AVAR ARI LALO
SUPERVISOR: RUNE WIGGO TIME

A numerical Python model of gas liquid flow in petroleum wells

From reservoir to wellhead assuming homogeneous flow with a generalized slip model

Bachelor thesis (2024)

Energy- and Petroleum Technology

Faculty of Science and Technology

Department of Energy and Petroleum Engineering

Abstract

The primary objective of this thesis is to develop a Python script for predicting reservoir production in petroleum wells, based on an existing MATLAB script. This transition is motivated by Python's cost-effectiveness and open-source nature, offering a free alternative to the commercially licensed MATLAB. There are three codes: one main code that has been translated from the MATLAB script, and two supplementary scripts that are built on the main code. The study investigates the interplay between key parameters, such as wellhead pressure, gas fraction, mixture velocity, and boiling pressures, utilizing three distinct Python scripts. The main script iteratively evaluates varying wellhead pressures, while two supplementary scripts explore the effects of boiling pressure on reservoir production and the relationship between gas velocity and pressure gradient. This thesis comprises three key aspects:

- The first aspect of the thesis focuses on validating the Python model against theoretical expectations for single-phase flow conditions. Results demonstrate the model's accuracy, highlighting its capacity to simulate realistic fluid dynamics within the well.
- The second aspect examines the influence of wellhead pressures on gas fraction, mixture velocity, and boiling pressures. Findings indicate that higher wellhead pressures correlate with lower gas fractions and reduced mixture velocities, aligning with fluid behavior in multiphase flow.
- In the third part, the study explores the interaction between boiling pressures, wellhead pressures, and reservoir production. The results reveal complex trends, where increasing boiling pressure initially enhances production due to reduced hydrostatic pressure, but at higher pressures, frictional pressure becomes dominant, diminishing production.

Overall, this thesis provides a robust framework for understanding gas-liquid flow dynamics in petroleum wells, demonstrating the effectiveness of Python for digital modeling in the energy sector. The insights gained contribute to optimizing well operations and enhancing production efficiency. Due to the availability of Python, further development of the models is encouraged.

Preface

This thesis is submitted in partial fulfillment of the requirements for the Bachelor's degree in Energy and Petroleum Technology. I extend my sincere gratitude to Professor Rune Wiggo Time for his invaluable guidance and patience throughout my research. His expertise and advice have been crucial to the completion of this project. I also wish to thank my parents for their unwavering support and belief in my abilities during this time. Their encouragement has been a constant source of motivation.

Stavanger, May 15, 2024

[illegible]

Avar lalo

List of Figures

List of Tables

Symbols

A	Area
C	Empirical constant by Dukler
D	Diameter of the pipe
dP/dL	Frictional pressure gradient
dP_{hyd}/dh	Hydrostatic pressure gradient
dP_{tot}/dh	Total pressure gradient
Δh	Change in height
ΔP	Change in pressure
f	Friction factor
g	Gravitational constant
K	Perforation inflow constant
m	Mass
$\dot{m}$	Mass flow
n	Empirical exponent by Dukler
q_G	Volumetric flow rate of gas phase
q_L	Volumetric flow rate of liquid phase
Re	Reynolds number
S	Slip ratio
U_{GS}	Superficial gas velocity
U_{LS}	Superficial liquid velocity
V	Volume

Symbols

v	Flow velocity
v_G	Flow velocity of gas
v_L	Flow velocity of liquid
ϵ_G	Gas fraction
ϵ_{Gs}	Gas fraction after slip calculation
ϵ_L	Liquid fraction
μ	Dynamic viscosity
μ_G	Dynamic viscosity of the gas phase
μ_L	Dynamic viscosity of the liquid phase
μ_m	Mixture viscosity
ρ	Density
ρ_o	Oil density
ρ_G	Gas density
ρ_{G0}	Gas density at reference point
ρ_L	Liquid density
ρ_m	Mixture density
ν	Kinematic viscosity
ν_g	Kinematic gas viscosity

Abbreviations

$A - AW$	Annular (wavy) flow
$CO2$	Carbon dioxide
DB	Dispersed bubble flow
$Dhor$	Diameter of horizontal section
$Dver$	Diameter of vertical section
Dlh	Segment length for horizontal section
Dlv	Segment length for vertical section
$Dpdxf$	Frictional pressure gradient
$DpdxHyd$	Hydrostatic pressure gradient
$DpdxTotal$	Total pressure gradient
EB	Elongated bubble flow
$Errlimit$	Error limit
$Epsgi$	Gas fraction after slip calculation
I	Slug flow
Kxi	Perforation inflow contributions
Lh	Length of horizontal well section
Lv	Height of vertical section

Abbreviations

$Mtot$	Mass flow
Myl	Liquid viscosity
NaN	Not A Number
Nyg	Gas viscosity
Pb	Boiling pressure
$Perforr$	Perforation locations
Pwh	Wellhead pressure
Pwh_values	Array of wellhead pressures
$PWSTART$	Initial constant pressure
$PWHI$	Sequence of wellhead pressures
$P\_wh, 0$	Hydrostatic pressure at the wellhead
$P\_wh, calculated$	Calculated pressure at the wellhead
Pr	Reservoir pressure
Pwi	Well pressure
$Relerr$	Relative error
SI	International System of Units
SS	Stratified smooth flow
SW	Stratified wavy flow
$Ugsi$	Velocity of the gas phase
$Ulsi$	Velocity of the liquid phase
$Umix$	Mixture velocity

Contents

1 Introduction

As the petroleum engineering landscape undergoes significant digital transformation, the crucial role of advanced computational models is increasingly recognized [1]. These models, essential for simulating complex gas-liquid flows in petroleum wells, stand at the forefront of enhancing operational efficiency and ensuring improved productivity within the oil and gas sector [2]. By optimizing production processes and minimizing the need for excessive resource extraction, efficient production models can reduce CO2 emissions and create revenue growth opportunities [3]. This not only aligns with the broader industry goals of sustainable energy production and environmental stewardship but also underscores the necessity of comprehending parameters critical to petroleum production. Moreover, it emphasizes the demand for precise, accurate models—a focal point of this thesis.

Transitioning from conventional Matlab-based tools to Python, this work champions the broader shift towards more accessible, collaborative, and open-source methods in scientific research and education. Python's widespread adoption across various scientific disciplines, attributed to its simplicity, versatility, and robust community support, positions it as an ideal platform for developing and disseminating computational models in petroleum engineering [4] [5].

The primary motivation behind this model is to provide a comprehensive tool for analyzing the intricate interplay among various physical quantities that influence the behavior of gas and liquid within petroleum wells. The model includes both a vertical section and a horizontal section, allowing for detailed exploration of how changes in parameters such as reservoir pressures, boiling pressure, and wellhead pressure impact overall well performance. This approach not only enables more accurate predictions of oil production and flow behaviors but also serves as an invaluable educational resource, allowing users to experiment with different scenarios and gain insights into the dynamics of well operations.

Key to this model is its ability to simulate critical aspects of well dynamics, such as the impact of wellhead pressures on reservoir production, variations in gas fraction along the wellbore, and the velocities of mixed gas-liquid flows. These capabilities ensure that the model can be employed not merely as a tool for operational planning and optimization but also as a means of advancing our theoretical understanding of well mechanics.

By integrating this Python model into the petroleum engineering toolkit, this thesis not only contributes to the ongoing digitalization of the industry but also exemplifies how such tools can foster learning and innovation. It represents a step forward in preparing the next generation of petroleum engineers to navigate the challenges of an ever-evolving energy landscape, armed with the knowledge and skills derived from cutting-edge digital technologies. Furthermore, by promoting more efficient and environmentally friendly production practices, the model underscores the critical role of engineering innovation in achieving sustainability goals within the energy sector.

2 Theory

2.1 Fluid Dynamics

Fluid dynamics is a subject well-known to most engineers, given its extensive application across various engineering disciplines. In this thesis, the principles of fluid dynamics are employed to construct a numerical model. Therefore, a solid understanding of fluid dynamics fundamentals is crucial for comprehending the intricacies of the thesis work. In this thesis, adherence to the International System of Units (SI) is maintained [6].

2.2 Density

Density is defined as mass per unit volume of a given substance and is a fundamental concept in fluid dynamics, affecting the behavior of fluids in motion. Density is denoted by the greek letter ρ and expressed in kilograms per cubic meter *kg/m³* and calculated with this formula where m is mass, and V is volume [7].

$$\rho = \frac{m}{V} \tag{1}$$

In the context of this thesis, density is paramount in the modeling of gas-liquid flow within petroleum wells, due to its extensive application in the computational aspects of the study

2.3 Perforation Inflow

Perforations in oil and gas wells allow fluids to flow from the reservoir into the wellbore. The rate at which fluid enters the well through these perforations is known as the perforation inflow rate. The perforation inflow rate is influenced by several factors, including the pressure differential between the reservoir and the wellbore, the fluid density, and the perforation inflow constant K. The perforation inflow constant characterizes the effectiveness of the perforations in allowing fluid to enter the well and is typically expressed in units of $m^3/s \cdot Pa$ [8]. In this thesis, the inflow through the perforation will be considered constant, assuming it is a known value.

