE2016-54293

[8] P. Oudeman, "Kill Procedures to Avoid Formation Damage in the High Rate Gas Wells of an Underground Storage Project," in *SPE European Formation Damage Conference*, 1999, vol. All Days, SPE-54761-MS, doi: 10.2118/54761-ms. [Online]. Available: https://doi.org/10.2118/54761-MS

[9] P. Oudeman, D. ter Avest, E. O. Grodal, H. A. Asheim, and R. J. H. Meissner, "Bull Heading To Kill Live Gas Wells," in *European Petroleum Conference*, 1994, vol. All Days, SPE-28896-MS, doi: 10.2118/28896-ms. [Online]. Available: https://doi.org/10.2118/28896-MS

[10] S. Evje and K. K. Fjelde, "On a rough AUSM scheme for a one-dimensional two-phase model," *Computers & Fluids,* vol. 32, no. 10, pp. 1497-1530, 2003/12/01/ 2003, doi: https://doi.org/10.1016/S0045-7930(02)00113-5.

[11] M. Ishii, *Thermo-fluid dynamic theory of two-phase flow*. France: Eyrolles, Paris, France (in English), 1975, pp. Medium: X; Size: Pages: 248 2008-02-05.

[12] J. E. Udegbunam, K. K. Fjelde, S. Evje, and G. Nygaard, "A Simple Transient Flow Model for MPD and UBD Applications," in *SPE/IADC Managed Pressure Drilling & Underbalanced Operations Conference & Exhibition*, 2014, vol. All Days, SPE-168960-MS, doi: 10.2118/168960-ms. [Online]. Available: https://doi.org/10.2118/168960-MS

[13] K. K. Fjelde, J. Frøyen, and A. A. Ghauri, "A Numerical Study of Gas Kick Migration Velocities and Uncertainty," in *SPE Bergen One Day Seminar*, 2016, vol. Day 1 Wed, April 20, 2016, D011S007R002, doi: 10.2118/180053-ms. [Online]. Available: https://doi.org/10.2118/180053-MS

[14] J. E. Udegbunam, K. K. Fjelde, S. Evje, and G. Nygaard, "On the Advection-Upstream-Splitting-Method Hybrid Scheme: A Simple Transient-Flow Model for Managed-Pressure-Drilling and Underbalanced-Drilling Applications," *SPE Drilling & Completion,* vol. 30, no. 02, pp. 98-109, 2015, doi: 10.2118/168960-pa.

[15] N. Zuber and J. A. Findlay, "Average Volumetric Concentration in Two-Phase Flow Systems," *Journal of Heat Transfer,* vol. 87, no. 4, pp. 453-468, 1965, doi: 10.1115/1.3689137.

[16] F. Hovland and R. Rommetveit, "Analysis of Gas-Rise Velocities From Full-Scale Kick Experiments," in *SPE Annual Technical Conference and Exhibition*, 1992, vol. All Days, SPE-24580-MS, doi: 10.2118/24580-ms. [Online]. Available: https://doi.org/10.2118/24580-MS

[17] A. B. Johnson and D. B. White, "Gas-Rise Velocities During Kicks," *SPE Drilling Engineering,* vol. 6, no. 04, pp. 257-263, 1991, doi: 10.2118/20431-pa.

[18] A. R. Hasan, C. S. Kabir, and M. Sayarpour, "A Basic Approach to Wellbore Two-Phase Flow Modeling," in *SPE Annual Technical Conference and Exhibition*, 2007, vol. All Days, SPE-109868-MS, doi: 10.2118/109868-ms. [Online]. Available: https://doi.org/10.2118/109868-MS

[19] T. Z. Harmathy, "Velocity of large drops and bubbles in media of infinite or restricted extent," *AIChE Journal,* vol. 6, no. 2, pp. 281-288, 1960, doi: https://doi.org/10.1002/aic.690060222.

[20] H. V. Nickens, "A Dynamic Computer Model of a Kicking Well," *Spe Drilling Engineering,* vol. 2, pp. 159-173, 1987.

[21] V. Kelessidis and A. Dukler, "Modeling flow pattern transitions for upward gas-liquid flow in vertical concentric and eccentric annuli," *International Journal of Multiphase Flow,* vol. 15, no. 2, pp. 173-191, 1989.

[22] E. Caetano Filho, "Upward Vertical Two-Phase Flow through an Annulus," 01/01 1985.

[23] R. M. Davies and G. Taylor, "The Mechanics of Large Bubbles Rising through Extended Liquids and through Liquids in Tubes," *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences,* vol. 200, no. 1062, pp. 375-390, 1950. [Online]. Available: http://www.jstor.org/stable/98449.

[24] D. W. Rader, A. T. Bourgoyne, Jr., and R. H. Ward, "Factors Affecting Bubble-Rise Velocity Of Gas Kicks," *Journal of Petroleum Technology,* vol. 27, no. 05, pp. 571-584, 1975, doi: 10.2118/4647-pa.

[25] T. H. Ngoc Tat, D. Gomes, and K. K. Fjelde, "Use of a Transient Model for Studying Kick Migration Velocities and Build-Up Pressures in a Closed Well," in *ASME 2021 40th International Conference on Ocean, Offshore and Arctic Engineering*, 2021, vol. Volume 10: Petroleum Technology, V010T11A061, doi: 10.1115/omae2021-61143. [Online]. Available: https://doi.org/10.1115/OMAE2021-61143

[26] S. Evje and K. K. Fjelde, "Hybrid Flux-Splitting Schemes for a Two-Phase Flow Model," *Journal of Computational Physics,* vol. 175, no. 2, pp. 674-701, 2002/01/20/ 2002, doi: https://doi.org/10.1006/jcph.2001.6962.

[27] S. Benzoni-Gavage, "Analyse numérique des modèles hydrodynamiques d'écoulements diphasiques instationnaires dans les réseaux de production pétrolière," Lyon 1, 1991.

[28] R. J. Lorentzen and K. K. Fjelde, "Use of slopelimiter techniques in traditional numerical methods for multi-phase flow in pipelines and wells," *International Journal for Numerical Methods in Fluids,* vol. 48, pp. 723-745, 2005.

Appendices

The code for this study was provided by the school supervisor. The changes made are marked in red.