2.4 Mass flow

Mass flow, denoted as $\dot{m}$ refers to the movement of mass through a system over a period of time, typically quantified in units of kilograms per second *kg/s*. In the context of this thesis, mass flow will be synonymous with reservoir production. Understanding mass flow, or reservoir production, is crucial as it represents one of the most critical parameters to be studied. Analyzing factors that influence production is essential for gaining insights into industry operations and optimizing production processes. In this report, the perforation inflow constant will be utilized to model mass inflow through the perforations using the following formula [9]:

$$\dot{m} = \rho_o \cdot \mathrm{K} \cdot (\mathrm{Pr} - \mathrm{Pwi}) \tag{2}$$

where:

- $\dot{m}$ is the mass inflow in kg/s.
- ρ_o is the oil density in kg/m^3.
- K is the perforation inflow constant in m^3/s $\cdot$ Pa.
- Pr is the reservoir pressure in Pa.
- Pwi is the well pressure in Pa.

Equation 2 is used in the Python model to simulate the mass inflow through the perforations of a well, accounting for the pressure difference between the reservoir and the wellbore. The perforation inflow constant K determines the inflow rate for a given pressure differential and fluid density.

2.5 Viscosity

Viscosity describes the resistance of fluid (liquid or gas) to flow in a medium, such as a pipeline. It is the opposition to flow and may be thought of as internal friction between the molecules. A fluid with low viscosity flows easily due to minimal internal friction among its molecules during movement. Variations in viscosity plays a pivotal role in determining the transition from laminar to turbulent flow. This transition is crucially dependent on the Reynolds number, further connecting with the detailed discussion on turbulence and laminar flow in the flow regimes section. Viscosity is divided into two parts, dynamic and kinematic viscosity, where both will be used in the python model. [9] [10]

2.5.1 Dynamic viscosity

Dynamic viscosity refers the fluid's resistance to flow when an external force is applied. Dynamic viscosity is denoted with the greek letter μ [11].

2.5.2 Kinematic viscosity

Kinematic viscosity refers to the fluid's resistive flow of fluid under the weight of gravity. In compare to kinematic, there are no external force acting on the fluid beside gravity [11]. Kinematic viscosity is denoted with greek letter ν. The mathematical formula is shown below, and as mentioned μ is dynamic viscosity and ρ is density. [9].

$$\nu = \frac{\mu}{\rho} \tag{3}$$

2.6 Flow regimes

Flow regime refers to the pattern and characteristics of fluid movement within a pipeline or other channel, which can be affected by several factors such as fluid properties, pressure, channel dimensions, and gravity. In fluid mechanics, flow regimes are commonly divided into single-phase and multi-phase categories. [9].

2.6.1 Single-phase flow

In single-phase flow, the fluid behaves homogeneously, meaning it consists of one phase, either liquid or gas, flowing through a pipe or conduit. The flow regime in such a situation can be categorized based on the Reynolds number, which is a dimensionless quantity indicating whether the flow will be laminar or turbulent. [9].

Historically, Osborne Reynolds' experiments in the late 19th century were fundamental in characterizing the transition between laminar and turbulent flow. Figure 1 illustrates his original setup and visualization techniques that clearly distinguish between laminar and turbulent flow based on the behavior of dye in the fluid stream, providing a visual foundation for the concept of Reynolds number [12].

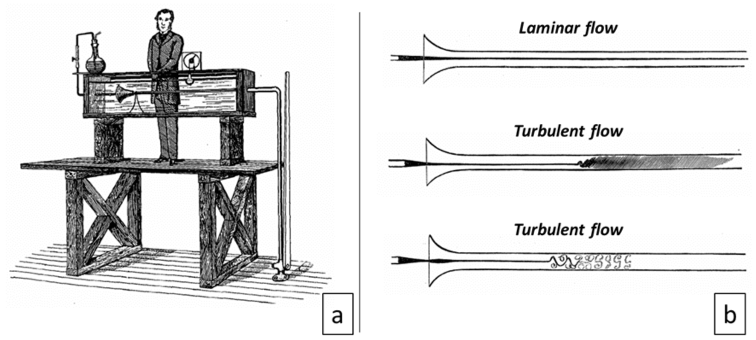

Figure 1: Osborne Reynolds' experimental setup and flow regimes visualization: (a) The apparatus used to demonstrate fluid flow. (b) Distinction between laminar and turbulent flow in a pipe.

[12]

- **Laminar flow**: Laminar flow describes a flow where the fluid moves in smooth, parallel layers or streams with minimal mixing between them. The movement of the fluid particles is orderly with all particles moving in straight lines parallel to the pipe walls. Laminar flow is typically characterized by a Reynolds number (Re) less than or equal to 2000. This means the inertial forces leading to turbulence are weak compared to the viscous forces that resist it [9].

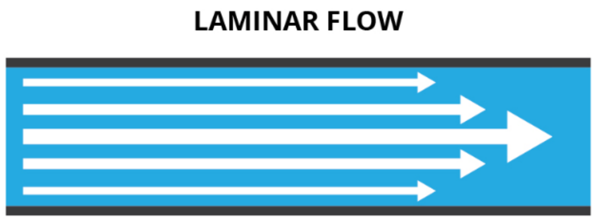

Figure 2: Illustration of laminar flow in a pipe.

[13]

- **Transitional flow**: Transitional flow is characterized as a flow regime that occurs when the flow is not clearly laminar or turbulent. It is typically observed within a Reynolds number range of $2000 < Re \leq 4000$. In this regime, the flow exhibits characteristics of both laminar and turbulent flow, with occasional bursts of turbulence that decay back into a laminar state. [9]
- **Turbulent flow**: In turbulent flow, the fluid experiences irregular fluctuations and mixing, with particles moving in a chaotic manner. Unlike laminar flow, the layers of fluid do not remain distinct, and there are eddies, swirls, and unpredictability in the flow pattern. This type of flow is generally associated with Reynolds numbers greater than 4000, indicating that inertial forces are strong enough to overcome the viscous forces and disrupt the orderly flow pattern [9]

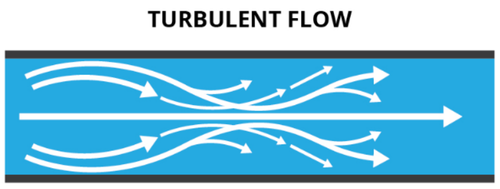

Figure 3: Illustration of turbulent flow in a pipe.
[13]

2.6.2 Multi-phase flow

The study of multi-phase flow in fluid dynamics is essential for understanding the behavior and distribution of phases within a well's flow. It differentiates between the flow of single-phase fluid, which can be either gas or liquid, and two-phase fluid, a mixture of both. These distinct behaviors are influenced by variables such as flow velocity, phase distribution, and the well's orientation, whether vertical or horizontal. Under are flow regimes listed from lowest to highest flow velocity for horizontal and vertical section.

Flow regimes at horizontal section

- **Stratified smooth flow & Stratified wavy flow (Low Velocity)**: In Stratified smooth flow, the gas and liquid phases are distinctly separate, with the lighter phase (gas) above the heavier phase (liquid), characterized by a smooth interface at low velocities. As velocity increases, the interface becomes wavy, transitioning into Stratified wavy flow, identifiable by the small wave formations on the liquid's surface. These regimes typically occur at the lowest flow velocities within horizontal flows [9].
- **Elongated bubble flow & Slug flow (Intermediate Velocity)**: Elongated bubble flow is marked by the presence of elongated gas regions within the liquid phase, resembling large bubbles. This regime is a precursor to Slug flow, which features pronounced, alternating large slugs of gas and liquid. Both regimes represent a blend of phase separation and mixing, often categorized as transitional flow patterns, occurring at medium velocities [9].

- **Dispersed bubble flow & Annular flow (High Velocity)**: When the flow velocity increases further, small gas bubbles become uniformly dispersed throughout the liquid phase, defining the Dispersed bubble flow regime. At even higher velocities, Annular flow emerges, characterized by a gas core surrounded by a thin film of liquid along the pipe walls. In the horizontal orientation, this film may exhibit wave-like behavior, hence the designation "wavy." This regime is encountered at the highest flow velocities in horizontal wells [9].