Appendix A

Main file code for replication of small-scale experiment

```
% Transient two-phase code based on AUSMV scheme: Gas and Water
% The code assumes uniform geometry

% time - Seconds

% p - pressure at new time level (Pa)
% dl - density of liquid at new time level (kg/m3)
% dg - density of gas at new time level (kg/m3)
% eg - phase volume fraction of gas at new time level (0-1)
% ev - phase volume fraction of liquid at new time level (0-1)
% vg - phase velocity of gas at new time level (m/s)
% vl - phase velocity of liquid at new time level (m/s)
% qv - conservative variables at new time level (3 in each cell)
% temp - temperature in well (K)

% po - pressure at old time level (Pa)
% dlo - density of liquid at old time level (kg/m3)
% dgo - density of gas at new old level (kg/m3)
% ego - phase volume fraction of gas at old time level (0-1)
% evo - phase volume fraction of liquid at old time level (0-1)
% vgo - phase velocity of gas at old time level (m/s)
% vlo - phase velocity of liquid at old time level (m/s)
% qvo - conservative variables at old time level (3 in each cell)
% temp - temperature in well (K)

clear;
t = cputime
tic,

% Geometry data/ Must be specified
welldepth = 3.685;
nobox = 25; %Number of boxes in the well

% Note that one can use more refined grid, 50, 100 boxes.
% When doing this, remember to reduce time step to keep the CFL number
% fixed below 0.25.. dt < cfl x dx/ speed of sound in water. If boxes are
% doubled, then half the time step.
nofluxes = nobox+1;  % Number of cell boundaries
dx = welldepth/nobox; % Boxlength
%dt = 0.005;

% Welldepth. Cell 1 start at bottom
x(1)= -1.0*welldepth+0.5*dx;
for i=1:nobox-1
    x(i+1)=x(i)+ dx;
end
```

```
% VERY IMPORTANT: BELOW THE TIMESTEP IS SET. MAKE SURE THAT THE
% CFL CONDIDTION IS FULFILLED. IF NUMBER OF BOXES IS CHANGED. DX WILL
% CHANGE AND DT HAS TO BE ADJUSTED TO KEEP THE CFL NUMBER FIXED.
dt = 0.00001; % Timestep (seconds)
dtdx = dt/dx;
time = 0; % initial time.
endtime = 10 % Time for ending simulation (seconds)
nosteps = endtime/dt; %Number of total timesteps. Used in for loop.
timebetweensavingtimedata = 0.001; % How often in s we save data vs time for
plotting.
nostepsbeforesavingtimedata = timebetweensavingtimedata/dt;

% Slip parameters used in the gas slip relation. vg =Kvmix+S
k = 1.2;
s = 0.2169;% calculated from Taylor bubble law

% Boundary condition at outlet
pbondout=101325; % Pascal (1 bar)

% Initial temperature distribution. (Kelvin)
% Note that this is only used if we use density models that depend on
% temperature
tempbot = 20+273;
temptop = 20+273;
tempgrad= (tempbot-temptop)/welldepth;
tempo(1)=tempbot-dx/2*tempgrad;
for i = 1:nobox-2
    tempo(i+1)=tempo(i)-dx*tempgrad;
end
tempo(nobox)=tempo(nobox-1)-dx*tempgrad;

temp = tempo;

% Different fluid density parameters
% Note how we switch between different models later.
% These parameters are used when finding the
% primitive variables pressure, densities in an analytical manner.
% Changing parameters here, you must also change parameters inside the
% density routines roliq and rogas.

% Simple Water density model & Ideal Gas. See worknote Extension of AUSMV
% scheme.
rho0=1000; % Water density at STC (Standard Condition) kg/m3
Bheta=2.2*10^9; % Parameter that depend on the compressibility of water
Alpha=0.000207; % Parameter related to thermal expansion/compression
R = 286.9; % Ideal gas parameter
P0=100000; % Pressure at STC (Pa)
T0=20+273.15; % Temperature at STC (K)

% Very simple models (PET510 compendium)
al = 1500; % Speed of sound in water.
rt= 100000; % Ideal gas parameter in model rhog = p/rt (rt = ag^2)
rho0=1000; % Water density at STC (Standard Condition) kg/m3
P0=100000; % Pressere at STC (Pa)
T0=20+273.15; % Temperature at STC (K)

% Viscosities (Pa*s)/Used in the frictional pressure loss model (dpfric).
viscl = 0.001; % Liquid phase
```

```
viscg = 0.0000182; % Gas phase

% Gravity constant
g = 9.81; % Gravitational constant m/s2

% Well opening. opening = 1, fully open well, opening = 0 (<0.01), the well
% is fully closed. This variable will control what boundary conditions that
% will apply at the outlet (both physical and numerical): We must change
% this further below in the code if we want to change status on this.

wellopening = 1.0;  % This variable determines if
%the well is closed or not, wellopening = 1.0 -> open. welllopening = 0
%-> Well is closed. This variable affects the boundary treatment.

bullheading = 0.0; % This variable can be set to 1.0 if we want to simulate
% a bullheading operation. But the normal is to set this to zero.

% Specify if the primitive variables shall be found either by
% a numerical or analytical approach. If analytical = 1, analytical
% solution is used. If analytical = 0. The numerical approach is used.
% using the itsolver subroutine where the bisection numerical method
% is used. We use analytical.
analytical = 1;

% Initialization of rest of geometry.
% Here we specify the outer and inner diameter and the flow area
% We assume 12.25 x 5 inch annulus. But this can be modified.
for i = 1:nobox
    do(i)=0.0392;
    di(i) = 0.0;
    area(i) = 3.14/4*(do(i)*do(i)- di(i)*di(i));
end

% Initialization of slope limiters. These are used for
% reducing numerical diffusion and will be calculated for each timestep.
% They make the numerical scheme second order.
for i = 1:nobox
    sl1(i)=0;
    sl2(i)=0;
    sl3(i)=0;
    sl4(i)=0;
    sl5(i)=0;
    sl6(i)=0;
end

% Now comes the intialization of the physical variables in the well.
% First primitive variables, then the conservative ones.

% Below we intialize pressure and fluid densities. We start from top of
% the well and calculate downwards. The calculation is done twice with
% updated values to get better approximation. Only hydrostatic
% considerations since we start with a static well.
for i = 1:nobox
    eg(i)=0.0;  % Gas volume fraction
    ev(i)=1-eg(i); % Liquid volume fraction
end

p(nobox)= pbondout+0.5*9.81*dx*...
    (ev(nobox)*rholiq(P0,T0)+eg(nobox)*rogas(P0,T0));   % Pressure (Pa)
```

```
dl(nobox)=rholiq(p(nobox),tempo(nobox));  % Liquid density kg/m3
dg(nobox)=rogas(p(nobox),tempo(nobox));   % Gas density kg/m3

for i=nobox-1:-1:1
    p(i)=p(i+1)+dx*9.81*(ev(i+1)*dl(i+1)+eg(i+1)*dg(i+1));
    dl(i)=rholiq(p(i),tempo(i));
    dg(i)=rogas(p(i),tempo(i));
end

for i=nobox-1:-1:1
    rhoavg1= (ev(i+1)*dl(i+1)+eg(i+1)*dg(i+1));
    rhoavg2= (ev(i)*dl(i)+eg(i)*dg(i));
    p(i)=p(i+1)+dx*9.81*(rhoavg1+rhoavg2)*0.5;
    dl(i)=rholiq(p(i),tempo(i));
    dg(i)=rogas(p(i),tempo(i));
end

% Intitialize phase velocities, volume fractions, conservative variables
% and friction and hydrostatic gradients.
% The basic assumption is static fluid, one phase liquid.
for i = 1:nobox
    vl(i)=0; % Liquid velocity new time level.
    vg(i)=0; % Gas velocity at new time level
    eg(i)=0.0;  % Gas volume fraction
    ev(i)=1-eg(i); % Liquid volume fraction
    qv(i,1)=dl(i)*ev(i)*area(i);  % Conservative variable for liquid mass
(kg/m)
    qv(i,2)=dg(i)*eg(i)*area(i);  % Conservative variable for gas mass (kg/m)
    qv(i,3)=(dl(i)*ev(i)*vl(i)+dg(i)*eg(i)*vg(i))*area(i); % Conservative
variable for mixture moementum
    fricgrad(i)=0;   % Pa/m
    hydgrad(i)=g*(dl(i)*ev(i)+eg(i)*dg(i)); % Pa/m
end

% Section where we also initialize values at old time level
for i=1:nobox
    dlo(i)=dl(i);
    dgo(i)=dg(i);
    po(i)=p(i);
    ego(i)=eg(i);
    evo(i)=ev(i);
    vlo(i)=vl(i);
    vgo(i)=vg(i);
    qvo(i,1)=qv(i,1);
    qvo(i,2)=qv(i,2);
    qvo(i,3)=qv(i,3);
end

% Intialize fluxes between the cells/boxes
for i = 1:nofluxes
    for j =1:3
        flc(i,j)=0.0; % Flux of liquid over box boundary
        fgc(i,j)=0.0; % Flux of gas over box boundary
        fp(i,j)= 0.0; % Pressure flux over box boundary
    end
end

% Main program.
% Here we will progress in time. First some intializations
% and definitions to take out results. The for loop below runs until the
% simulation is finished.
countsteps = 0;
counter=0;
printcounter = 1;
pin(printcounter) = (p(1)+dx*0.5*hydgrad(1))/100000; % Pressure in bar at
bottom for time storage
pout(printcounter)= pbondout/100000; % Pressure at outlet of uppermost cell
pnobox(printcounter)= p(nobox)/100000; % Pressure in middle of uppermost cell
liquidmassrateout(printcounter) = 0; % liquid mass rate at outlet kg/s
gasmassrateout(printcounter)=0; % gass mass rate at outlet kg/s
timeplot(printcounter)=time; % Array for time and plotting of variables vs
time
pitvolume=0;
pitrate =0;
pitgain(printcounter)=0;

kickvolume=0;
bullvolume=0;

% The temperature is not updated but kept fixed according to the
% initialization.

% Now comes the for loop that runs forward in time. This is repeated for
% every timestep.
for i = 1:nosteps
    countsteps=countsteps+1;
    counter=counter+1;
    time = time+dt; % Step one timestep and update time.

    % Then a section where specify the boundary conditions.
    % Here we specify the inlet rates of the different phases at the
    % bottom of the pipe in kg/s. We interpolate to make things smooth.
    % It is also possible to change the outlet boundary status of the well
    % here. First we specify rates at the bottom and the pressure at the
outlet
    % in case we have an open well. This is a place where we can change the
    % code to control simulations. If the well shall be close, wellopening
must
    % be set to 0. It is also possible to reverse the flow (bullheading).
    % In the example below, we take a gas kick and then circulate this

    % out of the well without closing the well. (how you not should perform
    % well control)
    % Note there are two variables wellopening and bullheading that can be

    % changed in the control structure below to close the well or start
    % reversing the flow i.e. pumping downwards.
    % Note that if we will change to bullheading throughout the control
stucture,

    % the variable inletligmassrate
    % has to be defined as negative since pumping downwards at outlet will be
    % in negative direction (postive direction of flow has been chosen to be
    % upwards)
    % NB, NOTE THAT THIS IS ONE OF THE MAIN PLACES WHERE YOU HAVE TO ADJUST
THE

    % CODE TO CONTROL THE SIMULATION SCENARIO.
```

```
inletligmassrate=0.0;
inletgasmassrate=0.0;

%activate tests as required
%test 1
XX = 0.000295;
bullheadingrate  = 0.606;  % Note this is kg/s or l/s

% test 2
%     XX = 0.000296;
%     bullheadingrate  = 0.735;

%test 3
%     XX = 0.000305;
%     bullheadingrate  = 0.896;

%     %test 4
%     XX = 0.000276;
%     bullheadingrate  = 1.079;

%here we introduce xp to control the concentration of the kick and
%maintain same total amount of gas XX
xp = 1.0;%seconds
XX = XX/xp;% constant injection rate kg/s

if (time>=0)&&(time <=0.25*xp)
    inletligmassrate=0.0;
    inletgasmassrate=XX*time/(0.25*xp);
elseif (time>0.25*xp)&&(time<=xp)
    inletligmassrate=0.0;
    inletgasmassrate=XX;
elseif (time>xp)&&(time<=1.25*xp)
    inletligmassrate=0.0;
    inletgasmassrate=XX-XX*(time-xp)/(0.25*xp);
elseif (time>1.25*xp)&&(time<=(6.25))
    wellopening = 0.0;    % This is the closed in period
    inletligmassrate=0.0;
    inletgasmassrate=0.0;
elseif(time>6.25)&&(time<=6.50)
    wellopening = 1.0;  % Here we start ramping up the bullheading
    bullheading = 1.0;
    inletligmassrate=-1.0*bullheadingrate*(time-6.25)/0.25;
    inletgasmassrate=0.0;
    intparam = (time-6.25)/(6.50-6.25);
    k = 1.2*(1-intparam)+1.12*intparam; % Here we use timeinterpolation
    % to change the slip parameter from K = 1.2 to K = 1.12 before
    % bullheading. We need smooth transition of parameters.
elseif (time>6.50)
    k = 1.12;
    wellopening = 1.0;
    bullheading = 1.0;
    inletligmassrate=-1.0*bullheadingrate;
    inletgasmassrate=0.0;
end

% XX = 4; % Gasrate in kg/s
%
% YY= 40; % Liquidrate in kg/s
%
```

```
    % if (time < 10)
    %
    %   inletligmassrate=0.0;
    %   inletgasmassrate=0.0;
    %
    % elseif ((time>=10) & (time < 20))
    %   inletligmassrate = YY*(time-10)/10;  % Interpolate the rate from 0 to
value wanted.
    %   inletgasmassrate = XX*(time-10)/10;
    %
    % elseif ((time >=20) && (time<200))
    %   inletligmassrate = YY;
    %   inletgasmassrate = XX;
    % elseif ((time >=200) & (time<210))
    % %  inletligmassrate = YY-YY*(time-200)/10;
    %   inletligmassrate = YY-YY*(time-200)/10;
    %   inletgasmassrate = XX-XX*(time-200)/10;
    % elseif (time > 210)
    %   inletligmassrate=0;
    %   inletgasmassrate=0;
    %   wellopening = 0;
    % end

    % The commented code below are from some previous runs. It shows. e.g.
how
    % we can close the well.
    %elseif((time>=500)&(time<510))
    %   inletligmassrate = YY-YY*(time-500)/10;
    %   inletgasmassrate = XX-XX*(time-500)/10;
    % elseif(time>=510)
    %   inletligmassrate=0;
    %   inletgasmassrate=0;
    %   wellopening=0.0;
    % end

    %XX = 4;
    % XX (kg/s) is a variable for introducing a kick in the well.
    %YY = 15; % Liquid flowrate (kg/s) (1 kg/s = 1 l/s approx)
    % if (time < 10)
    %
    %   inletligmassrate=0.0;
    %   inletgasmassrate=0.0;
    %
    % elseif ((time>=10) & (time < 20))
    %   inletligmassrate = 0*(time-10)/10;
    %   inletgasmassrate = XX*(time-10)/10;
    %
    % elseif ((time >=20) & (time<110))
    %   inletligmassrate = 0;
    %   inletgasmassrate = XX;
    %
    % elseif ((time>=110)& (time<120))
    %   inletligmassrate = 0;
    %   inletgasmassrate = XX-XX*(time-110)/10;
    % elseif ((time>=120&time<130))
    %   inletligmassrate =0;
    %   inletgasmassrate =0;
    % elseif ((time>=130)&(time<300))
    %   inletligmassrate =0;
    %   inletgasmassrate =0;
    % elseif ((time>=300)&(time<310))
    %   inletligmassrate= YY*(time-300)/10;
    %   inletgasmassrate =0;
    % elseif((time>=310))
    %   inletligmassrate= YY;
    %   inletgasmassrate =0;
    % end

    kickvolume = kickvolume+inletgasmassrate/dgo(1)*dt;  % Here we find the
kickvolume
    % initially induced in the well.

    % Here we specify the physical outlet pressure. Here we have given the
pressure as
    % constant. It would be possible to adjust it during openwell conditions
    % either by giving the wanted pressure directly (in the command lines
    % above) or by finding it indirectly through a chokemodel where the
chokeopening
    % would have had to be  an input parameter. The chokeopening variable
would equally had
    % to be adjusted inside the controle structure given above.
    pressureoutlet = pbondout;

    % Based on these given physical boundary values combined with use
    % of extrapolations techniques
    % for the remaining unknowns at the boundaries, we will define the mass
and
    % momentum fluxes at the boundaries (inlet and outlet of pipe).

    % inlet/bottom fluxes first.
    if (bullheading<=0)
        % here we pump from bottom
        % liquid
        flc(1,1)= inletligmassrate/area(1);
        flc(1,2)= 0.0;
        flc(1,3)= flc(1,1)*vlo(1);
        % gas
        fgc(1,1)= 0.0;
        fgc(1,2)= inletgasmassrate/area(1);
        fgc(1,3)= fgc(1,2)*vgo(1);
        % pressure
        fp(1,1)= 0.0;
        fp(1,2)= 0.0;
        % Old way of treating the boundary
        %     fp(1,3)= po(1)+0.5*(po(1)-po(2)); %Interpolation used to find
the
        % pressure at the inlet/bottom of the well.
        % New way of treating the boundary
        fp(1,3)= po(1)...
            +0.5*dx*(dlo(1)*evo(1)+dgo(1)*ego(1))*g...
            +0.5*dx*fricgrad(1);
    else
        % Here we pump from the top. All masses are assumed to flow out of
the
        % well into the formation. We use first order extrapolation.
        %liquid
        flc(1,1)=dlo(1)*evo(1)*vlo(1);
        flc(1,2)=0.0;
        flc(1,3)=flc(1,1)*vlo(1);
        %gas
        fgc(1,1)=0.0;
        fgc(1,2)=dgo(1)*ego(1)*vgo(1);
        fgc(1,3)=fgc(1,2)*vgo(1);
        %pressure
        fp(1,1)=0.0;
        fp(1,2)=0.0;
        fp(1,3)=135000;   % This was a fixed pressure set at bottom when
bullheading (Pascal)
    end

    % Outlet fluxes (open & closed conditions)
    if (wellopening>0.01)
        % Here open end condtions are given. We distinguish between
bullheading
        % & normal circulation.
        if (bullheading<=0)  % Here we don't bullhead, i.e we circulate from
bottom
            % Here the condition is normal ciruclation and open well)
            %Liquid flux
            flc(nofluxes,1)= dlo(nobox)*evo(nobox)*vlo(nobox);
            flc(nofluxes,2)= 0.0;
            flc(nofluxes,3)= flc(nofluxes,1)*vlo(nobox);
            % gas flux
            fgc(nofluxes,1)= 0.0;
            fgc(nofluxes,2)= dgo(nobox)*ego(nobox)*vgo(nobox);
            %         fgc(nofluxes,2)=0; Activate if gas is sucked in!?
            fgc(nofluxes,3)= fgc(nofluxes,2)*vgo(nobox);
            %pressure flux
            fp(nofluxes,1)= 0.0;
            fp(nofluxes,2)= 0.0;
            fp(nofluxes,3)= pressureoutlet;
        else
            % Here we are bullheading.
            %liquid flux
            flc(nofluxes,1)= inletligmassrate/area(nobox);
            flc(nofluxes,2)= 0.0;
            flc(nofluxes,3)= flc(nofluxes,1)*vlo(nobox);
            %gas flux
            fgc(nofluxes,1)=0.0;
            fgc(nofluxes,2)=0.0;
            fgc(nofluxes,3)=0.0;
            %pressure flux
            fp(nofluxes,1)=0.0;
            fp(nofluxes,2)=0.0;
            fp(nofluxes,3)= po(nobox)...
                -0.5*dx*(dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g...
                +0.5*dx*fricgrad(nobox); %check sign here on friction
            % Physcially, the friction should be added when going from
            % mid point in upper cell to outlet. But if fricgrad(nobox) is
            % negative there should be a minus in front of the term to have