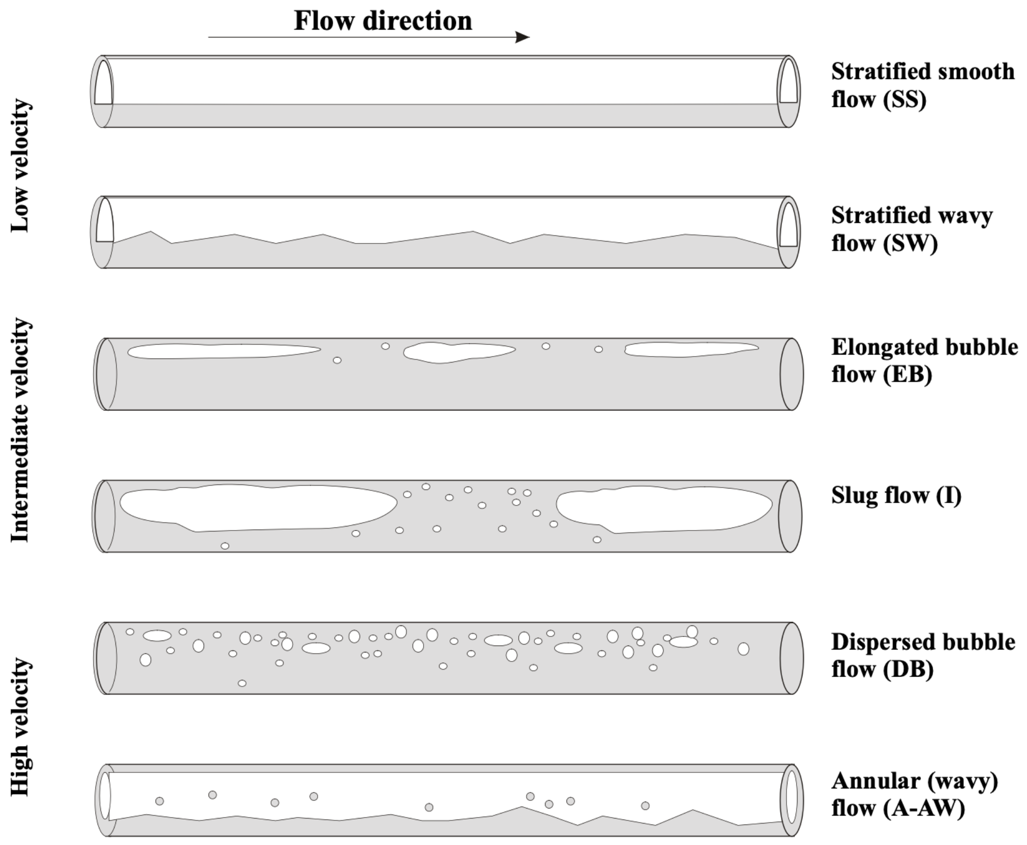
Flow direction
Low velocity
Intermediate velocity
High velocity
Stratified smooth flow (SS)
Stratified wavy flow (SW)
Elongated bubble flow (EB)
Slug flow (I)
Dispersed bubble flow (DB)
Annular (wavy) flow (A-AW)

Figure 4: Illustration of horizontal flow regimes
[9]

In our analysis, the flow velocities can be categorized into three distinct regimes: low, intermediate, and high velocity. These classifications are based on the correlations established by Mandhane, represented with dashed lines, and Taitel and Dukler for air-water flow in a *2.5 cm i.d.* pipe under atmospheric conditions, indicated by solid lines. The variables U_{GS} and U_{LS} denote the superficial gas and superficial liquid velocities respectively, measured in meters per second (m/s). Superficial velocity refers to the velocity of a phase (gas or liquid) if it alone were occupying the entire cross-sectional area of the pipe, ignoring the presence of the other phase. The flow regime map in Figure 5 illustrates the relationship between gas and liquid velocities to determine the specific flow regimes encountered in horizontal two-phase flow conditions. This map, based on the theoretical frameworks established by

Mandhane [14], and Taitel and Dukler [15], has been adapted from Time [9] to reflect current understanding and graphical presentation styles.

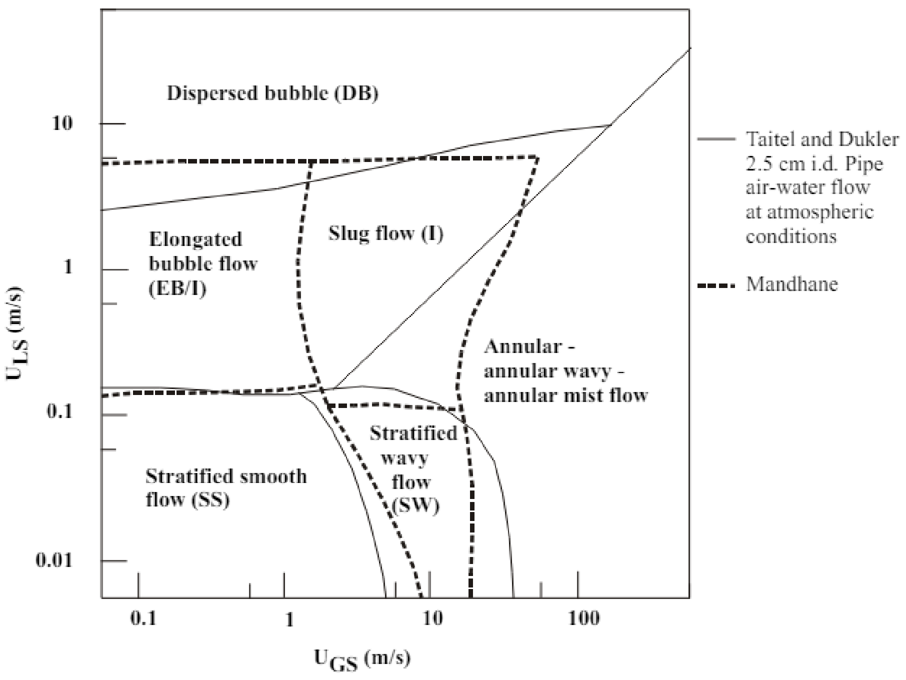

Figure 5: Flow regimes and flow regime map in horizontal two-phase flow, adapted from Time

[9]

Flow regimes at vertical section

- **Slug flow (Low Velocity)**: Similar to its occurrence in horizontal sections, slug flow in vertical pipes is characterized by alternating large slugs of liquid and gas, typically containing bubbles. This regime presents a notable mix of the two phases illustrated in figure 6 [9].
- **Churn Flow & Dispersed Bubble Flow (Intermediate Velocity)**: These vertical flow regimes are characterized by intermediate velocities where distinct behaviors of gas and liquid mixing are observed. Churn flow is notably turbulent and chaotic, with both gas and liquid phases intermingling without any specific pattern — a stark contrast to the structured slugs of the slug flow regime. This intense mixing often transitions into the Dispersed Bubble Flow regime as the velocity increases, leading to small gas bubbles being spread throughout the liquid phase. Together, these regimes highlight the complex dynamics of multi-phase flow at varying velocities and the transitional nature of the flow patterns between the lower velocity stratified flows and the higher velocity annular flows.

This is illustrated in Figure 6, which shows an example of churn flow in a vertical section [9].

- **Annular flow (High Velocity)**: Resembling annular flow in horizontal sections, this regime in vertical pipes shows a central gas core surrounded by a liquid film. However, due to gravity, the liquid film is less evenly distributed than in horizontal flow, often leading to a thicker layer at the bottom. This regime is observed at the highest velocities and is significantly influenced by gravitational forces. See Figure 6 for illustration of annular flow in horizontal section [9].

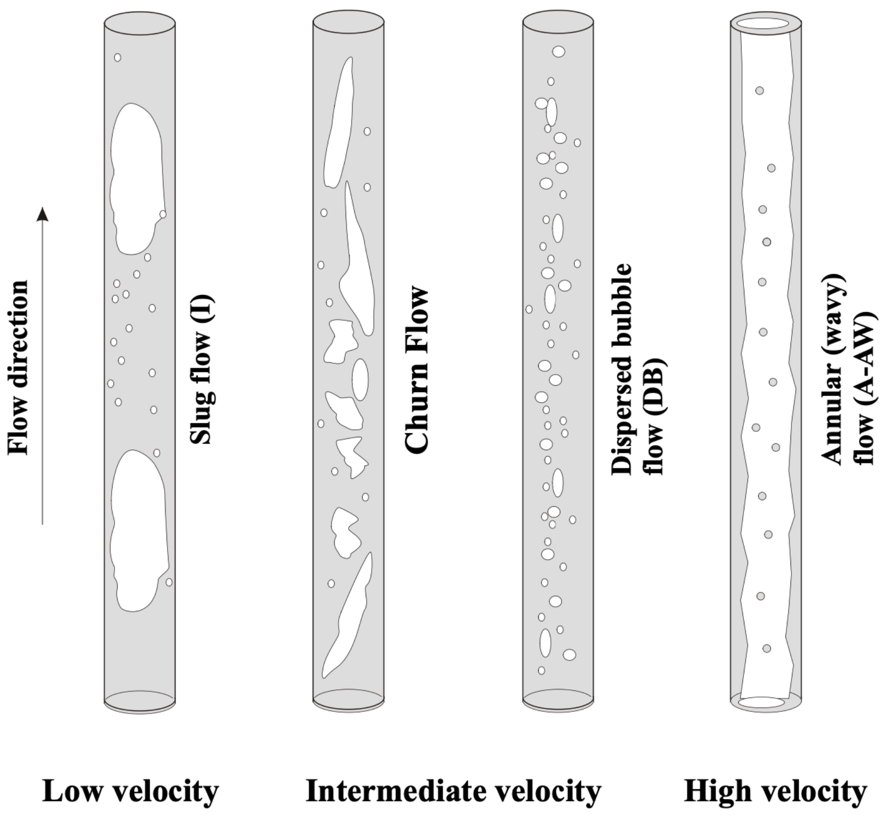

Figure 6: Illustration of vertical flow regimes
[9]

Similar to horizontal flow regimes, vertical flows also categorize into low, intermediate, and high velocities. These classifications are based on the correlations established by Barnea, Taitel, and Dukler [16], as depicted in Figure 7. The variables U_{GS} and U_{LS} denote the superficial gas and liquid velocities, respectively, measured in meters per second (m/s). The flow regime map in Figure 7 illustrates the relationship between gas and liquid velocities to determine the specific flow regimes encountered in vertical two-phase flow conditions.