            % + in the end.
        end
    else
        % Here closed end conditions are given
        %liquid flux
        flc(nofluxes,1)= 0.0;
        flc(nofluxes,2)= 0.0;
        flc(nofluxes,3)= 0.0;
        %gas flux
        fgc(nofluxes,1)= 0.0;
        fgc(nofluxes,2)= 0.0;
        fgc(nofluxes,3)= 0.0;
        %pressure flux
        fp(nofluxes,1)=0.0;
        fp(nofluxes,2)=0.0;
        %    Old way of treating the boundary
        %     fp(nofluxes,3)= po(nobox)-0.5*(po(nobox-1)-po(nobox));
        %    New way of treating the boundary
        fp(nofluxes,3)= po(nobox)...
            -0.5*dx*(dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g;
        %     -0.5*dx*fricgrad(nobox); % Neglect friction since well is
closed.
    end

    % Implementation of slopelimiters. They are applied on the physical
    % variables like phase densities, phase velocities and pressure.
    % It was found that if the slopelimiters were set to zero in
    % the boundary cells, the pressure in these became wrong. E.g. the upper
    % cell get an interior pressure that is higher than it should be e.g.
when
    % being static (hydrostatic pressure was too high). The problem was
reduced
    % by copying the slopelimiters from the interior cells. However, both
    % approaches seems to give the same BHP pressure vs time but the latter
    % approach give a more correct pressure vs depth profile. It is also
better
    % to use when simulating pressure build up where the upper cell pressure
    % must be monitored. It should be checked more in detail before
concluding.
    % BUT; there has been mass conservation problems with the scheme for the
    % case where the slopelimiters were copied (see master thesis of Keino)
    % A possible fix has been included below where the slopelimiter related
to
    % the gas volume fraction is set to zero in the first cell.
    for i=2:nobox-1
        sl1(i)=minmod(dlo(i-1),dlo(i),dlo(i+1),dx);
        sl2(i)=minmod(po(i-1),po(i),po(i+1),dx);
        sl3(i)=minmod(vlo(i-1),vlo(i),vlo(i+1),dx);
        sl4(i)=minmod(vgo(i-1),vgo(i),vgo(i+1),dx);
        sl5(i)=minmod(ego(i-1),ego(i),ego(i+1),dx);
        sl6(i)=minmod(dgo(i-1),dgo(i),dgo(i+1),dx);
    end
    % Slopelimiters in outlet boundary cell are set to zero!
    %      sl1(nobox)=0;
    %      sl2(nobox)=0;
    %      sl3(nobox)=0;
    %      sl4(nobox)=0;
    %      sl5(nobox)=0;
    %      sl6(nobox)=0;
    % Slopelimiters in outlet boundary cell are copied from neighbour cell!
    sl1(nobox)=sl1(nobox-1);
    sl2(nobox)=sl2(nobox-1);
    sl3(nobox)=sl3(nobox-1);
    sl4(nobox)=sl4(nobox-1);
    sl5(nobox)=sl5(nobox-1);
    sl6(nobox)=sl6(nobox-1);
    % Slopelimiters in inlet boundary cell are set to zero!
    %      sl1(1)=0;
    %      sl2(1)=0;
    %      sl3(1)=0;
    %      sl4(1)=0;
    %      sl5(1)=0;
    %      sl6(1)=0;
    % Slopelimiters in inlet boundary cell are copied from neighbour cell!
    sl1(1)=sl1(2);
    sl2(1)=sl2(2);
    sl3(1)=sl3(2);
    sl4(1)=sl4(2);
    sl5(1)=sl5(2);
    sl6(1)=sl6(2);
    % FIX FOR OMITTING THE GAS MASS CONSERVATION PROBLEM
    sl5(1)=0;
    % Now we will find the fluxes between the different cells.
    % NB - IMPORTANE -  Note that if we change the compressibilities/sound
velocities of
    % the fluids involved, we may need to do changes inside the csound
function.
    % But the effect of this is unclear.
    for j = 2:nofluxes-1
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % First order method is from here: If you want to test this, activate
this
        % and comment the second order code below.
        %        cl = csound(ego(j-1),po(j-1),dlo(j-1),k);
        %        cr = csound(ego(j),po(j),dlo(j),k);
        %        c = max(cl,cr);
        %        pll = psip(vlo(j-1),c,evo(j));
        %        plr = psim(vlo(j),c,evo(j-1));
        %        pgl = psip(vgo(j-1),c,ego(j));
        %        pgr = psim(vgo(j),c,ego(j-1));
        %        vmixr = vlo(j)*evo(j)+vgo(j)*ego(j);
        %        vmixl = vlo(j-1)*evo(j-1)+vgo(j-1)*ego(j-1);
        %
        %        pl = pp(vmixl,c);
        %        pr = pm(vmixr,c);
        %        mll= evo(j-1)*dlo(j-1);
        %        mlr= evo(j)*dlo(j);
        %        mgl= ego(j-1)*dgo(j-1);
        %        mgr= ego(j)*dgo(j);
        %
        %        flc(j,1)= mll*pll+mlr*plr;
        %        flc(j,2)= 0.0;
        %        flc(j,3)= mll*pll*vlo(j-1)+mlr*plr*vlo(j);
        %
        %        fgc(j,1)=0.0;
        %        fgc(j,2)= mgl*pgl+mgr*pgr;
        %        fgc(j,3)= mgl*pgl*vgo(j-1)+mgr*pgr*vgo(j);
        %
        %        fp(j,1)= 0.0;
        %        fp(j,2)= 0.0;
        %        fp(j,3)= pl*po(j-1)+pr*po(j);
        %  First order methods ends here
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Second order method starts here:
% Here slopelimiter is used on all variables except phase velocoties
psll = po(j-1)+dx/2*sl2(j-1);
pslr = po(j)-dx/2*sl2(j);
dsll = dlo(j-1)+dx/2*sl1(j-1);
dslr = dlo(j)-dx/2*sl1(j);
dgll = dgo(j-1)+dx/2*sl6(j-1);
dglr = dgo(j)-dx/2*sl6(j);

vlv = vlo(j-1)+dx/2*sl3(j-1);
vlh = vlo(j)-dx/2*sl3(j);
vgv = vgo(j-1)+dx/2*sl4(j-1);
vgh = vgo(j)-dx/2*sl4(j);

gvv = ego(j-1)+dx/2*sl5(j-1);
gvh = ego(j)-dx/2*sl5(j);
lvv = 1-gvv;
lvh = 1-gvh;

cl = csound(gvv,psll,dsll,k);
cr = csound(gvh,pslr,dslr,k);
c = max(cl,cr);

pll = psip(vlo(j-1),c,lvh);
plr = psim(vlo(j),c,lvv);
pgl = psip(vgo(j-1),c,gvh);
pgr = psim(vgo(j),c,gvv);
vmixr = vlo(j)*lvh+vgo(j)*gvh;
vmixl = vlo(j-1)*lvv+vgo(j-1)*gvv;

pl = pp(vmixl,c);
pr = pm(vmixr,c);

mll= lvv*dsll;
mlr= lvh*dslr;
mgl= gvv*dgll;
mgr= gvh*dglr;

flc(j,1)= mll*pll+mlr*plr;
flc(j,2)= 0.0;
flc(j,3)= mll*pll*vlo(j-1)+mlr*plr*vlo(j);

fgc(j,1)=0.0;
fgc(j,2)= mgl*pgl+mgr*pgr;
fgc(j,3)= mgl*pgl*vgo(j-1)+mgr*pgr*vgo(j);

fp(j,1)= 0.0;
fp(j,2)= 0.0;
fp(j,3)= pl*psll+pr*pslr;
%%% Second order method ends here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Here sloplimiters is used on all variables. This
% has not worked so well yet. Therefore it is commented away.
%       psll = po(j-1)+dx/2*sl2(j-1);
%       pslr = po(j)-dx/2*sl2(j);
%       dsll = dlo(j-1)+dx/2*sl1(j-1);
```

```
    %       dslr = dlo(j)-dx/2*sl1(j);
    %       dgll = dgo(j-1)+dx/2*sl6(j-1);
    %       dglr = dgo(j)-dx/2*sl6(j);
    %
    %       vlv = vlo(j-1)+dx/2*sl3(j-1);
    %       vlh = vlo(j)-dx/2*sl3(j);
    %       vgv = vgo(j-1)+dx/2*sl4(j-1);
    %       vgh = vgo(j)-dx/2*sl4(j);
    %
    %       gvv = ego(j-1)+dx/2*sl5(j-1);
    %       gvh = ego(j)-dx/2*sl5(j);
    %       lvv = 1-gvv;
    %       lvh = 1-gvh;
    %
    %       cl = csound(gvv,psll,dsll,k);
    %       cr = csound(gvh,pslr,dslr,k);
    %       c = max(cl,cr);
    %
    %       pll = psip(vlv,c,lvh);
    %       plr = psim(vlh,c,lvv);
    %       pgl = psip(vgv,c,gvh);
    %       pgr = psim(vgh,c,gvv);
    %       vmixr = vlh*lvh+vgh*gvh;
    %       vmixl = vlv*lvv+vgv*gvv;
    %
    %       pl = pp(vmixl,c);
    %       pr = pm(vmixr,c);
    %       mll= lvv*dsll;
    %       mlr= lvh*dslr;
    %       mgl= gvv*dgll;
    %       mgr= gvh*dglr;
    %
    %       flc(j,1)= mll*pll+mlr*plr;
    %       flc(j,2)= 0.0;
    %       flc(j,3)= mll*pll*vlv+mlr*plr*vlh;
    %
    %       fgc(j,1)=0.0;
    %       fgc(j,2)= mgl*pgl+mgr*pgr;
    %       fgc(j,3)= mgl*pgl*vgv+mgr*pgr*vgh;
    %
    %       fp(j,1)= 0.0;
    %       fp(j,2)= 0.0;
    %       fp(j,3)= pl*psll+pr*pslr;
end

% Fluxes have now been calculated. We will now update the conservative
% variables in each of the numerical cells.
% The source terms can be calculated by using a for loop.
% Note that the model is sensitive to how we treat the model
% for low Reynolds numbers (possible discontinuity in the model)
for j=1:nobox
    fricgrad(j)=dpfric(vlo(j),vgo(j),evo(j),ego(j),dlo(j),dgo(j), ...
        po(j),do(j),di(j),viscl,viscg); % Pa/m
    hydgrad(j)=g*(dlo(j)*evo(j)+dgo(j)*ego(j)); % Pa/m
end
sumfric = 0;
sumhyd= 0;
for j=1:nobox
    % Here we solve the three conservation laws for each cell and update
```

```
        % the conservative variables qv
        ar = area(j);
        % Liquid mass conservation
        qv(j,1)=qvo(j,1)-dtdx*((ar*flc(j+1,1)-ar*flc(j,1))...
            +(ar*fgc(j+1,1)-ar*fgc(j,1))...
            +(ar*fp(j+1,1)-ar*fp(j,1)));
        % Gas mass conservation:
        qv(j,2)=qvo(j,2)-dtdx*((ar*flc(j+1,2)-ar*flc(j,2))...
            +(ar*fgc(j+1,2)-ar*fgc(j,2))...
            +(ar*fp(j+1,2)-ar*fp(j,2)));
        % Mixture momentum conservation:
        qv(j,3)=qvo(j,3)-dtdx*((ar*flc(j+1,3)-ar*flc(j,3))...
            +(ar*fgc(j+1,3)-ar*fgc(j,3))...
            +(ar*fp(j+1,3)-ar*fp(j,3)))...
            -dt*ar*(fricgrad(j)+hydgrad(j));
        %  Add up the hydrostatic pressure  and friction  in the whole well.
        sumfric=sumfric+fricgrad(j)*dx;
        sumhyd=sumhyd+hydgrad(j)*dx;
    end
    % Section where we find the physical variables (pressures, densities etc)
    % from the conservative variables. Some tricks to ensure stability. These
    % are induced to avoid negative masses.
    gasmass=0;
    liqmass=0;
    for j=1:nobox
        % Remove the area from the conservative variables to find the
        % the primitive variables from the conservative ones.
        qv(j,1)= qv(j,1)/area(j);
        qv(j,2)= qv(j,2)/area(j);
        if (qv(j,1)<0.00000001)  % Trick to avoid negative masses.
            qv(j,1)=0.00000001;
        end
        if (qv(j,2)< 0.00000001)  % Trick to avoid negative masses.
            qv(j,2)=0.00000001;
        end

        % Here we summarize the mass of gas and liquid in the well
respectively.
        % These variables are important to show that the scheme is conserving
        % mass. (if e.g. gas leaks in our out of the well unintentionally in
the simulation
        % without being specified in the code,something fundamental is wrong.
        gasmass = gasmass+qv(j,2)*area(j)*dx;
        liqmass = liqmass+qv(j,1)*area(j)*dx;
        % Below, we find the primitive variables pressure and densities based
on
        % the conservative variables q1,q2. One can choose between getting
them by
        % analytical or numerical solution approach specified in the
beginning of
        % the program. Ps. For more advanced density models, this must be
changed.
        if (analytical == 1)
            %       % Analytical solution:
            % here the simple density models used in PET 510 Wellflow
modelling
            % compendium is used.
            %        t1=rho0-P0/al^2;
            %
            % %  Coefficients:
            %        a = 1/(al*al);
            %        b = t1-qv(j,1)-rt*qv(j,2)/(al*al);
            %        c = -1.0*t1*rt*qv(j,2);
            %
            % %       Note here we use the very simple models from the PET510
course
            %         p(j)=(-b+sqrt(b*b-4*a*c))/(2*a);  % Pressure
            %         dl(j)=rholiq(p(j),temp(j)); % Density of liquid
            %         dg(j)=rogas(p(j),temp(j)); % Density of gas

            %     The code below can be activated if we want to switch to the
other set
            %     of density models. Also then remember to do the changes
inside
            %     functions rogas og rholiq if we change density models.
            x1=rho0-P0*rho0/Bheta-rho0*Alpha*(temp(j)-T0);
            x2=rho0/Bheta;
            x3=-qv(j,2)*R*temp(j);
            a = x2;
            b = x1+x2*x3-qv(j,1);
            c = x1*x3;
            p(j)=(-b+sqrt(b*b-4*a*c))/(2*a);  % Pressure
            dl(j)=rholiq(p(j),temp(j));
            dg(j)=rogas(p(j),temp(j));
        else
            %Numerical Solution: This might be used if we use more complex
            %density models. Has not been used for years.
            [p(j),error]=itsolver(po(j),qv(j,1),qv(j,2)); % Pressure
            dl(j)=rholiq(p(j),temp(j)); % Density of liquid
            dg(j)=rogas(p(j)); % Density of gas
            % Incase a numerical solution is not found, the program will
write out "error":
            if error > 0
                error
            end
        end
        %   Find phase volume fractions
        eg(j)= qv(j,2)/dg(j);
        ev(j)=1-eg(j);
        %    Reset average conservative varibles in cells with area included
in the variables.
        qv(j,1)=qv(j,1)*area(j);
        qv(j,2)=qv(j,2)*area(j);
    end  % end of loop

    %    Below we find the phase velocities by combining the
    %    conservative variable defined by the mixture momentum equation
    %    with the gas slip relation.
    %    At the same time we try to summarize the gas volume in the well.
This
    %    also measure the size of the kick.
    gasvol=0;
    for j=1:nobox
        % The  interpolations introduced below are included
        % to omit a singularity in the slip relation when the gas volume
        % fraction becomes equal to 1/K. In additon, S is interpolated to
        % zero when approaching one phase gas flow. In the transition to
        % one phase gas flow, we have no slip condtions (K=1, S=0)
        % We will let the k0,s0,k1,s1 be arrays to make it easier to
incorporate
        % different flow regimes later.  In that case, the slip parameters
will
        % vary from cell to cell and we must have slip parameter values for
each
        % cell.
        ktemp=k;
        stemp=s;
        k0(j) = ktemp;
        s0(j) = stemp;
        % Interpolation to handle that (1-Kxgasvolumefraction) does not
become zero
        if ((eg(j)>=0.7) & (eg(j)<=0.8))
            xint = (eg(j)-0.7)/0.1;
            k0(j) =1.0*xint+k*(1-xint);
        elseif(eg(j)>0.8)
            k0(j)=1.0;
        end
        % Interpolate S to zero in transition to pure gas phase
        if ((eg(j)>=0.9) & (eg(j)<=1.0))
            xint = (eg(j)-0.9)/0.1;
            s0(j) = 0.0*xint+s*(1-xint);
        end
        % Note that the interpolations above and below can be changed
        % if numerical stability problems
        % are encountered.
        if (eg(j)>=0.999999)
            % Pure gas
            k1(j) = 1.0;
            s1(j) = 0.0;
        else
            %Two phase flow
            k1(j) = (1-k0(j)*eg(j))/(1-eg(j));
            s1(j) = -1.0*s0(j)*eg(j)/(1-eg(j));
        end
        help1 = dl(j)*ev(j)*k1+dg(j)*eg(j)*k0;
        help2 = dl(j)*ev(j)*s1+dg(j)*eg(j)*s0;

        vmixhelpl = (qv(j,3)/area(j)-help2)/help1;
        vg(j)=k0(j)*vmixhelpl+s0(j);
        vl(j)=k1(j)*vmixhelpl+s1(j);

        % Variable for summarizing the gas volume content in the well.
        gasvol=gasvol+eg(j)*area(j)*dx;
    end
    % Old values are now set equal to new values in order to prepare
    % computation of next time level.
    po=p;
    dlo=dl;
    dgo=dg;
    vlo=vl;
    vgo=vg;
    ego=eg;
    evo=ev;
    qvo=qv;

    % Section where we save some timedependent variables in arrays.
    % e.g. the bottomhole pressure. They will be saved for certain
    % timeintervalls defined in the start of the program in order to ensure
    % that the arrays do not get to long!
    if (counter>=nostepsbeforesavingtimedata)
        printcounter=printcounter+1;
        time  % Write time to screen.

        % Outlet massrates (kg/s) vs time

liquidmassrateout(printcounter)=dl(nobox)*ev(nobox)*vl(nobox)*area(nobox);

gasmassrateout(printcounter)=dg(nobox)*eg(nobox)*vg(nobox)*area(nobox);

        % Outlet flowrates (lpm) vs time
        liquidflowrateout(printcounter)=liquidmassrateout(printcounter)/...
            rholiq(P0,T0)*1000*60;
        gasflowrateout(printcounter)=gasmassrateout(printcounter)/...
            rogas(P0,T0)*1000*60;

        % Hydrostatic and friction pressure (bar) in well vs time
        hyd(printcounter)=sumhyd/100000;
        fric(printcounter)=sumfric/100000;

        % Volume of gas in well vs time (m3). Also used for indicating kick
        % size in well.
        volgas(printcounter)=gasvol;

        % Total phase masses (kg) in the well vs time
        % Used for checking mass conservation.
        massgas(printcounter)=gasmass;
        massliq(printcounter)=liqmass;

        % pout calculates the pressure at the outletboundary. I.e. upper edge
        % of uppermost cell. Corresponds where the well ends at surface. The
        % reason we do this is the fact than in AUSMV is all variables
defined
        % in the mid point of the numerical cells.
        pout(printcounter)=(p(nobox)-0.5*dx*...
            (dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g-
dx*0.5*fricgrad(nobox))/100000;

        % pin (bar) defines the  pressure at the inlet boundary, I.e lower
edge
        % of the lowermost cell. Corresponds to TD of well.
        pin(printcounter)=
(p(1)+0.5*dx*(dlo(1)*evo(1)+dgo(1)*ego(1))*g+0.5*dx*fricgrad(1))/100000;

        % Pressure in the middle of top box (bar).
        pnobox(printcounter)=p(nobox)/100000;  %

        % Time variable
        timeplot(printcounter)=time;
        counter = 0;
    end
end
% end of stepping forward in time.
% Printing of resultssection
countsteps % Marks number of simulation steps.

% Plot commands for variables vs time. The commands can also
% be copied to command screen where program is run for plotting other
% variables.
toc,
e = cputime-t

% Plot bottomhole pressure
% plot(timeplot,pin)
% Show cfl number used.
disp('cfl')
cfl = al*dt/dx

%plot(timeplot,pin)
%plot(timeplot,hyd)
%plot(timeplot,fric)
%plot(timeplot,liquidmassrateout)
%plot(timeplot,gasmassrateout)
% plot(timeplot,volgas)
%plot(timeplot,liquidflowrateout)
%plot(timeplot,gasflowrateout)
%plot(timeplot,massgas)
%plot(timeplot,massliq)
%plot(timeplot,pout)
%plot(timeplot,pnobox)
%Plot commands for variables vs depth/Only the last simulated
%values at endtime is visualised
% %plot(vl,x);
% plot(vg,x);
% axis([-1 0 -4 0])
% plot(eg,x);
%plot(p,x);
%plot(dl,x);
%plot(dg,x);
```

Appendix B

Appendix 0-1: Re-Start File Code

```
clear
% load('Cells100_xp_0p2_6p25_0p606.mat')
% load('7p5_0p606.mat')
% load('Cells100_xp_0p2_6p25_0p735.mat')
% load('7p5_0p735.mat')
% load('Cells100_xp_0p2_6p25_0p896.mat')
% load('7p5_0p896.mat')
% load('Cells100_xp_0p2_6p25_1p079.mat')
% load('7p5_1p079.mat')
% load('Cells25_xp_1p0_6p25_1p079.mat')
% load('1p0_7p5_1p079.mat')
% load('Cells100_xp_0p2_6p25_1p079_s0p5.mat')
% load('7p5_1p079_s0p5.mat')
% load('Cells100_xp_0p2_6p25_0p606_s0p5.mat')
load('7p5_0p606_s0p5.mat')

starting = countsteps+1
endtime = 8.0;
ending = endtime/dt;

viscl = 0.2;
viscg = 0.02;


% Now comes the for loop that runs forward in time. This is repeated for
% every timestep.

for i = starting:ending
   countsteps=countsteps+1;
   counter=counter+1;
   time = time+dt;  % Step one timestep and update time.
% Then a section where specify the boundary conditions.
% Here we specify the inlet rates of the different phases at the
% bottom of the pipe in kg/s. We interpolate to make things smooth.
% It is also possible to change the outlet boundary status of the well
% here. First we specify rates at the bottom and the pressure at the outlet
% in case we have an open well. This is a place where we can change the
% code to control simulations. If the well shall be close, wellopening must
% be set to 0. It is also possible to reverse the flow (bullheading).

% In the example below, we take a gas kick and then circulate this
% out of the well without closing the well. (how you not should perform
% well control)

% Note there are two variables wellopening and bullheading that can be
% changed in the control structure below to close the well or start
% reversing the flow i.e. pumping downwards.

% Note that if we will change to bullheading throughout the control stucture,
% the variable inletligmassrate
% has to be defined as negative since pumping downwards at outlet will be
% in negative direction (postive direction of flow has been chosen to be
% upwards)

% NB, NOTE THAT THIS IS ONE OF THE MAIN PLACES WHERE YOU HAVE TO ADJUST THE
% CODE TO CONTROL THE SIMULATION SCENARIO.
inletligmassrate=0.0;
inletgasmassrate=0.0;

 %test 1
 XX = 0.000295;
 bullheadingrate  = 0.606;  % Note this is kg/s or l/s

 % test 2
 % XX = 0.000296;
 % bullheadingrate  = 0.735;
 % test 3
 % XX = 0.000305;
 % bullheadingrate  = 0.896;

 % test 4
 % XX = 0.000276;
 % bullheadingrate  = 1.079;

xp = 0.2;
XX = XX/xp;

if (time>0)&&(time<0.25*xp)
    inletligmassrate=0.0;
    inletgasmassrate=XX*time/(0.25*xp);
elseif (time>0.25*xp)&&(time<xp)
    inletligmassrate=0.0;
    inletgasmassrate=XX;
elseif (time>=xp)&&(time<1.25*xp)
    inletligmassrate=0.0;
    inletgasmassrate=XX-XX*(time-xp)/(0.25*xp);
elseif (time>1.25*xp)&&(time<6.25)
    wellopening = 0.0;    % This is the closed in period
    inletligmassrate=0.0;
    inletgasmassrate=0.0;

% if (time>=0)&&(time <0.25)
%     inletligmassrate=0.0;
%     inletgasmassrate=XX*time/0.25;
% elseif (time>0.25)&&(time<1.0)
%     inletligmassrate=0.0;
%     inletgasmassrate=XX;
% elseif (time>1.0)&&(time<1.25)
%     inletligmassrate=0.0;
%     inletgasmassrate=XX-XX*(time-1.0)/0.25;
% elseif (time>1.25)&&(time<6.25)
%     wellopening = 0.0;    % This is the closed in period
%     inletligmassrate=0.0;
%     inletgasmassrate=0.0;
elseif((time>6.25)&&(time<6.75))
    wellopening = 1.0;  % Here we start ramping up the bullheading
    bullheading = 1.0;
    inletligmassrate=-1.0*bullheadingrate*(time-6.25)/0.5;
    inletgasmassrate=0.0;
    intparam = (time-6.25)/(6.75-6.25);
    k = 1.2*(1-intparam)+1.12*intparam; % Here we use timeinterpolation
    % to change the slip parameter from K = 1.2 to K = 1.12 before
    % bullheading. We need smooth transition of parameters.
    % Lack the bullvolume
elseif (time>6.75)
    k = 1.12;
    wellopening = 1.0;
    bullheading = 1.0;
    inletligmassrate=-1.0*bullheadingrate;
    inletgasmassrate=0.0;
    % Lack bullvolume
end

% XX = 4; % Gasrate in kg/s
%
% YY= 40; % Liquidrate in kg/s
%
% if (time < 10)
%
%   inletligmassrate=0.0;
%   inletgasmassrate=0.0;
%
% elseif ((time>=10) & (time < 20))
%   inletligmassrate = YY*(time-10)/10;  % Interpolate the rate from 0 to
value wanted.
%   inletgasmassrate = XX*(time-10)/10;
%
% elseif ((time >=20) && (time<200))
%   inletligmassrate = YY;
%   inletgasmassrate = XX;
% elseif ((time >=200) & (time<210))
% %  inletligmassrate = YY-YY*(time-200)/10;
%   inletligmassrate = YY-YY*(time-200)/10;
%   inletgasmassrate = XX-XX*(time-200)/10;
% elseif (time > 210)
%   inletligmassrate=0;
%   inletgasmassrate=0;
%   wellopening = 0;
% end

% The commented code below are from some previous runs. It shows. e.g. how
% we can close the well.
%elseif((time>=500)&(time<510))
%   inletligmassrate = YY-YY*(time-500)/10;
%   inletgasmassrate = XX-XX*(time-500)/10;
% elseif(time>=510)
%   inletligmassrate=0;
%   inletgasmassrate=0;
%   wellopening=0.0;
% end


%XX = 4;
% XX (kg/s) is a variable for introducing a kick in the well.
%YY = 15; % Liquid flowrate (kg/s) (1 kg/s = 1 l/s approx)
% if (time < 10)
%
```

```
%   inletligmassrate=0.0;
%   inletgasmassrate=0.0;
%
% elseif ((time>=10) & (time < 20))
%   inletligmassrate = 0*(time-10)/10;
%   inletgasmassrate = XX*(time-10)/10;
%
% elseif ((time >=20) & (time<110))
%   inletligmassrate = 0;
%   inletgasmassrate = XX;
%
% elseif ((time>=110)& (time<120))
%   inletligmassrate = 0;
%   inletgasmassrate = XX-XX*(time-110)/10;
% elseif ((time>=120&time<130))
%   inletligmassrate =0;
%   inletgasmassrate =0;
% elseif ((time>=130)&(time<300))
%   inletligmassrate =0;
%   inletgasmassrate =0;
% elseif ((time>=300)&(time<310))
%   inletligmassrate= YY*(time-300)/10;
%   inletgasmassrate =0;
% elseif((time>=310))
%   inletligmassrate= YY;
%   inletgasmassrate =0;
% end

kickvolume = kickvolume+inletgasmassrate/dgo(1)*dt;  % Here we find the
kickvolume

% initially induced in the well.

% Here we specify the physical outlet pressure. Here we have given the
pressure as
% constant. It would be possible to adjust it during openwell conditions
% either by giving the wanted pressure directly (in the command lines
% above) or by finding it indirectly through a chokemodel where the
chokeopening
% would have had to be  an input parameter. The chokeopening variable would
equally had
% to be adjusted inside the controle structure given above.

 pressureoutlet = pbondout;

% Based on these given physical boundary values combined with use
% of extrapolations techniques
% for the remaining unknowns at the boundaries, we will define the mass and
% momentum fluxes at the boundaries (inlet and outlet of pipe).

% inlet/bottom fluxes first.
   if (bullheading<=0)
 % Here we pump from bottom
     flc(1,1)= inletligmassrate/area(1);
     flc(1,2)= 0.0;
     flc(1,3)= flc(1,1)*vlo(1);


     fgc(1,1)= 0.0;
     fgc(1,2)= inletgasmassrate/area(1);
     fgc(1,3)= fgc(1,2)*vgo(1);

     fp(1,1)= 0.0;
     fp(1,2)= 0.0;

% Old way of treating the boundary
%     fp(1,3)= po(1)+0.5*(po(1)-po(2)); %Interpolation used to find the
% pressure at the inlet/bottom of the well.

% New way of treating the boundary
      fp(1,3)= po(1)...
            +0.5*dx*(dlo(1)*evo(1)+dgo(1)*ego(1))*g...
            +0.5*dx*fricgrad(1);
   else
     % Here we pump from the top. All masses are assumed to flow out of the
     % well into the formation. We use first order extrapolation.
     flc(1,1)=dlo(1)*evo(1)*vlo(1);
     flc(1,2)=0.0;
     flc(1,3)=flc(1,1)*vlo(1);

     fgc(1,1)=0.0;
     fgc(1,2)=dgo(1)*ego(1)*vgo(1);
     fgc(1,3)=fgc(1,2)*vgo(1);

     fp(1,1)=0.0;
     fp(1,2)=0.0;
     fp(1,3)=135000;   % This was a fixed pressure set at bottom when
bullheading (Pascal)
   end
% Outlet fluxes (open & closed conditions)

    if (wellopening>0.01)

% Here open end condtions are given. We distinguish between bullheading
% & normal circulation.

        if (bullheading<=0)  % Here we dont bullhead, i.e we circulate from
bottom

          % Here the is normal ciruclation and open well)
            flc(nofluxes,1)= dlo(nobox)*evo(nobox)*vlo(nobox);
            flc(nofluxes,2)= 0.0;
            flc(nofluxes,3)= flc(nofluxes,1)*vlo(nobox);

            fgc(nofluxes,1)= 0.0;
            fgc(nofluxes,2)= dgo(nobox)*ego(nobox)*vgo(nobox);
   %         fgc(nofluxes,2)=0; Activate if gas is sucked in!?
            fgc(nofluxes,3)= fgc(nofluxes,2)*vgo(nobox);

            fp(nofluxes,1)= 0.0;
            fp(nofluxes,2)= 0.0;
            fp(nofluxes,3)= pressureoutlet;
        else
            % Here we are bullheading.
            flc(nofluxes,1)= inletligmassrate/area(nobox);
            flc(nofluxes,2)= 0.0;
            flc(nofluxes,3)= flc(nofluxes,1)*vlo(nobox);
            fgc(nofluxes,1)=0.0;
            fgc(nofluxes,2)=0.0;
            fgc(nofluxes,3)=0.0;

            fp(nofluxes,1)=0.0;
            fp(nofluxes,2)=0.0;
            fp(nofluxes,3)= po(nobox)...
            -0.5*dx*(dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g...
            +0.5*dx*fricgrad(nobox); %check sign here on friction
            % Physcially, the friction should be added when going from
            % mid point in upper cell to outlet. But if fricgrad(nobox) is
            % negative there should be a minus in front of the term to have
            % + in the end.
        end
    else

% Here closed end conditions are given

         flc(nofluxes,1)= 0.0;
         flc(nofluxes,2)= 0.0;
         flc(nofluxes,3)= 0.0;

         fgc(nofluxes,1)= 0.0;
         fgc(nofluxes,2)= 0.0;
         fgc(nofluxes,3)= 0.0;

         fp(nofluxes,1)=0.0;
         fp(nofluxes,2)=0.0;

    %    Old way of treating the boundary
    %     fp(nofluxes,3)= po(nobox)-0.5*(po(nobox-1)-po(nobox));

    %    New way of treating the boundary
         fp(nofluxes,3)= po(nobox)...
         -0.5*dx*(dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g;
    %     -0.5*dx*fricgrad(nobox); % Neglect friction since well is closed.
        end
 % Implementation of slopelimiters. They are applied on the physical
 % variables like phase densities, phase velocities and pressure.

% It was found that if the slopelimiters were set to zero in
% the boundary cells, the pressure in these became wrong. E.g. the upper
% cell get an interior pressure that is higher than it should be e.g. when
% being static (hydrostatic pressure was too high). The problem was reduced
% by copying the slopelimiters from the interior cells. However, both
% approaches seems to give the same BHP pressure vs time but the latter
% approach give a more correct pressure vs depth profile. It is also better
% to use when simulating pressure build up where the upper cell pressure
% must be monitored. It should be checked more in detail before concluding.
% BUT; there has been mass conservation problems with the scheme for the
% case where the slopelimiters were copied (see master thesis of Keino)
% A possible fix has been included below where the slopelimiter related to
% the gas volume fraction is set to zero in the first cell.

     for i=2:nobox-1
      sl1(i)=minmod(dlo(i-1),dlo(i),dlo(i+1),dx);
      sl2(i)=minmod(po(i-1),po(i),po(i+1),dx);
      sl3(i)=minmod(vlo(i-1),vlo(i),vlo(i+1),dx);
      sl4(i)=minmod(vgo(i-1),vgo(i),vgo(i+1),dx);
      sl5(i)=minmod(ego(i-1),ego(i),ego(i+1),dx);
      sl6(i)=minmod(dgo(i-1),dgo(i),dgo(i+1),dx);
     end

 % Slopelimiters in outlet boundary cell are set to zero!
%      sl1(nobox)=0;
%      sl2(nobox)=0;
%      sl3(nobox)=0;
%      sl4(nobox)=0;
%      sl5(nobox)=0;
%      sl6(nobox)=0;

 % Slopelimiters in outlet boundary cell are copied from neighbour cell!
     sl1(nobox)=sl1(nobox-1);
     sl2(nobox)=sl2(nobox-1);
     sl3(nobox)=sl3(nobox-1);
     sl4(nobox)=sl4(nobox-1);
     sl5(nobox)=sl5(nobox-1);
     sl6(nobox)=sl6(nobox-1);

% Slopelimiters in inlet boundary cell are set to zero!
%      sl1(1)=0;
%      sl2(1)=0;
%      sl3(1)=0;
%      sl4(1)=0;
%      sl5(1)=0;
%      sl6(1)=0;

% Slopelimiters in inlet boundary cell are copied from neighbour cell!
     sl1(1)=sl1(2);
     sl2(1)=sl2(2);
     sl3(1)=sl3(2);
     sl4(1)=sl4(2);
     sl5(1)=sl5(2);
     sl6(1)=sl6(2);

% FIX FOR OMITTING THE GAS MASS CONSERVATION PROBLEM
     sl5(1)=0;


% Now we will find the fluxes between the different cells.
% NB - IMPORTANE -  Note that if we change the compressibilities/sound
velocities of
% the fluids involved, we may need to do changes inside the csound function.
% But the effect of this is unclear.

     for j = 2:nofluxes-1

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % First order method is from here: If you want to test this, activate this
 % and comment the second order code below.
%        cl = csound(ego(j-1),po(j-1),dlo(j-1),k);
%        cr = csound(ego(j),po(j),dlo(j),k);
%        c = max(cl,cr);
%        pll = psip(vlo(j-1),c,evo(j));
%        plr = psim(vlo(j),c,evo(j-1));
%        pgl = psip(vgo(j-1),c,ego(j));
%        pgr = psim(vgo(j),c,ego(j-1));
%        vmixr = vlo(j)*evo(j)+vgo(j)*ego(j);
%        vmixl = vlo(j-1)*evo(j-1)+vgo(j-1)*ego(j-1);