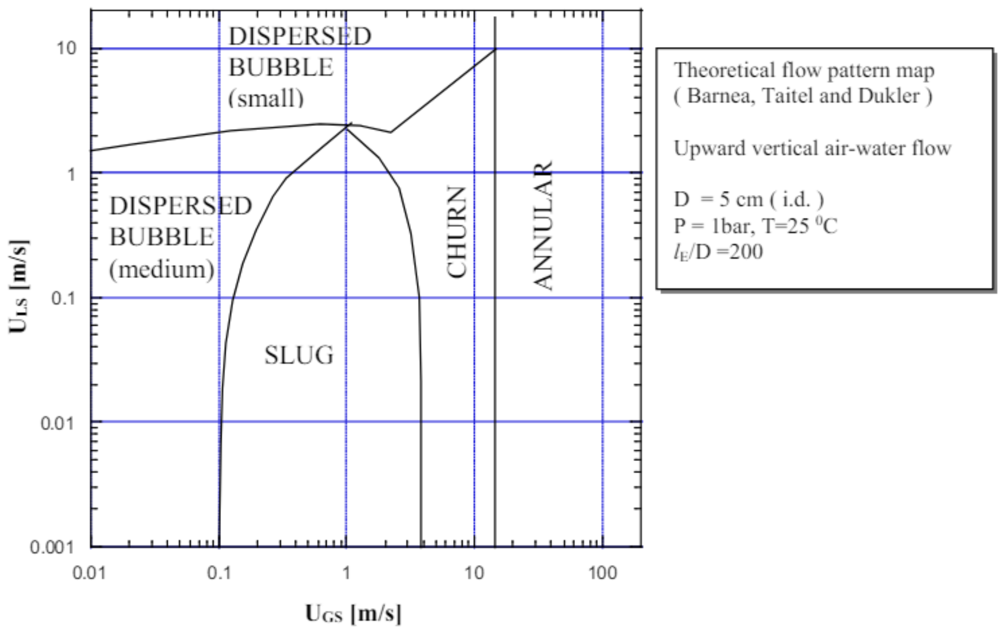

Figure 7: Flow regimes and flow regime map in horizontal two-phase flow, adapted from Time

[9]

2.7 Fluid fractions

Fluid fractions refer to the proportions or fractions of different components within a fluid mixture. In multi-phase flow scenarios, fluid fractions indicate the relative amounts of each component present, such as the amount of liquid and gas. In this thesis, the following fluid fractions will be utilized:

- **Gas Fraction**: The gas fraction, denoted as ϵ_G, represents the proportion of gas present in the fluid mixture at various positions along the wellbore. This gas fraction is crucial for understanding the behavior of multi-phase flow, especially in scenarios where gas and liquid phases coexist. The gas fraction is influenced by factors such as reservoir pressure, boiling point pressure, wellhead pressure, and fluid properties (density and viscosity), as well as flow conditions. The gas fraction can be computed by the equation under[9]:

$$\epsilon_G = \frac{q_G}{q_L + q_G} \tag{4}$$

  where:

  - q_G is the volumetric flow rate for the gas phase,
  - q_L is the volumetric flow rate for the liquid phase.

  Assuming that the entire vertical section has a constant cross-sectional area, dividing by the area converts flow rates into velocities.

$$\epsilon_G = \frac{v_G}{v_G + v_L} \tag{5}$$

where:

- v_G is the flow velocity of the gas phase
- v_L is the flow velocity of the liquid phase

Similarly, in a two-phase flow, the liquid fraction ϵ_L can be computed by:

$$\epsilon_L = 1 - \epsilon_G \tag{6}$$

- **Mixture Density & Viscosity**: In multi-phase flow involving two different phases, the properties of the mixture density ρ_m and mixture viscosity μ_m are calculated using the following formulas, which incorporate the previously defined gas fraction [9]:

$$\rho_m = \epsilon_G \rho_G + \epsilon_L \rho_L \tag{7}$$

$$\mu_m = \epsilon_G \mu_G + \epsilon_L \mu_L \tag{8}$$

where:

- ϵ_G is the gas fraction,
- ϵ_L is the liquid fraction,
- ρ_G is the density of the gas phase,
- ρ_L is the density of the liquid phase,
- μ_G is the dynamic viscosity of the gas phase,
- μ_L is the dynamic viscosity of the liquid phase.

2.8 Flow velocity

Flow velocity, denoted as v in fluid dynamics refers to the speed at which a fluid particle moves through a medium, typically measured in a specific direction. In the context of a fluid flowing through a pipe or over a surface, the flow velocity indicates how fast the fluid is traveling in the direction of the flow and it units are m/s.

2.9 Slip in two-phase flow

The slip ratio, denoted by S, is a dimensionless quantity that measures the relative velocities of the gas and liquid phases in two-phase flow. It is commonly defined as the ratio of the velocity of the gas phase to the velocity of the liquid phase [9]:

$$S = \frac{v_G}{v_L} \tag{9}$$

where:

- v_G is the velocity of the gas phase
- v_L is the velocity of the liquid phase.

This ratio is important for predicting flow patterns and managing the operational aspects of petroleum wells, especially in vertical segments where gravity markedly influences phase separation.

In the context of this thesis, an alternative approximation for the slip ratio S was suggested by the supervisor based on the gas fraction ϵ_G [17]:

$$S = 1 + 10 \cdot \epsilon_G \tag{10}$$

where ϵ_G denotes the gas fraction before slip calculation. This empirical approximation is often used in specific flow conditions.

The gas fraction after slip calculation, denoted as ϵ_{Gs}, is then given by [9]:

$$\epsilon_{Gs} = \frac{v_G}{v_G + S \cdot v_L} \tag{11}$$

The choice of slip ratio S varies across different flow regimes. Based on private communication with the supervisor, the following slip ratios were suggested for various flow regimes [18]:

- Dispersed bubble flow: $S \leq 1.1$,
- Slug flow: $S = 1.25 - 1.3$,
- Churn flow: $S = 3 - 4$, and
- Annular mist flow: $S \geq 10$.

These slip ratios provide insights into how the slip factor S varies across different flow regimes, allowing for more accurate modeling of multiphase flow dynamics.

In our numerical model for simulating gas-liquid flow in petroleum wells, understanding slip helps refine predictions of how the phases behave as they move through the wellbore, which is crucial for optimizing well performance [9].

2.10 Reynolds number

The Reynolds number denotes as Re is a dimensionless number used to describe the flow pattern of a fluid. It helps predict whether the flow is laminar, turbulent, or in a transitional state between the two. The Reynolds number quantifies the ratio of inertial forces to viscous forces within a fluid flow and it is expressed mathematically as [12] [9]:

$$Re = \frac{\rho v D}{\mu} = \frac{\rho v^2}{\mu(\frac{v}{D})} \tag{12}$$

where:

- ρ is the density of the fluid, contributing to the kinetic energy term.
- v is the flow velocity, affecting both the kinetic energy and the viscous force.
- D is the diameter of the pipe, influencing the scale of viscous resistance.
- μ is the dynamic viscosity

This equation shows that the Reynolds number can be viewed in terms of a ratio of kinetic energy to viscous forces. The numerator, ρv^2, indicates the fluid's kinetic energy per unit volume, while the denominator, $\mu \frac{v}{D}$, represents the viscous forces per unit area, scaled by the velocity gradient across the characteristic length D. Understanding this relationship helps in analyzing the conditions under which the flow transitions from laminar to turbulent, with higher Reynolds numbers indicating a greater dominance of inertial forces over viscous forces [9].

The Reynolds's number aids in distinguishing between different flow regimes for single-phase flow [9]:

- Laminar flow ($Re \leq 2000$)
- Transitional flow ($2000 < Re \leq 4000$)
- Turbulent flow ($Re > 4000$)

Based on Equation 12, the transition from laminar to turbulent flow is influenced by changes in the Reynolds number (Re). Specifically, increasing the flow velocity (v), the diameter of the pipe (D), or the fluid density (ρ), while keeping other factors constant, will lead to an increase in Re. Conversely, decreasing the viscosity (μ) of the fluid also increases Re. These changes enhance the dominance of inertial forces, represented by the kinetic energy (ρv^2), over viscous forces, denoted by ($\mu v/D$). This balance is crucial as high Reynolds number reflects a condition where inertial forces overpower the viscous forces, making the flow more susceptible to turbulence [9] [12].

This formulation of the Reynolds number indicates that as the kinetic energy of the fluid surpasses the damping effect of the viscosity, the flow becomes unstable and transitions into chaotic turbulence. High Reynolds numbers reflect conditions where inertial forces dominate over viscous forces, increasing the likelihood of turbulent flow [9] [12].

2.10.1 Effects of Turbulence

Turbulence has several critical effects on the flow within pipes [9]:

- It flattens the velocity profile across the pipe, reducing the velocity gradient from the center towards the walls seen in Figure 8.
- It breaks up bubbles and droplets into smaller units, which is particularly important in multiphase flows.
- It reduces the relative velocity between different phases (e.g., liquid and gas), thereby lowering the slip ratio towards 1 in highly mixed flows. This aspect is crucial for accurate simulation of such systems.

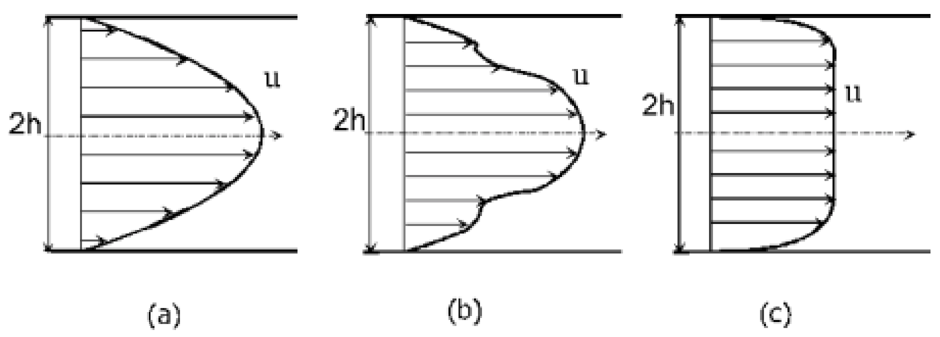

Figure 8: Velocity profile during transition from laminar flow to turbulence for plane Poiseuille flow: (a) Laminar flow, (b) Transitional flow, (c) Turbulence.
[19]

2.11 Friction Factor

The friction factor, denoted as f, is a crucial dimensionless number in fluid dynamics that quantifies the resistance a fluid encounters within a pipe system. The value of the friction factor is determined by the Reynolds number, which helps discern if the flow is laminar or turbulent. In turbulent flow, resistance increases due to a more uniform velocity profile toward the pipe wall (although fluctuating), which can be interpreted as an increase in effective viscosity, hence necessitating a modified friction factor.

For the turbulent regime, empirical formulas are used to account for this increased resistance. In this thesis, the empirical relation known as the Blasius form is adopted [9]:

$$f = C \cdot Re^{-n} \tag{13}$$

where:

- C is a empirical constant by Dukler for turbulent regime
- Re is the Reynolds number, indicating the flow regime (laminar or turbulent),
- n is the exponent derived empirically by Dukler to fit the turbulent flow data.

For this thesis, Dukler's values are used for C and n, specifically $C = 0.046$ and $n = 0.2$, leading to the following equation [9]:

$$f = 0.046 \cdot Re^{-0.2} \tag{14}$$

This equation simplifies the computation of the friction factor for high Reynolds numbers in turbulent flow conditions, omitting the need for complex calculations such as those required by the implicit Colebrook equation, which necessitates numerical solutions [20].

For laminar flow, the relationship is more straightforward, with the friction factor being inversely proportional to the Reynolds number [9]:

$$f = \frac{64}{Re} \tag{15}$$

2.12 Pressure gradients

Pressure gradient describe how the pressure develops. In this thesis, an examination is conducted on the changes of pressure through petroleum wells. This chapter will examine two different pressure gradients.

2.12.1 Frictional pressure gradient

The frictional pressure gradient in a fluid system describes the rate of pressure loss per unit length as the fluid flows through a pipe. This loss is due to the friction between the fluid and the pipe wall and within the fluid itself. The gradient is determined by the friction factor, flow velocity, and fluid density. For laminar and turbulent flows, it is given by the Fanning equation [9]:

$$\frac{dP_{\text{fric}}}{dL} = -\frac{4}{D} \cdot f \cdot \frac{1}{2} \cdot \rho v^2 \tag{16}$$

where:

- $\frac{dP_{\text{fric}}}{dL}$ is the frictional pressure gradient, indicating the rate of pressure decrease per unit length of the pipe.
- f is the friction factor, computed as described in Equations 14 or 15 depending on whether the flow is laminar or turbulent.
- D is the diameter of the pipe
- ρ is the density of the fluid
- v is the flow velocity, the square of which directly affects the rate of pressure loss; higher velocities result in greater frictional losses.

The negative sign indicates that the pressure decreases in the direction of flow due to friction.

2.12.2 Hydrostatic pressure gradient

The hydrostatic pressure gradient in fluid dynamics refers to the rate of change in pressure in a fluid column due to the weight of the fluid above. This gradient is directly proportional to the density of the fluid and the acceleration due to gravity, and it varies with the depth of the fluid column. The formula is shown below [9]:

$$\frac{dP_{\text{hyd}}}{dh} = \rho g \tag{17}$$

where

- dP_{hyd} is the change in pressure in a fluid column due to hydrostatics
- dh is the change in depth of the fluid column
- ρ is the density of the fluid
- g is the gravitational constant

2.13 Boiling pressure

The term boiling pressure in this thesis refers to the pressure at which a substance undergoes a phase transition from liquid to vapor at a specified temperature, maintaining a constant temperature. In the context of this thesis, boiling pressure serves as a critical parameter for identifying the point at which single-phase flow transitions to multi-phase flow. The boiling pressure will be treated as a constant, selected specifically for each simulation run.

3 Methodology

3.1 MATLAB

MATLAB is a programming platform designed specifically for engineers and scientists. It is centered around the MATLAB language, which is a matrix-based language optimized for expressing computational mathematics. The platform is used globally by millions of professionals across various industries and academic fields, including deep learning, signal processing, and computational biology, to analyze data, develop algorithms, and create models and applications. MATLAB is known for its ease of learning and comprehensive support resources, making it accessible to both beginners and experts [21].

3.2 Python

Python is a high-level, interpreted programming language known for its easy-to-understand syntax and dynamic semantics. It is object-oriented and particularly suited for Rapid Application Development and as a scripting language. Python is popular for its emphasis on code readability and maintainability, support for modules and packages which facilitate code reuse, and a comprehensive standard library. It is freely available and distributable, even for commercial use. Python is appreciated for its efficiency, enabling programmers to achieve more with less code and less time [12].

3.3 Differences between MATLAB and Python

The differences between MATLAB and Python are crucial for translating a MATLAB script into Python. Unlike MATLAB, Python is free, while MATLAB requires a paid license. Therefore, translating a MATLAB script to Python can make the model more accessible for educational purposes. The following bullet points describe key differences between MATLAB and Python [22] [23]:

- **Syntax**: Python's syntax is designed with clarity and readability in mind, notably through its requirement of indentation to delineate code blocks within loops. Unlike Python, MATLAB does not rely on indentation to structure loops but uses an 'end' statement instead, which may make Python's syntax appear more readable to some. Additionally, MATLAB's syntax is optimized for mathematical operations with built-in libraries, whereas Python requires external libraries for similar functionalities. Notably, MATLAB indexes from 1, while Python starts from 0 [22].

- **Versality**: Python is renowned for its versatility across various domains like web development, data analysis, machine learning, and more, due to its extensive libraries and community support, whereas MATLAB relies heavily on its toolboxes and has a comparatively smaller community [22].

- **Community support**: Python has a more diverse community compared to MATLAB, partly because Python is free to use and has broader applications. According to a MathWorks article, LinkedIn searches as of May 2022 indicated approximately 7.6 million Python users and 4.1 million MATLAB users [23]. This showcases the large community surrounding Python.

  It's noteworthy that at the University of Stavanger, all engineering students are obligated to take a course titled "Introduction to Programming," which exclusively uses Python. This specificity suggests Python could be a more suitable choice for educational purposes in this context [24].

3.4 Debugging

Translating the script from MATLAB to Python and ensuring its functionality posed significant challenges. This chapter discusses the methods employed to address issues when the script did not perform as expected.

- **Handling ZeroDivisionError:** During the translation process, instances of ZeroDivisionError occurred, which typically happen when an attempt is made to divide by zero in the code. Initially, these errors led to the belief that there were incorrect implementations requiring redefinition to avoid such computational mistakes.

- **Dealing with NaN Values:** The occurrence of NaN (Not a Number) values was frequent. This issue often arises in data processing when operations result in undefined or unrepresentable values, especially during the translation of complex numerical computations.

These challenges listed over were solved by several steps:

- **Output Monitoring:** By printing the intermediate values computed by Python, and similarly in MATLAB, This made it possible to compare outputs between the two environments. Printing values helped in identifying discrepancies early in the debugging process.

- **Step-by-Step Debugging:** Utilizing Python's debugging tools, by executing the code line by line, paralleling this process in MATLAB. This method allowed for a meticulous comparison of the computed values at each step.

- **Automation of Repetitive Tasks:** Given the time-intensive nature of manual debugging, a tool named "OP Auto Clicker." was employed [25]. This tool automated the process of stepping through the code, enabling focus on monitoring and comparing the values from MATLAB and Python without the need to manually advance the debugger.

Through these methods, systematic addressing and resolution of the errors encountered during the script translation were achieved, enhancing the reliability and accuracy of the computational model.

3.5 Python libraries

Python libraries are collections of modules that provide functionalities for a wide range of programming tasks, without the need for writing them from scratch. These libraries can include anything from mathematical functions, and data manipulation to more complex tasks like web scraping, data analysis, machine learning, and scientific computing. The libraries that are utilized in this thesis are NumPy and Matplotlib [26].

3.5.1 NumPy

NumPy is a popular Python library used for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays [27]. In this thesis, NumPy is used primarily for numerical operations such as defining arrays for perforation locations and wellhead pressures, and for performing calculations on these arrays.