%
%        pl = pp(vmixl,c);
%        pr = pm(vmixr,c);
%        mll= evo(j-1)*dlo(j-1);
%        mlr= evo(j)*dlo(j);
%        mgl= ego(j-1)*dgo(j-1);
%        mgr= ego(j)*dgo(j);
%
%        flc(j,1)= mll*pll+mlr*plr;
%        flc(j,2)= 0.0;
%        flc(j,3)= mll*pll*vlo(j-1)+mlr*plr*vlo(j);
%
%        fgc(j,1)=0.0;
%        fgc(j,2)= mgl*pgl+mgr*pgr;
%        fgc(j,3)= mgl*pgl*vgo(j-1)+mgr*pgr*vgo(j);
%
%        fp(j,1)= 0.0;
%        fp(j,2)= 0.0;
%        fp(j,3)= pl*po(j-1)+pr*po(j);

 %  First order methods ends here
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Second order method starts here:
% Here slopelimiter is used on all variables except phase velocoties

       psll = po(j-1)+dx/2*sl2(j-1);
       pslr = po(j)-dx/2*sl2(j);
       dsll = dlo(j-1)+dx/2*sl1(j-1);
       dslr = dlo(j)-dx/2*sl1(j);
       dgll = dgo(j-1)+dx/2*sl6(j-1);
       dglr = dgo(j)-dx/2*sl6(j);

       vlv = vlo(j-1)+dx/2*sl3(j-1);
       vlh = vlo(j)-dx/2*sl3(j);
       vgv = vgo(j-1)+dx/2*sl4(j-1);
       vgh = vgo(j)-dx/2*sl4(j);

       gvv = ego(j-1)+dx/2*sl5(j-1);
       gvh = ego(j)-dx/2*sl5(j);
       lvv = 1-gvv;
       lvh = 1-gvh;

       cl = csound(gvv,psll,dsll,k);
       cr = csound(gvh,pslr,dslr,k);
       c = max(cl,cr);

       pll = psip(vlo(j-1),c,lvh);
       plr = psim(vlo(j),c,lvv);
       pgl = psip(vgo(j-1),c,gvh);
       pgr = psim(vgo(j),c,gvv);
       vmixr = vlo(j)*lvh+vgo(j)*gvh;
       vmixl = vlo(j-1)*lvv+vgo(j-1)*gvv;

       pl = pp(vmixl,c);
       pr = pm(vmixr,c);


      mll= lvv*dsll;
      mlr= lvh*dslr;
      mgl= gvv*dgll;
      mgr= gvh*dglr;

      flc(j,1)= mll*pll+mlr*plr;
      flc(j,2)= 0.0;
      flc(j,3)= mll*pll*vlo(j-1)+mlr*plr*vlo(j);


      fgc(j,1)=0.0;
      fgc(j,2)= mgl*pgl+mgr*pgr;
      fgc(j,3)= mgl*pgl*vgo(j-1)+mgr*pgr*vgo(j);

      fp(j,1)= 0.0;
      fp(j,2)= 0.0;
      fp(j,3)= pl*psll+pr*pslr;

%%% Second order method ends here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Here sloplimiters is used on all variables. This
% has not worked so well yet. Therefore it is commented away.

%       psll = po(j-1)+dx/2*sl2(j-1);
%       pslr = po(j)-dx/2*sl2(j);
%       dsll = dlo(j-1)+dx/2*sl1(j-1);
%       dslr = dlo(j)-dx/2*sl1(j);
%       dgll = dgo(j-1)+dx/2*sl6(j-1);
%       dglr = dgo(j)-dx/2*sl6(j);
%
%       vlv = vlo(j-1)+dx/2*sl3(j-1);
%       vlh = vlo(j)-dx/2*sl3(j);
%       vgv = vgo(j-1)+dx/2*sl4(j-1);
%       vgh = vgo(j)-dx/2*sl4(j);
%
%       gvv = ego(j-1)+dx/2*sl5(j-1);
%       gvh = ego(j)-dx/2*sl5(j);
%       lvv = 1-gvv;
%       lvh = 1-gvh;
%
%       cl = csound(gvv,psll,dsll,k);
%       cr = csound(gvh,pslr,dslr,k);
%       c = max(cl,cr);
%
%       pll = psip(vlv,c,lvh);
%       plr = psim(vlh,c,lvv);
%       pgl = psip(vgv,c,gvh);
%       pgr = psim(vgh,c,gvv);
%       vmixr = vlh*lvh+vgh*gvh;
%       vmixl = vlv*lvv+vgv*gvv;
%
%       pl = pp(vmixl,c);
%       pr = pm(vmixr,c);
%       mll= lvv*dsll;
%       mlr= lvh*dslr;
%       mgl= gvv*dgll;
%       mgr= gvh*dglr;
%
%       flc(j,1)= mll*pll+mlr*plr;
%       flc(j,2)= 0.0;
%       flc(j,3)= mll*pll*vlv+mlr*plr*vlh;
%
%
%       fgc(j,1)=0.0;
%       fgc(j,2)= mgl*pgl+mgr*pgr;
%       fgc(j,3)= mgl*pgl*vgv+mgr*pgr*vgh;
%
%       fp(j,1)= 0.0;
%       fp(j,2)= 0.0;
%       fp(j,3)= pl*psll+pr*pslr;
     end

% Fluxes have now been calculated. We will now update the conservative
% variables in each of the numerical cells.


% The source terms can be calculated by using a
% for loop.
% Note that the model is sensitive to how we treat the model
% for low Reynolds numbers (possible discontinuity in the model)
       for j=1:nobox
        fricgrad(j)=dpfric(vlo(j),vgo(j),evo(j),ego(j),dlo(j),dgo(j), ...
          po(j),do(j),di(j),viscl,viscg); % Pa/m
        hydgrad(j)=g*(dlo(j)*evo(j)+dgo(j)*ego(j)); % Pa/m
       end

      sumfric = 0;
      sumhyd= 0;

      for j=1:nobox

     % Here we solve the three conservation laws for each cell and update
     % the conservative variables qv

       ar = area(j);


      % Liquid mass conservation
      qv(j,1)=qvo(j,1)-dtdx*((ar*flc(j+1,1)-ar*flc(j,1))...
                            +(ar*fgc(j+1,1)-ar*fgc(j,1))...
                            +(ar*fp(j+1,1)-ar*fp(j,1)));

      % Gas mass conservation:

      qv(j,2)=qvo(j,2)-dtdx*((ar*flc(j+1,2)-ar*flc(j,2))...
                            +(ar*fgc(j+1,2)-ar*fgc(j,2))...
                            +(ar*fp(j+1,2)-ar*fp(j,2)));
      % Mixture momentum conservation:

      qv(j,3)=qvo(j,3)-dtdx*((ar*flc(j+1,3)-ar*flc(j,3))...
                            +(ar*fgc(j+1,3)-ar*fgc(j,3))...
                            +(ar*fp(j+1,3)-ar*fp(j,3)))...
                   -dt*ar*(fricgrad(j)+hydgrad(j));
```

```
%  Add up the hydrostatic pressure  and friction  in the whole well.
      sumfric=sumfric+fricgrad(j)*dx;
      sumhyd=sumhyd+hydgrad(j)*dx;

      end

% Section where we find the physical variables (pressures, densities etc)
% from the conservative variables. Some trickes to ensure stability. These
% are induced to avoid negative masses.

     gasmass=0;
     liqmass=0;

     for j=1:nobox

% Remove the area from the conservative variables to find the
% the primitive variables from the conservative ones.

      qv(j,1)= qv(j,1)/area(j);
      qv(j,2)= qv(j,2)/area(j);

      if (qv(j,1)<0.00000001)  % Trick to avoid negative masses.
        qv(j,1)=0.00000001;
      end

      if (qv(j,2)< 0.00000001)  % Trick to avoid negative masses.
        qv(j,2)=0.00000001;
      end

 % Here we summarize the mass of gas and liquid in the well respectively.
 % These variables are important to show that the scheme is conserving
 % mass. (if e.g. gas leaks in our out of the well unintentionally in the
simulation
 % without being specified in the code,something fundamental is wrong.

      gasmass = gasmass+qv(j,2)*area(j)*dx;
      liqmass = liqmass+qv(j,1)*area(j)*dx;

% Below, we find the primitive variables pressure and densities based on
% the conservative variables q1,q2. One can choose between getting them by
% analytical or numerical solution approach specified in the beginning of
% the program. Ps. For more advanced density models, this must be changed.


      if (analytical == 1)
%       % Analytical solution:

% here the simple density models used in PET 510 Wellflow modelling
% compendium is used.

%        t1=rho0-P0/al^2;
%
% %  Coefficients:
%        a = 1/(al*al);
%        b = t1-qv(j,1)-rt*qv(j,2)/(al*al);
%        c = -1.0*t1*rt*qv(j,2);
% %
%
```

```
% %       Note here we use the very simple models from the PET510 course
%         p(j)=(-b+sqrt(b*b-4*a*c))/(2*a);  % Pressure
%         dl(j)=rholiq(p(j),temp(j)); % Density of liquid
%         dg(j)=rogas(p(j),temp(j)); % Density of gas

%     The code below can be activated if we want to switch to the other set
%     of density models. Also then remember to do the changes inside
%     functions rogas og rholiq if we change density models.

          x1=rho0-P0*rho0/Bheta-rho0*Alpha*(temp(j)-T0);
          x2=rho0/Bheta;
          x3=-qv(j,2)*R*temp(j);

          a = x2;
          b = x1+x2*x3-qv(j,1);
          c = x1*x3;

          p(j)=(-b+sqrt(b*b-4*a*c))/(2*a);  % Pressure
          dl(j)=rholiq(p(j),temp(j));
          dg(j)=rogas(p(j),temp(j));
      else


      %Numerical Solution: This might be used if we use more complex
      %density models. Has not been used for years.

       [p(j),error]=itsolver(po(j),qv(j,1),qv(j,2)); % Pressure
       dl(j)=rholiq(p(j),temp(j)); % Density of liquid
       dg(j)=rogas(p(j)); % Density of gas

      % Incase a numerical solution is not found, the program will write out
"error":
       if error > 0
          error
       end
      end

%   Find phase volume fractions
      eg(j)= qv(j,2)/dg(j);
      ev(j)=1-eg(j);

 %    Reset average conservative varibles in cells with area included in the
variables.

     qv(j,1)=qv(j,1)*area(j);
     qv(j,2)=qv(j,2)*area(j);

     end  % end of loop

 %    Below we find the phase velocities by combining the
 %    conservative variable defined by the mixture momentum equation
 %    with the gas slip relation.
 %    At the same time we try to summarize the gas volume in the well. This
 %    also measure the size of the kick.

   gasvol=0;

   for j=1:nobox
   % The  interpolations introduced below are included
   % to omit a singularity in the slip relation when the gas volume
   % fraction becomes equal to 1/K. In additon, S is interpolated to
   % zero when approaching one phase gas flow. In the transition to
   % one phase gas flow, we have no slip condtions (K=1, S=0)
   % We will let the k0,s0,k1,s1 be arrays to make it easier to incorporate
   % different flow regimes later.  In that case, the slip parameters will
   % vary from cell to cell and we must have slip parameter values for each
   % cell.

      ktemp=k;
      stemp=s;

      k0(j) = ktemp;
      s0(j) = stemp;

    % Interpolation to handle that (1-Kxgasvolumefraction) does not become
zero
      if ((eg(j)>=0.7) & (eg(j)<=0.8))
        xint = (eg(j)-0.7)/0.1;
        k0(j) =1.0*xint+k*(1-xint);
      elseif(eg(j)>0.8)
        k0(j)=1.0;
      end


    % Interpolate S to zero in transition to pure gas phase
       if ((eg(j)>=0.9) & (eg(j)<=1.0))
         xint = (eg(j)-0.9)/0.1;
         s0(j) = 0.0*xint+s*(1-xint);
       end

  % Note that the interpolations above and below can be changed
  % if numerical stability problems
  % are encountered.

  %
      if (eg(j)>=0.999999)
       % Pure gas
        k1(j) = 1.0;
        s1(j) = 0.0;
      else
        %Two phase flow
        k1(j) = (1-k0(j)*eg(j))/(1-eg(j));
        s1(j) = -1.0*s0(j)*eg(j)/(1-eg(j));
      end

      help1 = dl(j)*ev(j)*k1+dg(j)*eg(j)*k0;
      help2 = dl(j)*ev(j)*s1+dg(j)*eg(j)*s0;


      vmixhelpl = (qv(j,3)/area(j)-help2)/help1;
      vg(j)=k0(j)*vmixhelpl+s0(j);
      vl(j)=k1(j)*vmixhelpl+s1(j);
      % Variable for summarizing the gas volume content in the well.
      gasvol=gasvol+eg(j)*area(j)*dx;

   end

% Old values are now set equal to new values in order to prepare
% computation of next time level.


   po=p;
   dlo=dl;
   dgo=dg;
   vlo=vl;
   vgo=vg;
   ego=eg;
   evo=ev;
   qvo=qv;

% Section where we save some timedependent variables in arrays.
% e.g. the bottomhole pressure. They will be saved for certain
% timeintervalls defined in the start of the program in order to ensure
% that the arrays do not get to long!

  if (counter>=nostepsbeforesavingtimedata)
    printcounter=printcounter+1;
    time  % Write time to screen.

    % Outlet massrates (kg/s) vs time

liquidmassrateout(printcounter)=dl(nobox)*ev(nobox)*vl(nobox)*area(nobox);
    gasmassrateout(printcounter)=dg(nobox)*eg(nobox)*vg(nobox)*area(nobox);

    % Outlet flowrates (lpm) vs time
    liquidflowrateout(printcounter)=liquidmassrateout(printcounter)/...
        rholiq(P0,T0)*1000*60;
    gasflowrateout(printcounter)=gasmassrateout(printcounter)/...
        rogas(P0,T0)*1000*60;

    % Hydrostatic and friction pressure (bar) in well vs time
    hyd(printcounter)=sumhyd/100000;
    fric(printcounter)=sumfric/100000;

    % Volume of gas in well vs time (m3). Also used for indicating kick
    % size in well.

    volgas(printcounter)=gasvol;

    % Total phase masses (kg) in the well vs time
    % Used for checking mass conservation.

     massgas(printcounter)=gasmass;
     massliq(printcounter)=liqmass;


    % pout calculates the pressure at the outletboundary. I.e. upper edge
    % of uppermost cell. Corresponds where the well ends at surface. The
    % reason we do this is the fact than in AUSMV is all variables defined
    % in the mid point of the numerical cells.
    pout(printcounter)=(p(nobox)-0.5*dx*...
    (dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g-
dx*0.5*fricgrad(nobox))/100000;

    % pin (bar) defines the  pressure at the inlet boundary, I.e lower edge
    % of the lowermost cell. Corresponds to TD of well.
    pin(printcounter)=
(p(1)+0.5*dx*(dlo(1)*evo(1)+dgo(1)*ego(1))*g+0.5*dx*fricgrad(1))/100000;

    % Pressure in the middle of top box (bar).
    pnobox(printcounter)=p(nobox)/100000;  %


    % Time variable
    timeplot(printcounter)=time;

    counter = 0;

  end
end

% end of stepping forward in time.

% Printing of resultssection

countsteps % Marks number of simulation steps.

% Plot commands for variables vs time. The commands can also
% be copied to command screen where program is run for plotting other
% variables.

% Plot bottomhole pressure
plot(timeplot,pin)

% Show cfl number used.
disp('cfl')
cfl = al*dt/dx

 plot(timeplot,pin)
%plot(timeplot,hyd)
%plot(timeplot,fric)
%plot(timeplot,liquidmassrateout)
%plot(timeplot,gasmassrateout)
%plot(timeplot,volgas)
%plot(timeplot,liquidflowrateout)
%plot(timeplot,gasflowrateout)
%plot(timeplot,massgas)
%plot(timeplot,massliq)
%plot(timeplot,pout)
%plot(timeplot,pnobox)

%Plot commands for variables vs depth/Only the last simulated
%values at endtime is visualised

%plot(vl,x);
%plot(vg,x);
%plot(eg,x);
%plot(p,x);
%plot(dl,x);
%plot(dg,x);
```

Appendix C

Complete slip model

```
% Transient two-phase code based on AUSMV scheme: Gas and Water
% The code assumes uniform geometry

% time - Seconds

% p - pressure at new time level (Pa)
% dl - density of liquid at new time level (kg/m3)
% dg - density of gas at new time level (kg/m3)
% eg - phase volume fraction of gas at new time level (0-1)
% ev - phase volume fraction of liquid at new time level (0-1)
% vg - phase velocity of gas at new time level (m/s)
% vl - phase velocity of liquid at new time level (m/s)
% qv - conservative variables at new time level  ( 3 in each cell)
% temp - temperature in well (K)

% po - pressure at old time level (Pa)
% dlo - density of liquid at old time level (kg/m3)
% dgo - density of gas at new old level (kg/m3)
% ego - phase volume fraction of gas at old time level (0-1)
% evo - phase volume fraction of liquid at old time level (0-1)
% vgo - phase velocity of gas at old time level (m/s)
% vlo - phase velocity of liquid at old time level (m/s)
% qvo - conservative variables at old  time level  ( 3 in each cell)
% temp - temperature in well (K)

clear;
t = cputime
tic,

% Geometry data/ Must be specified
welldepth = 2000;
nobox = 50; %Number of boxes in the well

% Note that one can use more refined grid, 50, 100 boxes.
% When doing this, remember to reduce time step to keep the CFL number
% fixed below 0.25.. dt < cfl x dx/ speed of sound in water. If boxes are
% doubled, then half the time step.
nofluxes = nobox+1;  % Number of cell boundaries
dx = welldepth/nobox; % Boxlength
%dt = 0.005;

% Welldepth. Cell 1 start at bottom
x(1)= -1.0*welldepth+0.5*dx;
for i=1:nobox-1
    x(i+1)=x(i)+ dx;
end

% VERY IMPORTANT: BELOW THE TIMESTEP IS SET. MAKE SURE THAT THE
% CFL CONDIDTION IS FULFILLED. IF NUMBER OF BOXES IS CHANGED. DX WILL
% CHANGE AND DT HAS TO BE ADJUSTED TO KEEP THE CFL NUMBER FIXED.

% dt= 0.01;  % Timestep (seconds)
dt= 0.005;  % Timestep (seconds) for 50 nobox
% dt= 0.0025;  % Timestep (seconds) for 100 nobox
```

```
dtdx = dt/dx;
time = 0.0; % initial time.
endtime = 4000; % Time for ending simulation  (seconds)
nosteps = endtime/dt;  %Number of total timesteps. Used in for loop.
timebetweensavingtimedata = 5;  % How often in s we save data vs time for
plotting.
nostepsbeforesavingtimedata = timebetweensavingtimedata/dt;

% Slip parameters used in the gas slip relation. vg =Kvmix+S
k = 1.2;
s = 0.55;

% Boundary condition at outlet
pbondout=100000; % Pascal  (1 bar)

% Initial temperature distribution. (Kelvin)
% Note that this is only used if we use density models that depend on
% temperature
tempbot = 110+273;
temptop = 50+273;
tempgrad= (tempbot-temptop)/welldepth;
tempo(1)=tempbot-dx/2*tempgrad;
for i = 1:nobox-2
    tempo(i+1)=tempo(i)-dx*tempgrad;
end
tempo(nobox)=tempo(nobox-1)-dx*tempgrad;

temp = tempo;

% Different fluid density parameters
% Note how we switch between different models later.
% These parameters are used when finding the
% primitive variables pressure, densities in an analytical manner.
% Changing parameters here, you must also change parameters inside the
% density routines roliq and rogas.

% Simple Water density model & Ideal Gas. See worknote Extension of AUSMV
% scheme.
rho0=1000;  % Water density at STC (Standard Condition) kg/m3
Bheta=2.2*10^9; % Parameter that depend on the compressibility of water
Alpha=0.000207; % Parameter related to thermal expansion/compression
R = 286.9; % Ideal gas parameter
P0=100000; % Pressure at STC (Pa)
T0=15+273.15; % Temperature at STC (K)

% Very simple models (PET510 compendium)
al = 1500; % Speed of sound in water.
rt= 100000; % Ideal gas parameter in model rhog = p/rt  (rt = ag^2)
rho0=1000; % Water density at STC (Standard Condition) kg/m3
P0=100000; % Pressure at STC (Pa)
T0=15+273.15; % Temperature at STC (K)

% Viscosities (Pa*s)/Used in the frictional pressure loss model (dpfric).
viscl = 0.001; % Liquid phase
viscg = 0.0000182; % Gas phase

% Gravity constant
g = 9.81; % Gravitational constant m/s2
```

```
% Well opening. opening = 1, fully open well, opening = 0 (<0.01), the well
% is fully closed. This variable will control what boundary conditions that
% will apply at the outlet (both physical and numerical): We must change
% this further below in the code if we want to change status on this.

wellopening = 1.0;  % This variable determines if
%the well is closed or not, wellopening = 1.0 -> open. welllopening = 0
%-> Well is closed. This variable affects the boundary treatment.

bullheading = 0.0; % This variable can be set to 1.0 if we want to simulate
% a bullheading operation. But the normal is to set this to zero.

% Specify if the primitive variables shall be found either by
% a numerical or analytical approach. If analytical = 1, analytical
% solution is used. If analytical = 0. The numerical approach is used.
% using the itsolver subroutine where the bisection numerical method
% is used. We use analytical.

analytical = 1;

% Initialization of rest of geometry.
% Here we specify the outer and inner diameter and the flow area
% We assume 12.25 x 5 inch annulus. But this can be modified.
for i = 1:nobox
    do(i)=0.2159;
    di(i) = 0.127;
    area(i) = 3.14/4*(do(i)*do(i)- di(i)*di(i));
end

wellvolume=area(1)*welldepth;

% Initialization of slope limiters. These are used for
% reducing numerical diffusion and will be calculated for each timestep.
% They make the numerical scheme second order.
for i = 1:nobox
    sl1(i)=0;
    sl2(i)=0;
    sl3(i)=0;
    sl4(i)=0;
    sl5(i)=0;
    sl6(i)=0;
end

% Now comes the intialization of the physical variables in the well.
% First primitive variables, then the conservative ones.

% Below we intialize pressure and fluid densities. We start from top of
% the well and calculated downwards. The calculation is done twice with
% updated values to get better approximation. Only hydrostatic
% considerations since we start with a static well.
for i = 1:nobox
    eg(i)=0.0;  % Gas volume fraction
    ev(i)=1-eg(i); % Liquid volume fraction
end

p(nobox)= pbondout+0.5*9.81*dx*...
    (ev(nobox)*rholiq(P0,T0)+eg(nobox)*rogas(P0,T0));   % Pressure (Pa)
dl(nobox)=rholiq(p(nobox),tempo(nobox));  % Liquid density kg/m3
```

```
dg(nobox)=rogas(p(nobox),tempo(nobox));   % Gas density kg/m3

for i=nobox-1:-1:1
    p(i)=p(i+1)+dx*9.81*(ev(i+1)*dl(i+1)+eg(i+1)*dg(i+1));
    dl(i)=rholiq(p(i),tempo(i));
    dg(i)=rogas(p(i),tempo(i));
end

for i=nobox-1:-1:1
    rhoavg1= (ev(i+1)*dl(i+1)+eg(i+1)*dg(i+1));
    rhoavg2= (ev(i)*dl(i)+eg(i)*dg(i));
    p(i)=p(i+1)+dx*9.81*(rhoavg1+rhoavg2)*0.5;
    dl(i)=rholiq(p(i),tempo(i));
    dg(i)=rogas(p(i),tempo(i));
end

% Intitialize phase velocities, volume fractions, conservative variables
% and friction and hydrostatic gradients.
% The basic assumption is static fluid, one phase liquid.
for i = 1:nobox
    vl(i)=0; % Liquid velocity new time level.
    vg(i)=0; % Gas velocity at new time level
    eg(i)=0.0;  % Gas volume fraction
    ev(i)=1-eg(i); % Liquid volume fraction
    qv(i,1)=dl(i)*ev(i)*area(i);  % Conservative variable for liquid mass
(kg/m)
    qv(i,2)=dg(i)*eg(i)*area(i);  % Conservative variable for gas mass (kg/m)
    qv(i,3)=(dl(i)*ev(i)*vl(i)+dg(i)*eg(i)*vg(i))*area(i); % Conservative
variable for mixture moementum
    fricgrad(i)=0;   % Pa/m
    hydgrad(i)=g*(dl(i)*ev(i)+eg(i)*dg(i));  % Pa/m
end

% Section where we also initialize values at old time level
for i=1:nobox
    dlo(i)=dl(i);
    dgo(i)=dg(i);
    po(i)=p(i);
    ego(i)=eg(i);
    evo(i)=ev(i);
    vlo(i)=vl(i);
    vgo(i)=vg(i);
    qvo(i,1)=qv(i,1);
    qvo(i,2)=qv(i,2);
    qvo(i,3)=qv(i,3);
end

% Intialize fluxes between the cells/boxes
for i = 1:nofluxes
    for j =1:3
        flc(i,j)=0.0; % Flux of liquid over box boundary
        fgc(i,j)=0.0; % Flux of gas over box boundary
        fp(i,j)= 0.0; % Pressure flux over box boundary
    end
end

%  Main program. Here we will progress in time. First som intializations
% and definitions to take out results. The for loop below runs until the
% simulation is finished.
```

```
countsteps = 0;
counter=0;
printcounter = 1;
pin(printcounter) = (p(1)+dx*0.5*hydgrad(1))/100000; % Pressure in bar at
bottom for time storage
pout(printcounter)= pbondout/100000; % Pressure at outlet of uppermost cell
pnobox(printcounter)= p(nobox)/100000; % Pressure in middle of uppermost cell
liquidmassrateout(printcounter) = 0;  % liquid mass rate at outlet kg/s
gasmassrateout(printcounter)=0;   % gass mass rate at outlet kg/s
bulvol(printcounter)=0;  % Vector that stores accummulated bullheading volume
timeplot(printcounter)=time;  % Array for time and plotting of variables vs
time
pitvolume=0;
pitrate =0;
pitgain(printcounter)=0;

kickvolume=0;
bullvolume=0;

% The temperature is not updated but kept fixed according to the
% initialization.

% Now comes the for loop that runs forward in time. This is repeated for
% every timestep.
for i = 1:nosteps
    countsteps=countsteps+1;
    counter=counter+1;
    time = time+dt;  % Step one timestep and update time.

    % Then a section where specify the boundary conditions.
    % Here we specify the inlet rates of the different phases at the
    % bottom of the pipe in kg/s. We interpolate to make things smooth.
    % It is also possible to change the outlet boundary status of the well
    % here. First we specify rates at the bottom and the pressure at the
outlet
    % in case we have an open well. This is a place where we can change the
    % code to control simulations. If the well shall be close, wellopening
must
    % be set to 0. It is also possible to reverse the flow (bullheading).

    % In the example below, we take a gas kick and then circulate this
    % out of the well without closing the well. (how you not should perform
    % well control)

    % Note there are two variables wellopening and bullheading that can be
    % changed in the control structure below to close the well or start
    % reversing the flow i.e. pumping downwards.

    % Note that if we will change to bullheading throughout the control
stucture,
    % the variable inletligmassrate
    % has to be defined as negative since pumping downwards at outlet will be
    % in negative direction (postive direction of flow has been chosen to be
    % upwards)

    % NB, NOTE THAT THIS IS ONE OF THE MAIN PLACES WHERE YOU HAVE TO ADJUST
THE
    % CODE TO CONTROL THE SIMULATION SCENARIO.
    XX = 8.0; % Gasrate in kg/s
    %XX = 1;

    YY= 0; % Liquidrate in kg/s

    %YYB = 8.33;
    %YYB = 13;
    YYB=25;
    %YYB=33.33;
    %YYB = 11.751; %calculated critical bullheading rate

    if (time < 10)
        inletligmassrate=0.0;
        inletgasmassrate=0.0;
    elseif ((time>=10) & (time < 20))
        inletligmassrate = YY*(time-10)/10;  % Interpolate the rate from 0 to
value wanted.
        inletgasmassrate = XX*(time-10)/10;
    elseif ((time >=20) && (time<110))
        inletligmassrate = YY;
        inletgasmassrate = XX;
    elseif ((time >=110) & (time<120))
        %  inletligmassrate = YY-YY*(time-200)/10;
        inletligmassrate = YY-YY*(time-110)/10;
        inletgasmassrate = XX-XX*(time-110)/10;
    elseif ((time >=120) & (time<130))
        inletligmassrate=0;
        inletgasmassrate=0;
    elseif (time > 130)&(time<=3500)
        inletligmassrate=0;
        inletgasmassrate=0;
        wellopening = 0;
    elseif ((time>3500)&(time<3560))
        wellopening = 1.0;
        bullheading = 1.0;
        inletligmassrate=-1.0*YYB*(time-3500)/60;
        inletgasmassrate =0;
        xint = (time-3500)/60;
        k = 1.2*(1-xint)+xint*1.12;
        bullvolume = bullvolume+inletligmassrate/dlo(nobox)*dt*(-1.0);
    elseif((time>=3560))
        wellopening = 1.0;
        bullheading = 1.0;
        inletligmassrate=-1.0*YYB;
        inletgasmassrate =0;
        k = 1.12;
        bullvolume = bullvolume+inletligmassrate/dlo(nobox)*dt*(-1.0);
    end

    % The commented code below are from some previous runs. It shows. e.g.
how
    % we can close the well.
    %elseif((time>=500)&(time<510))
    %   inletligmassrate = YY-YY*(time-500)/10;
    %   inletgasmassrate = XX-XX*(time-500)/10;
    % elseif(time>=510)
    %   inletligmassrate=0;
    %   inletgasmassrate=0;
    %   wellopening=0.0;
    % end

    %XX = 4;
    % XX (kg/s) is a variable for introducing a kick in the well.
    %YY = 15; % Liquid flowrate (kg/s) (1 kg/s = 1 l/s approx)