3.5.2 Matplotlib.pyplot

Matplotlib.pyplot is another Python library favoured for its data visualization capabilities, notably for its MATLAB-like syntax. This feature facilitates a smoother transition for those familiar with MATLAB to Python for creating graphs and plots. This library is used to visualize the results from our Python model. [28]

3.6 Main Python code: Wellhead pressures

This code is based on the MATLAB script provided by the supervisor in Appendix D, which has been translated into Python and is shown in Appendix A. This model is crafted to simulate the dynamic behavior of gas-liquid flow within petroleum wells, spanning from the reservoir to the wellhead. It initiates by mimicking fluid flow through the well's perforations, encompassing both the horizontal and vertical sections of the well. At the heart of the simulation is the iterative adjustment of pressure at the well's toe to match predefined wellhead pressures, guided by a predetermined accuracy threshold. In this simulation, iteration occurs over a fixed error limit of 0.005. This ensures the model's fidelity to real-world scenarios. This chapter offers a comprehensive breakdown of the procedural steps encoded in the simulation. Furthermore, this code is also based on isothermal conditions, meaning that there is no change in temperature, so temperature-related parameters are not included in the code.

3.6.1 Initialization of Constants and Parameters

In this segment of the simulation program, the foundational elements necessary for accurately modeling gas-liquid flow within petroleum wells are established. This involves defining a set of constants and parameters that represent the physical and operational characteristics of the well and the fluid dynamics within it. The constants defined are listed under:

- **Gravitational acceleration** (g): This constant is used to compute the hydrostatic pressure gradient within the fluid column.
- **Reservoir pressure** (Pr): The driving force for fluid flow from the reservoir into the well.
- **Boiling point pressure** (Pb): The pressure at which the fluid transitions between phases under well conditions.
- **Wellhead pressure** (Pwh): The pressure at the top of the well, critical for controlling the flow rate and well safety.
- **Length of horizontal well section** (Lh): Determines the extent of the well's horizontal reach, impacting fluid flow dynamics.
- **Height of vertical section** (Lv): Affects the hydrostatic pressure and flow characteristics in the vertical part of the well.
- **Segment length for horizontal section** (Dlh): Used in discretizing the well for simulation, impacting resolution and accuracy.
- **Segment length for vertical section** (Dlv): Similar to Dlh, but for the vertical segments, affecting simulation granularity.
- **Perforation inflow constant** (K): Characterizes the fluid influx through perforations, essential for modeling well productivity.
- **Oil density** (Rol): Influences the hydrostatic pressure calculations and fluid flow regimes within the well.
- **Gas density at reference condition** ($Rog0$): Key for calculating gas phase behavior and its interaction with the liquid phase.
- **Liquid viscosity** (Myl): Affects the fluid's resistance to flow, impacting pressure drops and flow rates.
- **Gas viscosity** (Nyg): Plays a role in determining the gas phase's flow characteristics and its interaction with the liquid phase.
- **Diameter of horizontal section** ($Dhor$): Affects the flow area available in the horizontal sections, influencing flow regimes and pressure drops.
- **Diameter of vertical section** ($Dver$): Similar to $Dhor$, but for vertical sections, impacting fluid dynamics and pressure profiles.

By initializing these constants and parameters, the groundwork is laid for the simulation, setting the stage for the complex interplay of physical forces and operational controls that will be explored in subsequent sections of the program. This initialization ensures that the simulation is grounded in realistic conditions, making the results it generates relevant and valuable for understanding and optimizing petroleum well operations.

3.6.2 Preprocessing

In the preprocessing phase of the simulation, essential steps are undertaken to organize the computational environment for effectively simulating dynamic fluid flow within the petroleum well. This phase is characterized by the spatial discretization of the well's trajectory, encompassing both its horizontal and vertical extents. Specifically, the script employs segment lengths (Dlh for horizontal and Dlv for vertical sections) to divide the well into discrete cells (Nhor and Nver respectively), forming a structured grid that underpins our numerical analyses. This grid allows for the meticulous examination of fluid dynamics at various well locations, captured by the linear array Lpos_hor for horizontal positions and Lpos_ver for vertical positions. Such discretization is pivotal for a granular evaluation of pressure, velocity, and fluid phase distribution throughout the well's length.

Moreover, the initialization of key arrays and variables is a critical component of this phase, enabling the tracking of evolving well conditions during the simulation. Of notable importance is the Perfor array, which delineates the perforation sites crucial for simulating fluid influx from the reservoir into the well. This meticulous setup of the computational framework during the preprocessing stage is indispensable for accurately simulating both multiphase and single-phase flows within the well. Enabled by the powerful NumPy library, this numerical analysis leverages arrays to handle complex calculations and data manipulations efficiently [27]. The precision in this preparatory work is fundamental to ensuring the reliability and accuracy of the simulation outcomes, shedding light on the behavior of gas and liquid phases under varied operational scenarios.

3.6.3 Main iteration over wellhead pressures

The iteration over the wellhead pressures constitutes a cornerstone of the simulation, meticulously engineered to calibrate the internal pressures within the well to correspond with a predefined sequence of wellhead pressures, denoted as PWHI in the code. This iterative calibration is crucial for the authentic simulation of gas-liquid flow dynamics in petroleum wells, capturing the intricate transition from the reservoir to the wellhead across diverse operational landscapes.

Initialization: The iterative journey commences with an array of wellhead pressures, PWHI, set for iteration. A baseline pressure of 247 bar (PWSTART) is initialized throughout the well's extent. Setting this baseline pressure is crucial as it provides a controlled starting condition from which precise adjustments can be made during the simulation. PWSTART is the pressure in the first cell, an assumption made for the purposes of the analysis.

Iterative refinement loop: In this iterative loop, the simulation systematically refines the initial pressures to align with the target pressures specified in PWHI. The loop's structure is outlined as follows:

- **Pressure adjustment:** Each iteration begins with adjustments to the pressure at the well's base (PWSTART), initially set to 247 bar, based on the relative error (Relerr). This error quantifies the deviation from the target wellhead pressure, guiding iterative adjustments to ensure gradual convergence.
- **Dynamic parameter updates:** Alongside pressure adjustments, the simulation updates key fluid dynamic parameters, reflecting the new pressure conditions. This includes recalculating velocities for the liquid (Ulsi) and gas (Ugsi) phases, the overall mixture velocity (Umix), and the gas fraction after slip calculation (Epsgi), accommodating the well's complex internal dynamics.
- **Mass inflow computation:** A critical aspect of each iteration is the assessment of total mass inflow (Mtot), which is intricately tied to the dynamics at the perforations along the wellbore, specifically at distances of 0 m, 200 m, 400 m, 600 m, and 800 m from the toe. This involves evaluating the influence of each perforation's capacity to facilitate fluid flow from the reservoir into the wellbore, a capacity primarily determined by the perforation inflow constant (K). In the simulation, this evaluation is synthesized through an array, Kxi, where each element mirrors the contribution of an individual perforation to the overall fluid influx, based on its K value set to 0.0013. This step is essential for accurately simulating the well's productivity, as it encapsulates the combined effect of all perforations, taking into account the reservoir pressure, the existing pressure within the well at each perforation point, and the inherent properties of the flowing fluids with equation 2. By meticulously aggregating the contributions from each perforation, the simulation provides a comprehensive view of how perforations collectively influence the well's ability to draw fluids from the reservoir.
- **Boiling pressure considerations:** The simulation checks for phase changes, especially when fluid pressure dips below the boiling pressure (Pb). Under these conditions, adjustments are made to the gas fraction (Epsgi) by updating the slip ratio through equation 11, along with various physical properties of the fluid.

Convergence Criterion: The iterative loop includes a convergence check against a predefined error limit (Errlimit). This assessment involves comparing the relative error (Relerr) with Errlimit to determine the need for further iterations for the current wellhead pressure. Once Relerr falls within Errlimit, indicating sufficient alignment of simulated and target pressures at the wellhead, the simulation progresses to the next wellhead pressure in the PWHI sequence. This iterative process is repeated for each wellhead pressure, enabling a comprehensive exploration of their impacts on well dynamics. The relative error (Relerr) can be represented as:

$$\text{Relerr} = \frac{0.8 \times (P_{\text{wh, calc}} - P_{\text{wh}})}{P_{\text{wh}}} \tag{18}$$

where:

- $P_{\text{wh, calc}}$ represents the calculated wellhead pressure by simulation
- P_{wh} represents the wellhead pressure that the simulation aims to iterate towards

3.6.4 Pressure Drop Calculations

In this critical section of the simulation, the calculation of pressure drops along the petroleum well is addressed, a fundamental aspect in understanding the fluid dynamics within the wellbore. The pressure drop calculations are pivotal for assessing the flow performance and operational efficiency of the well, as they directly impact the fluid's velocity and the well's overall productivity.

The simulation incorporates a detailed methodology to compute the frictional and hydrostatic pressure drops that occur as the fluid moves through both the horizontal and vertical sections of the well. The frictional pressure drop accounts for the resistance encountered by the flowing fluid due to the internal surface roughness of the well and the interaction between the fluid particles, which is particularly significant in turbulent flow regimes. The nature of the frictional pressure drop—whether the flow is laminar or turbulent—is determined by the Reynolds number, a key factor discussed in the theory.

Hydrostatic pressure drop calculations are essential for the vertical sections of the well, where the fluid column's weight contributes significantly to the overall pressure profile. This aspect of the simulation takes into account the fluid's density, which can vary significantly between the gas and liquid phases, and the gravitational force, to determine the pressure changes attributable to the fluid column's height.