    % if (time < 10)
    %
    %   inletligmassrate=0.0;
    %   inletgasmassrate=0.0;
    %
    % elseif ((time>=10) & (time < 20))
    %   inletligmassrate = 0*(time-10)/10;
    %   inletgasmassrate = XX*(time-10)/10;
    %
    % elseif ((time >=20) & (time<110))
    %   inletligmassrate = 0;
    %   inletgasmassrate = XX;
    %
    % elseif ((time>=110)& (time<120))
    %   inletligmassrate = 0;
    %   inletgasmassrate = XX-XX*(time-110)/10;
    % elseif ((time>=120&time<130))
    %   inletligmassrate =0;
    %   inletgasmassrate =0;
    % elseif ((time>=130)&(time<300))
    %   inletligmassrate =0;
    %   inletgasmassrate =0;
    % elseif ((time>=300)&(time<310))
    %   inletligmassrate= YY*(time-300)/10;
    %   inletgasmassrate =0;
    % elseif((time>=310))
    %   inletligmassrate= YY;
    %   inletgasmassrate =0;
    % end

    kickvolume = kickvolume+inletgasmassrate/dgo(1)*dt;  % Here we find the
kickvolume

    % initially induced in the well.

    % Here we specify the physical outlet pressure. Here we have given the
pressure as
    % constant. It would be possible to adjust it during openwell conditions
    % either by giving the wanted pressure directly (in the command lines
    % above) or by finding it indirectly through a chokemodel where the
chokeopening
    % would have had to be  an input parameter. The chokeopening variable
would equally had
    % to be adjusted inside the controle structure given above.

    pressureoutlet = pbondout;

    % Based on these given physical boundary values combined with use
    % of extrapolations techniques
    % for the remaining unknowns at the boundaries, we will define the mass
and
    % momentum fluxes at the boundaries (inlet and outlet of pipe).

    % inlet/bottom fluxes first.
    if (bullheading<=0)
        % Here we pump from bottom
        flc(1,1)= inletligmassrate/area(1);
        flc(1,2)= 0.0;
        flc(1,3)= flc(1,1)*vlo(1);

        fgc(1,1)= 0.0;
        fgc(1,2)= inletgasmassrate/area(1);
        fgc(1,3)= fgc(1,2)*vgo(1);

        fp(1,1)= 0.0;
        fp(1,2)= 0.0;

        % Old way of treating the boundary
        %     fp(1,3)= po(1)+0.5*(po(1)-po(2)); %Interpolation used to find
the
        % pressure at the inlet/bottom of the well.

        % New way of treating the boundary
        fp(1,3)= po(1)...
            +0.5*dx*(dlo(1)*evo(1)+dgo(1)*ego(1))*g...
            +0.5*dx*fricgrad(1);
    else
        % Here we pump from the top. All masses are assumed to flow out of
the
        % well into the formation. We use first order extrapolation.
        flc(1,1)=dlo(1)*evo(1)*vlo(1);
        flc(1,2)=0.0;
        flc(1,3)=flc(1,1)*vlo(1);

        fgc(1,1)=0.0;
        fgc(1,2)=dgo(1)*ego(1)*vgo(1);
        fgc(1,3)=fgc(1,2)*vgo(1);

        fp(1,1)=0.0;
        fp(1,2)=0.0;
        fp(1,3)=20000000;   % This was a fixed pressure set at bottom when
bullheading (Pa)
    end

    % Outlet fluxes (open & closed conditions)

    if (wellopening>0.01)

        % Here open end condtions are given. We distinguish between
bullheading
        % & normal circulation.

        if (bullheading<=0)  % Here we dont bullhead, i.e we circulate from
bottom
            % Here the is normal ciruclation and open well)
            flc(nofluxes,1)= dlo(nobox)*evo(nobox)*vlo(nobox);
            flc(nofluxes,2)= 0.0;
            flc(nofluxes,3)= flc(nofluxes,1)*vlo(nobox);

            fgc(nofluxes,1)= 0.0;
            fgc(nofluxes,2)= dgo(nobox)*ego(nobox)*vgo(nobox);
            %          fgc(nofluxes,2)=0; Activate if gas is sucked in!?
            fgc(nofluxes,3)= fgc(nofluxes,2)*vgo(nobox);
            fp(nofluxes,1)= 0.0;
            fp(nofluxes,2)= 0.0;
            fp(nofluxes,3)= pressureoutlet;
        else
            % Here we are bullheading.
            flc(nofluxes,1)= inletligmassrate/area(nobox);
            flc(nofluxes,2)= 0.0;
            flc(nofluxes,3)= flc(nofluxes,1)*vlo(nobox);

            fgc(nofluxes,1)=0.0;
            fgc(nofluxes,2)=0.0;
            fgc(nofluxes,3)=0.0;

            fp(nofluxes,1)=0.0;
            fp(nofluxes,2)=0.0;
            fp(nofluxes,3)= po(nobox)...
                -0.5*dx*(dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g...
                +0.5*dx*fricgrad(nobox); %check sign here on friction
            % Physcially, the friction should be added when going from
            % mid point in upper cell to outlet. But if fricgrad(nobox) is
            % negative there should be a minus in front of the term to have
            % + in the end.
        end
    else
        % Here closed end conditions are given
        flc(nofluxes,1)= 0.0;
        flc(nofluxes,2)= 0.0;
        flc(nofluxes,3)= 0.0;

        fgc(nofluxes,1)= 0.0;
        fgc(nofluxes,2)= 0.0;
        fgc(nofluxes,3)= 0.0;

        fp(nofluxes,1)=0.0;
        fp(nofluxes,2)=0.0;

        %    Old way of treating the boundary
        %     fp(nofluxes,3)= po(nobox)-0.5*(po(nobox-1)-po(nobox));

        %    New way of treating the boundary
        fp(nofluxes,3)= po(nobox)...
            -0.5*dx*(dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g;
        %     -0.5*dx*fricgrad(nobox); % Neglect friction since well is
closed.
    end

    % Implementation of slopelimiters. They are applied on the physical
    % variables like phase densities, phase velocities and pressure.

    % It was found that if the slopelimiters were set to zero in
    % the boundary cells, the pressure in these became wrong. E.g. the upper
    % cell get an interior pressure that is higher than it should be e.g.
when
    % being static (hydrostatic pressure was too high). The problem was
reduced
    % by copying the slopelimiters from the interior cells. However, both
    % approaches seems to give the same BHP pressure vs time but the latter
    % approach give a more correct pressure vs depth profile. It is also
better
    % to use when simulating pressure build up where the upper cell pressure
    % must be monitored. It should be checked more in detail before
concluding.
    % BUT; there has been mass conservation problems with the scheme for the
    % case where the slopelimiters were copied (see master thesis of Keino)
    % A possible fix has been included below where the slopelimiter related
to
    % the gas volume fraction is set to zero in the first cell.

    for i=2:nobox-1
        sl1(i)=minmod(dlo(i-1),dlo(i),dlo(i+1),dx);
        sl2(i)=minmod(po(i-1),po(i),po(i+1),dx);
        sl3(i)=minmod(vlo(i-1),vlo(i),vlo(i+1),dx);
        sl4(i)=minmod(vgo(i-1),vgo(i),vgo(i+1),dx);
        sl5(i)=minmod(ego(i-1),ego(i),ego(i+1),dx);
        sl6(i)=minmod(dgo(i-1),dgo(i),dgo(i+1),dx);
    end

    % Slopelimiters in outlet boundary cell are set to zero!
    %      sl1(nobox)=0;
    %      sl2(nobox)=0;
    %      sl3(nobox)=0;
    %      sl4(nobox)=0;
    %      sl5(nobox)=0;
    %      sl6(nobox)=0;

    % Slopelimiters in outlet boundary cell are copied from neighbour cell!
    sl1(nobox)=sl1(nobox-1);
    sl2(nobox)=sl2(nobox-1);
    sl3(nobox)=sl3(nobox-1);
    sl4(nobox)=sl4(nobox-1);
    sl5(nobox)=sl5(nobox-1);
    sl6(nobox)=sl6(nobox-1);

    % Slopelimiters in inlet boundary cell are set to zero!
    %      sl1(1)=0;
    %      sl2(1)=0;
    %      sl3(1)=0;
    %      sl4(1)=0;
    %      sl5(1)=0;
    %      sl6(1)=0;

    % Slopelimiters in inlet boundary cell are copied from neighbour cell!
    sl1(1)=sl1(2);
    sl2(1)=sl2(2);
    sl3(1)=sl3(2);
    sl4(1)=sl4(2);
    sl5(1)=sl5(2);
    sl6(1)=sl6(2);

    % FIX FOR OMITTING THE GAS MASS CONSERVATION PROBLEM
    sl5(1)=0;
    sl5(nobox)=0;

    % Now we will find the fluxes between the different cells.
    % NB - IMPORTANE -  Note that if we change the compressibilities/sound
velocities of
```

```
    % the fluids involved, we may need to do changes inside the csound
function.
    % But the effect of this is unclear.
    for j = 2:nofluxes-1
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % First order method is from here: If you want to test this, activate
        % this and comment the second order code below.
        %        cl = csound(ego(j-1),po(j-1),dlo(j-1),k);
        %        cr = csound(ego(j),po(j),dlo(j),k);
        %        c = max(cl,cr);
        %        pll = psip(vlo(j-1),c,evo(j));
        %        plr = psim(vlo(j),c,evo(j-1));
        %        pgl = psip(vgo(j-1),c,ego(j));
        %        pgr = psim(vgo(j),c,ego(j-1));
        %        vmixr = vlo(j)*evo(j)+vgo(j)*ego(j);
        %        vmixl = vlo(j-1)*evo(j-1)+vgo(j-1)*ego(j-1);
        %
        %        pl = pp(vmixl,c);
        %        pr = pm(vmixr,c);
        %        mll= evo(j-1)*dlo(j-1);
        %        mlr= evo(j)*dlo(j);
        %        mgl= ego(j-1)*dgo(j-1);
        %        mgr= ego(j)*dgo(j);
        %
        %        flc(j,1)= mll*pll+mlr*plr;
        %        flc(j,2)= 0.0;
        %        flc(j,3)= mll*pll*vlo(j-1)+mlr*plr*vlo(j);
        %
        %        fgc(j,1)=0.0;
        %        fgc(j,2)= mgl*pgl+mgr*pgr;
        %        fgc(j,3)= mgl*pgl*vgo(j-1)+mgr*pgr*vgo(j);
        %
        %        fp(j,1)= 0.0;
        %        fp(j,2)= 0.0;
        %        fp(j,3)= pl*po(j-1)+pr*po(j);

        %  First order methods ends here
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Second order method starts here:
        % Here slopelimiter is used on all variables except phase velocoties

        psll = po(j-1)+dx/2*sl2(j-1);
        pslr = po(j)-dx/2*sl2(j);
        dsll = dlo(j-1)+dx/2*sl1(j-1);
        dslr = dlo(j)-dx/2*sl1(j);
        dgll = dgo(j-1)+dx/2*sl6(j-1);
        dglr = dgo(j)-dx/2*sl6(j);

        vlv = vlo(j-1)+dx/2*sl3(j-1);
        vlh = vlo(j)-dx/2*sl3(j);
        vgv = vgo(j-1)+dx/2*sl4(j-1);
        vgh = vgo(j)-dx/2*sl4(j);

        gvv = ego(j-1)+dx/2*sl5(j-1);
        gvh = ego(j)-dx/2*sl5(j);
```

```
lvv = 1-gvv;
lvh = 1-gvh;

cl = csound(gvv,psll,dsll,k);
cr = csound(gvh,pslr,dslr,k);
c = max(cl,cr);

pll = psip(vlo(j-1),c,lvh);
plr = psim(vlo(j),c,lvv);
pgl = psip(vgo(j-1),c,gvh);
pgr = psim(vgo(j),c,gvv);
vmixr = vlo(j)*lvh+vgo(j)*gvh;
vmixl = vlo(j-1)*lvv+vgo(j-1)*gvv;

pl = pp(vmixl,c);
pr = pm(vmixr,c);

mll= lvv*dsll;
mlr= lvh*dslr;
mgl= gvv*dgll;
mgr= gvh*dglr;

flc(j,1)= mll*pll+mlr*plr;
flc(j,2)= 0.0;
flc(j,3)= mll*pll*vlo(j-1)+mlr*plr*vlo(j);

fgc(j,1)=0.0;
fgc(j,2)= mgl*pgl+mgr*pgr;
fgc(j,3)= mgl*pgl*vgo(j-1)+mgr*pgr*vgo(j);

fp(j,1)= 0.0;
fp(j,2)= 0.0;
fp(j,3)= pl*psll+pr*pslr;

%%% Second order method ends here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Here sloplimiters is used on all variables. This
% has not worked so well yet. Therefore it is commented away.

%       psll = po(j-1)+dx/2*sl2(j-1);
%       pslr = po(j)-dx/2*sl2(j);
%       dsll = dlo(j-1)+dx/2*sl1(j-1);
%       dslr = dlo(j)-dx/2*sl1(j);
%       dgll = dgo(j-1)+dx/2*sl6(j-1);
%       dglr = dgo(j)-dx/2*sl6(j);
%
%       vlv = vlo(j-1)+dx/2*sl3(j-1);
%       vlh = vlo(j)-dx/2*sl3(j);
%       vgv = vgo(j-1)+dx/2*sl4(j-1);
%       vgh = vgo(j)-dx/2*sl4(j);
%
%       gvv = ego(j-1)+dx/2*sl5(j-1);
%       gvh = ego(j)-dx/2*sl5(j);
%       lvv = 1-gvv;
%       lvh = 1-gvh;
%
%       cl = csound(gvv,psll,dsll,k);
```

```
    %       cr = csound(gvh,pslr,dslr,k);
    %       c = max(cl,cr);
    %
    %       pll = psip(vlv,c,lvh);
    %       plr = psim(vlh,c,lvv);
    %       pgl = psip(vgv,c,gvh);
    %       pgr = psim(vgh,c,gvv);
    %       vmixr = vlh*lvh+vgh*gvh;
    %       vmixl = vlv*lvv+vgv*gvv;
    %
    %       pl = pp(vmixl,c);
    %       pr = pm(vmixr,c);
    %       mll= lvv*dsll;
    %       mlr= lvh*dslr;
    %       mgl= gvv*dgll;
    %       mgr= gvh*dglr;
    %
    %       flc(j,1)= mll*pll+mlr*plr;
    %       flc(j,2)= 0.0;
    %       flc(j,3)= mll*pll*vlv+mlr*plr*vlh;
    %
    %
    %       fgc(j,1)=0.0;
    %       fgc(j,2)= mgl*pgl+mgr*pgr;
    %       fgc(j,3)= mgl*pgl*vgv+mgr*pgr*vgh;
    %
    %       fp(j,1)= 0.0;
    %       fp(j,2)= 0.0;
    %       fp(j,3)= pl*psll+pr*pslr;
end

% Fluxes have now been calculated. We will now update the conservative
% variables in each of the numerical cells.

% The source terms can be calculated by using a
% for loop.
% Note that the model is sensitive to how we treat the model
% for low Reynolds numbers (possible discontinuity in the model)
for j=1:nobox
    fricgrad(j)=dpfric(vlo(j),vgo(j),evo(j),ego(j),dlo(j),dgo(j), ...
        po(j),do(j),di(j),viscl,viscg); % Pa/m
    hydgrad(j)=g*(dlo(j)*evo(j)+dgo(j)*ego(j)); % Pa/m
end

sumfric = 0;
sumhyd= 0;

for j=1:nobox

    % Here we solve the three conservation laws for each cell and update
    % the conservative variables qv
    ar = area(j);

    % Liquid mass conservation
    qv(j,1)=qvo(j,1)-dtdx*((ar*flc(j+1,1)-ar*flc(j,1))...
        +(ar*fgc(j+1,1)-ar*fgc(j,1))...
        +(ar*fp(j+1,1)-ar*fp(j,1)));

    % Gas mass conservation:
        qv(j,2)=qvo(j,2)-dtdx*((ar*flc(j+1,2)-ar*flc(j,2))...
            +(ar*fgc(j+1,2)-ar*fgc(j,2))...
            +(ar*fp(j+1,2)-ar*fp(j,2)));
        % Mixture momentum conservation:

        qv(j,3)=qvo(j,3)-dtdx*((ar*flc(j+1,3)-ar*flc(j,3))...
            +(ar*fgc(j+1,3)-ar*fgc(j,3))...
            +(ar*fp(j+1,3)-ar*fp(j,3)))...
            -dt*ar*(fricgrad(j)+hydgrad(j));

        %  Add up the hydrostatic pressure  and friction  in the whole well.
        sumfric=sumfric+fricgrad(j)*dx;
        sumhyd=sumhyd+hydgrad(j)*dx;

    end

    % Section where we find the physical variables (pressures, densities etc)
    % from the conservative variables. Some trickes to ensure stability.
    % These are induced to avoid negative masses.
    gasmass=0;
    liqmass=0;

    for j=1:nobox

        % Remove the area from the conservative variables to find the
        % the primitive variables from the conservative ones.

        qv(j,1)= qv(j,1)/area(j);
        qv(j,2)= qv(j,2)/area(j);

        if (qv(j,1)<0.00000001)  % Trick to avoid negative masses.
            qv(j,1)=0.00000001;
        end

        if (qv(j,2)< 0.00000001)  % Trick to avoid negative masses.
            qv(j,2)=0.00000001;
        end

        % Here we summarize the mass of gas and liquid in the well
respectively.
        % These variables are important to show that the scheme is conserving
        % mass. (if e.g. gas leaks in our out of the well unintentionally in
the simulation
        % without being specified in the code,something fundamental is wrong.

        gasmass = gasmass+qv(j,2)*area(j)*dx;
        liqmass = liqmass+qv(j,1)*area(j)*dx;

        % Below, we find the primitive variables pressure and densities based
on
        % the conservative variables q1,q2. One can choose between getting
them by
        % analytical or numerical solution approach specified in the
beginning of
        % the program. Ps. For more advanced density models, this must be
changed.

        if (analytical == 1)
```

```
            %       % Analytical solution:

            % here the simple density models used in PET 510 Wellflow
modelling
            % compendium is used.

            t1=rho0-P0/al^2;

            %  Coefficients:
            a = 1/(al*al);
            b = t1-qv(j,1)-rt*qv(j,2)/(al*al);
            c = -1.0*t1*rt*qv(j,2);

            %       Note here we use the very simple models from the PET510
course
            p(j)=(-b+sqrt(b*b-4*a*c))/(2*a);  % Pressure
            dl(j)=rholiq(p(j),temp(j)); % Density of liquid
            dg(j)=rogas(p(j),temp(j)); % Density of gas

            %     The code below can be activated if we want to switch to the
            %     other set of density models. Also then remember to do the
            %     changes inside functions rogas og rholiq if we change
            % density models.

            %           x1=rho0-P0*rho0/Bheta-rho0*Alpha*(temp(j)-T0);
            %           x2=rho0/Bheta;
            %           x3=-qv(j,2)*R*temp(j);

            %           a = x2;
            %           b = x1+x2*x3-qv(j,1);
            %           c = x1*x3;

            %           p(j)=(-b+sqrt(b*b-4*a*c))/(2*a);  % Pressure
            %           dl(j)=rholiq(p(j),temp(j));
            %           dg(j)=rogas(p(j),temp(j));
        else

            %Numerical Solution: This might be used if we use more complex
            %density models. Has not been used for years.

            [p(j),error]=itsolver(po(j),qv(j,1),qv(j,2)); % Pressure
            dl(j)=rholiq(p(j),temp(j)); % Density of liquid
            dg(j)=rogas(p(j)); % Density of gas

            % Incase a numerical solution is not found, the program will
            % write out "error":
            if error > 0
                error
            end
        end

        %   Find phase volume fractions
        eg(j)= qv(j,2)/dg(j);
        ev(j)=1-eg(j);

        %    Reset average conservative varibles in cells with area included
        % in the variables.

        qv(j,1)=qv(j,1)*area(j);
        qv(j,2)=qv(j,2)*area(j);

    end  % end of loop

    %    Below we find the phase velocities by combining the
    %    conservative variable defined by the mixture momentum equation
    %    with the gas slip relation.
    %    At the same time we try to summarize the gas volume in the well.
    %    This also measure the size of the kick.

    gasvol=0;

    for j=1:nobox

        % The  interpolations introduced below are included
        % to omit a singularity in the slip relation when the gas volume
        % fraction becomes equal to 1/K. In additon, S is interpolated to
        % zero when approaching one phase gas flow. In the transition to
        % one phase gas flow, we have no slip condtions (K=1, S=0)
        % We will let the k0,s0,k1,s1 be arrays to make it easier to
        % incorporate
        % different flow regimes later.  In that case, the slip parameters
        % will
        % vary from cell to cell and we must have slip parameter values for
        % each cell.

        % MODEL TO CONTROL FROM WHERE SUSPENSION GOES TO BUBBLE FLOW %%%
        ksusp=1; %%% suspended k-value
        ssusp=0; %%% suspended s-value
        sigma=0.0772; % Interfacial tension between phases in N/m

        kbubble=1.2; % bubble k-value = 1.2 from SPE 109868
        kslug = k;% either 1.2 for upward flow or 1.12 for bullheading

        minsusplim=0.10;
        maxsusplim=0.12;

        if (eg(j)<=minsusplim) % Case of suspension
            ktemp=ksusp;
            stemp=ssusp;
        elseif((eg(j)>minsusplim)&(eg(j)<maxsusplim)) % Transition from
suspension to bubble flow
            xinttemp=(eg(j)-minsusplim)/(maxsusplim-minsusplim);
            ktemp=xinttemp*kbubble+(1-xinttemp)*ksusp;
            sbubble=1.53*(g*(dlo(j)-dgo(j))*sigma/(dlo(j)^2))^0.25;
            stemp=xinttemp*sbubble+(1-xinttemp)*ssusp;
        elseif((eg(j)>=maxsusplim)&(eg(j)<0.20)) % Case of bubble flow for
high ...
            % transition zone from bubble to slug flow (gas fractions at 20-
25%)
            % elseif((eg(j)>=maxsusplim)&(eg(j)<0.10)) % For low transition
zone from ...
            % bubble to slug flow (gas fractions at 10-15%)
            ktemp=kbubble;
            sbubble=1.53*(g*(dlo(j)-dgo(j))*sigma/(dlo(j)^2))^.25;
            stemp=sbubble;
        elseif((eg(j)>=0.20)&(eg(j)<0.25))% Transition from bubble to slug
flow ...
```

```
            % for high transition zone from bubble to slug flow (gas
fractions at 20-25%)
            % elseif((eg(j)>=0.10)&(eg(j)<0.15)) % For low transition zone
from ...
            % bubble to slug flow (gas fractions at 10-15%)
            xinttemp=(eg(j)-0.20)/0.05;
            % xinttemp=(eg(j)-0.10)/0.05; % For low transition zone from
bubble ...
            % to slug flow (gas fractions at 10-15%)
            ktemp=xinttemp*kslug+(1-xinttemp)*kbubble;
            sbubble=1.53*(g*(dlo(j)-dgo(j))*sigma/(dlo(j)^2))^0.25;
            sslug=(0.35*(g*do(j)*(dlo(j)-dgo(j))/dlo(j))^0.5);%*ftheta*fa(j);
            stemp=xinttemp*sslug+(1-xinttemp)*sbubble;
        else % Case of slug flow
            ktemp=kslug;
            sslug=(0.35*(g*do(j)*(dlo(j)-dgo(j))/dlo(j))^0.5);%*ftheta*fa(j);
            stemp=sslug;
        end

        %ktemp=k;
        %stemp=s;

        k0(j) = ktemp;
        s0(j) = stemp;

        % Interpolation to handle that (1-Kxgasvolumefraction) does not
become zero
        if ((eg(j)>=0.7) & (eg(j)<=0.8))
            xint = (eg(j)-0.7)/0.1;
            k0(j) =1.0*xint+kslug*(1-xint);  % NB kslug instead of k
        elseif(eg(j)>0.8)
            k0(j)=1.0;
        end

        % Interpolate S to zero in transition to pure gas phase
        if ((eg(j)>=0.9) & (eg(j)<=1.0))
            xint = (eg(j)-0.9)/0.1;
            s0(j) = 0.0*xint+sslug*(1-xint);  % NM sslug instead of S
        end

        % Note that the interpolations above and below can be changed
        % if numerical stability problems
        % are encountered.
        if (eg(j)>=0.999999)
            % Pure gas
            k1(j) = 1.0;
            s1(j) = 0.0;
        else
            %Two phase flow
            k1(j) = (1-k0(j)*eg(j))/(1-eg(j));
            s1(j) = -1.0*s0(j)*eg(j)/(1-eg(j));
        end

        help1 = dl(j)*ev(j)*k1(j)+dg(j)*eg(j)*k0(j);
        help2 = dl(j)*ev(j)*s1(j)+dg(j)*eg(j)*s0(j);

        vmixhelpl = (qv(j,3)/area(j)-help2)/help1;
        vg(j)=k0(j)*vmixhelpl+s0(j);
        vl(j)=k1(j)*vmixhelpl+s1(j);
        % Variable for summarizing the gas volume content in the well.
        gasvol=gasvol+eg(j)*area(j)*dx;
    end

    % Old values are now set equal to new values in order to prepare
    % computation of next time level.
    po=p;
    dlo=dl;
    dgo=dg;
    vlo=vl;
    vgo=vg;
    ego=eg;
    evo=ev;
    qvo=qv;

    % Section where we save some timedependent variables in arrays.
    % e.g. the bottomhole pressure. They will be saved for certain
    % timeintervalls defined in the start of the program in order to ensure
    % that the arrays do not get to long!

    if (counter>=nostepsbeforesavingtimedata)
        printcounter=printcounter+1;
        time  % Write time to screen.
        % Outlet massrates (kg/s) vs time

liquidmassrateout(printcounter)=dl(nobox)*ev(nobox)*vl(nobox)*area(nobox);

gasmassrateout(printcounter)=dg(nobox)*eg(nobox)*vg(nobox)*area(nobox);

        % Outlet flowrates (lpm) vs time
        liquidflowrateout(printcounter)=liquidmassrateout(printcounter)/...
            rholiq(P0,T0)*1000*60;
        gasflowrateout(printcounter)=gasmassrateout(printcounter)/...
            rogas(P0,T0)*1000*60;

        % Hydrostatic and friction pressure (bar) in well vs time
        hyd(printcounter)=sumhyd/100000;
        fric(printcounter)=sumfric/100000;

        % Volume of gas in well vs time (m3). Also used for indicating kick
        % size in well.

        volgas(printcounter)=gasvol;

        % Total phase masses (kg) in the well vs time
        % Used for checking mass conservation.
        massgas(printcounter)=gasmass;
        massliq(printcounter)=liqmass;

        % pout calculates the pressure at the outletboundary. I.e. upper edge
        % of uppermost cell. Corresponds where the well ends at surface. The
        % reason we do this is the fact than in AUSMV is all variables
defined
        % in the mid point of the numerical cells.
        pout(printcounter)=(p(nobox)-0.5*dx*...
            (dlo(nobox)*evo(nobox)+dgo(nobox)*ego(nobox))*g-
dx*0.5*fricgrad(nobox))/100000;
```

```
        % pin (bar) defines the  pressure at the inlet boundary, I.e lower
        % edge of the lowermost cell. Corresponds to TD of well.
        pin(printcounter)=
(p(1)+0.5*dx*(dlo(1)*evo(1)+dgo(1)*ego(1))*g+0.5*dx*fricgrad(1))/100000;

        % Pressure in the middle of top box (bar).
        pnobox(printcounter)=p(nobox)/100000;  %

        % Vector saving pumped bullheading volumes

        bulvol(printcounter)=bullvolume; % Unit m3

        % Time variable
        timeplot(printcounter)=time;

        counter = 0;

    end
end

% end of stepping forward in time.
% Printing of resultssection
countsteps % Marks number of simulation steps.

% Plot commands for variables vs time. The commands can also
% be copied to command screen where program is run for plotting other
% variables.
toc,
e = cputime-t

% Plot bottomhole pressure
% plot(timeplot,pin)

% Show cfl number used.
disp('cfl')
cfl = al*dt/dx

% plot(timeplot,pin)
%plot(timeplot,hyd)
%plot(timeplot,fric)
%plot(timeplot,liquidmassrateout)
%plot(timeplot,gasmassrateout)
plot(timeplot,volgas)
%plot(timeplot,liquidflowrateout)
%plot(timeplot,gasflowrateout)
% plot(timeplot,massgas)
%plot(timeplot,massliq)
%plot(timeplot,pout)
%plot(timeplot,pnobox)

%Plot commands for variables vs depth/Only the last simulated
%values at endtime is visualised

%plot(vl,x);
%plot(vg,x);
% plot(eg,x);
%plot(p,x);
%plot(dl,x);
%plot(dg,x);
```