Moreover, the simulation models the impact of slip between gas and liquid phases on pressure drop calculations. Slip, the relative velocity difference between gas and liquid phases, can alter the flow characteristics and, consequently, the pressure profile along the well. Accurately modeling slip is crucial for predicting the well's performance under various operational scenarios.

By meticulously calculating these pressure drops, this section of the simulation provides insights into the fluid's behavior and flow regimes within the well, enabling the identification of potential bottlenecks or inefficiencies.

3.6.5 Visualization

The visualization stage of the simulation leverages Python's Matplotlib.pyplot library, a powerful tool for rendering intricate numerical data into clear, comprehensible graphical formats. Each visual representation is carefully constructed to enhance the interpretability of the simulation's findings.

3.7 Supplementary code: Boiling pressures

A supplementary code is introduced in order to investigate for reservoir production over different boiling pressures. This supplementary code is based on the main code with slightly adjustments. A key modification in the main code is the introduction of a new loop to iterate over a range of wellhead pressures, represented by:

```
for Pwh in Pwh_values: # Added loop for different wellhead pressures
```

This change allows the script to simulate and analyze the influence of varying wellhead pressures on reservoir production at each boiling pressure point. It enriches the analysis by offering insights into the combined effects of both parameters on the system's performance.

Additionally, the original script's primary loop:

```
for ipwh, Pwh in enumerate(PWHI):
```

has been altered to:

```
for ipwh, Pb in enumerate(PB):
```

The focus has shifted from iterating over a predefined set of wellhead pressures to exploring a range of boiling pressures. This change in the iteration mechanism is tailored to examine how boiling pressures affect the production rates across different wellhead pressures.

The result is a multi-variable analysis that illustrates the dynamics of reservoir production under various scenarios of boiling and wellhead pressures, providing a comprehensive view of the reservoir's behavior. The alterations to the code reflect a strategic pivot towards a more complex understanding of the production characteristics of the reservoir, facilitating the identification of optimal operational parameters. The code is available in Appendix B.

3.8 Supplementary code: Total pressure gradient

The following Python script, shown in Appendix C, is used to analyze the pressure gradients in a vertical section of a well. The script considers both frictional and hydrostatic pressure gradients as functions of varying gas velocities Ugsi, while keeping a constant liquid velocity Ulsi. The script uses a variety of constants, including the gravitational acceleration g, the diameter of the vertical section Dver, oil density in liquid phase ρ_O, and gas density at reference conditions ρ_{g0}. These values are applied to calculate the mixture density and viscosity, which are then used to determine the Reynolds number and friction factor for the flow. Using the calculated Reynolds number and friction factor, the script computes the frictional pressure gradient $\frac{dP_{\text{fric}}}{dL}$ known as Dpdxf in the code and hydrostatic pressure gradient $\frac{dP_{\text{hyd}}}{dh}$ known as Dpdz in the code for a range of gas velocities. It also computes the total pressure gradient $\frac{dP_{\text{tot}}}{dh}$ also known as DpdzTotal in the code, as the sum of these two components. The results are plotted to show the relationship between gas velocity and pressure gradient for the chosen parameters. This script provides valuable insights into the behavior of pressure gradients in a vertical well section, helping to understand the interplay between gas and liquid flow and their effects on pressure drops within the well. The computed pressure gradients as functions of gas velocity are plotted to offer visual insights into the impacts of different velocities on the pressure within the well. The approach and results of this simulation, detailed in the code listed in Appendix C, provide a profound understanding of the dynamics involved in the vertical well flow under various operational conditions.

4 Discussion and results

4.1 Constants displayed in Table 1

In the Discussion and results chapter, the constants and initial data outlined in Table 1 will be applied to the numerical Python model to simulate gas-liquid flow in petroleum wells. The outcomes of the model, generated using these parameters, will be thoroughly analyzed to provide insight into the flow dynamics.

Table 1: Constants and Initial Data for the Numerical Model from Appendix A

Constant	Value with Units
Gravitational acceleration	$g = 9.81\,\text{m/s}^2$
Reservoir pressure	$P_r = 240\,\text{bar}$
Boiling point pressure	$P_b = 175\,\text{bar}$
Length of horizontal section	$L_h = 1000\,\text{m}$
Height of vertical section	$L_v = 2000\,\text{m}$
Segment length horizontal	$\Delta L_h = 10\,\text{m}$
Segment length vertical	$\Delta L_v = 20\,\text{m}$
Perforation inflow constant	$K = 1.3 \times 10^{-3}\,\text{m}^3/\text{s} \cdot \text{Pa}$
Oil density	$\rho_o = 800\,\text{kg/m}^3$
Gas density at reference condition	$\rho_{g0} = 1.5\,\text{kg/m}^3$
Dynamic liquid viscosity	$\mu_L = 2 \times 10^{-3}\,\text{Pa s}$
Kinematic gas viscosity	$\nu_G = 2 \times 10^{-5}\,\text{m}^2/\text{s}$
Diameter of horizontal section	$D_{hor} = 0.15\,\text{m}$
Diameter of vertical section	$D_{ver} = 0.20\,\text{m}$
Wellhead pressures to iterate over	$P_{WH} = [60, 61, 62, \ldots, 80]$

4.2 Results from Table 1

The results obtained from the data in Table 1 enable us to generate a series of plots for wellhead pressures ranging from 60 to 80 bar. However, for the sake of clarity and conciseness in this report, The focus will be on examining plots for three specific wellhead pressures: 60, 70, and 80 bar. This targeted selection provides a representative view of the variations in results with changes in wellhead pressure. For those interested in a comprehensive review of the entire pressure range, the script in Appendix A is available to reproduce all the corresponding plots.

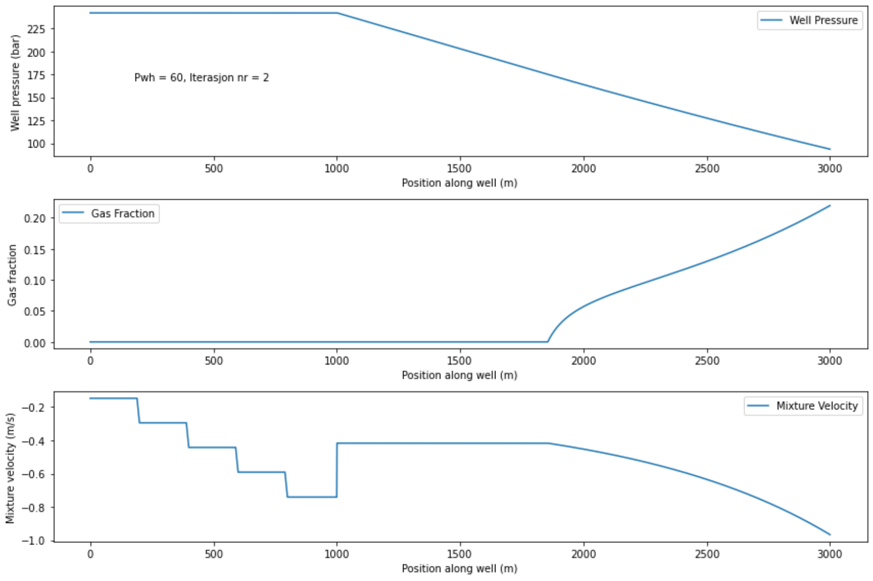

Figure 9: First iteration at 60 bar

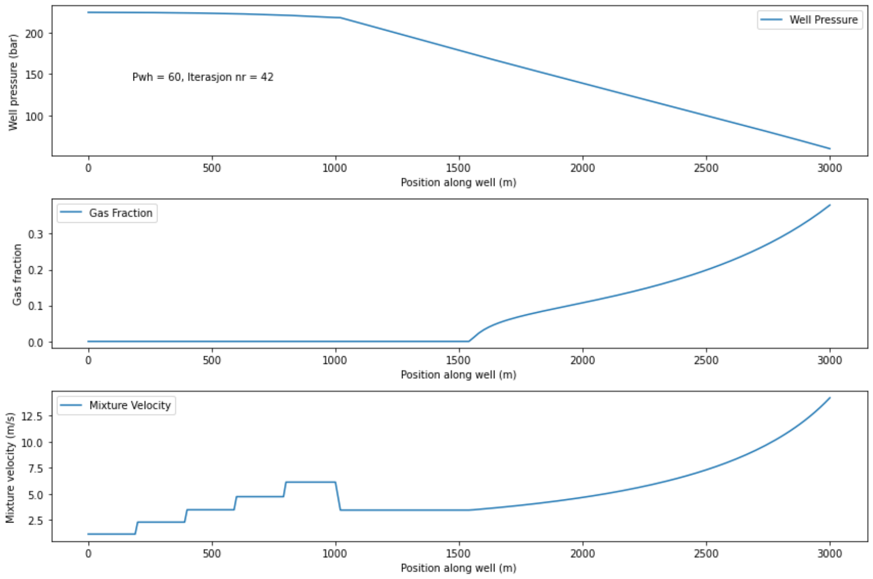

Figure 10: Last iteration at 60 bar

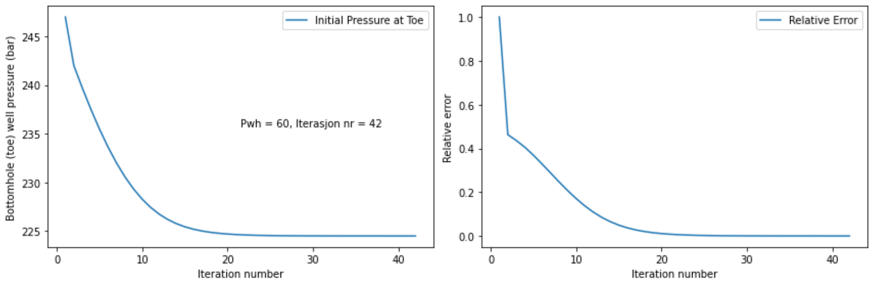

Figure 11: Iteration plot for 60 bar

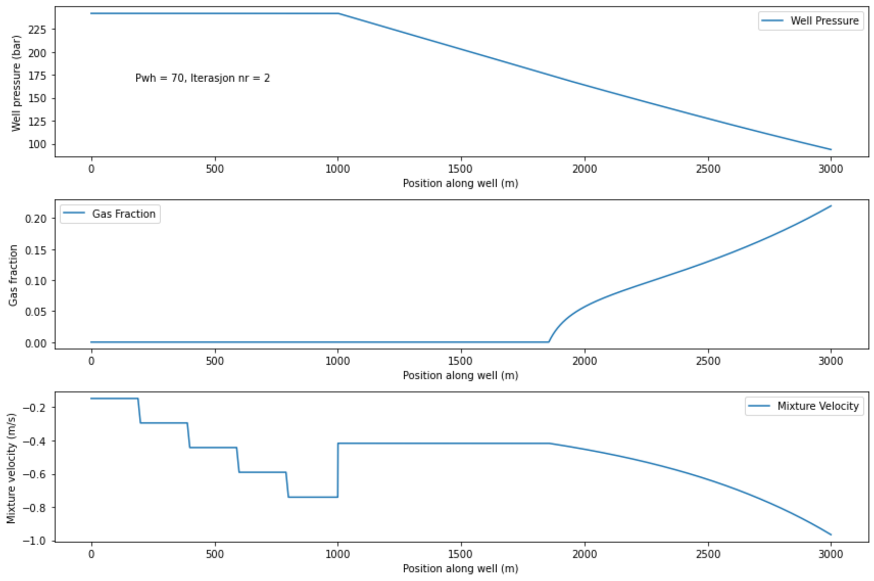

Figure 12: First iteration at 70 bar

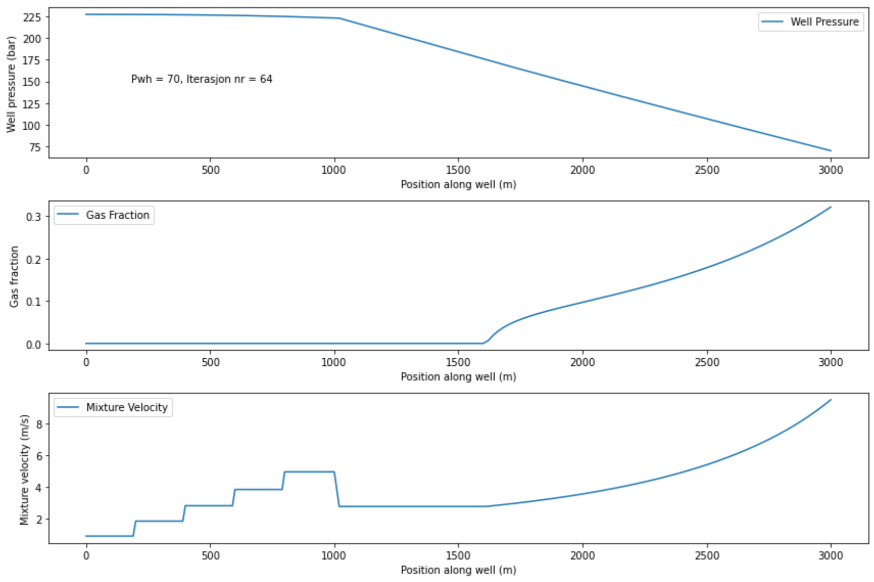

Figure 13: Last iteration at 70 bar

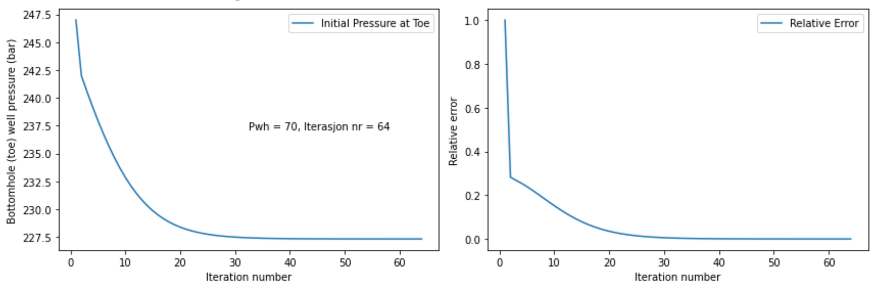

Figure 14: Iteration plot for 70 bar

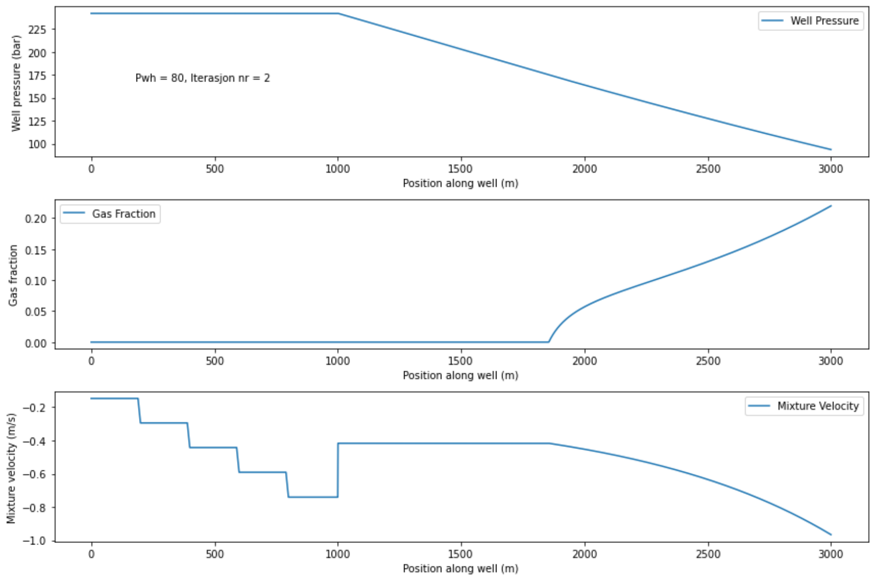

Figure 15: First iteration at 80 bar

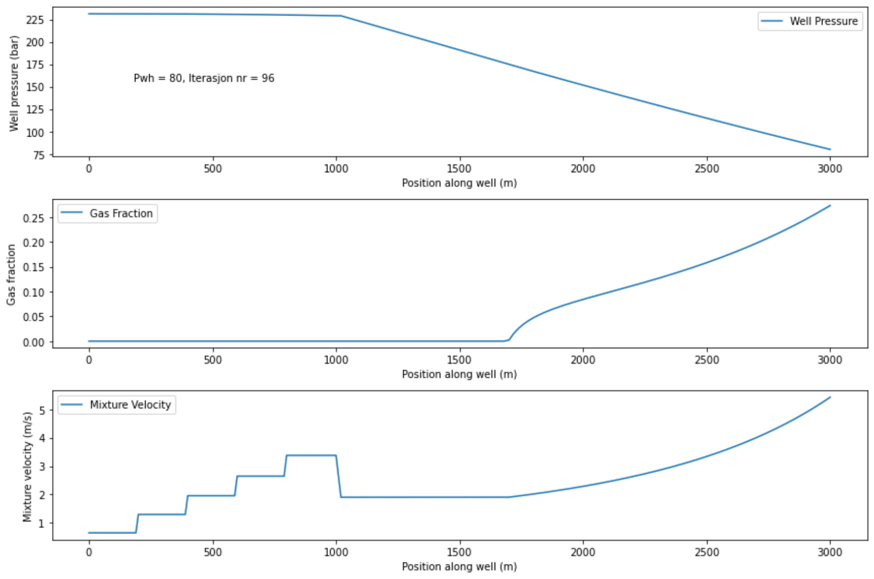

Figure 16: Last iteration at 80 bar

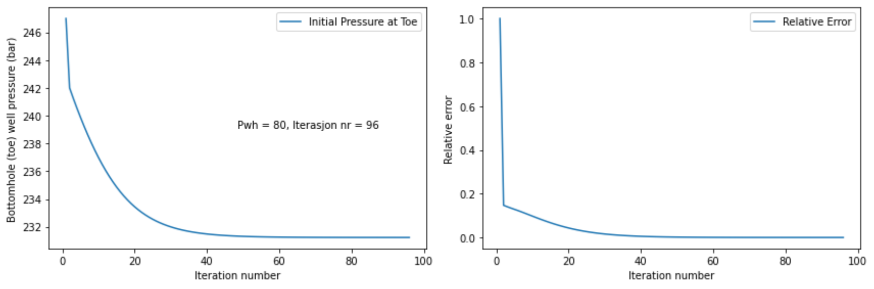

Figure 17: Iteration plot for 80 bar

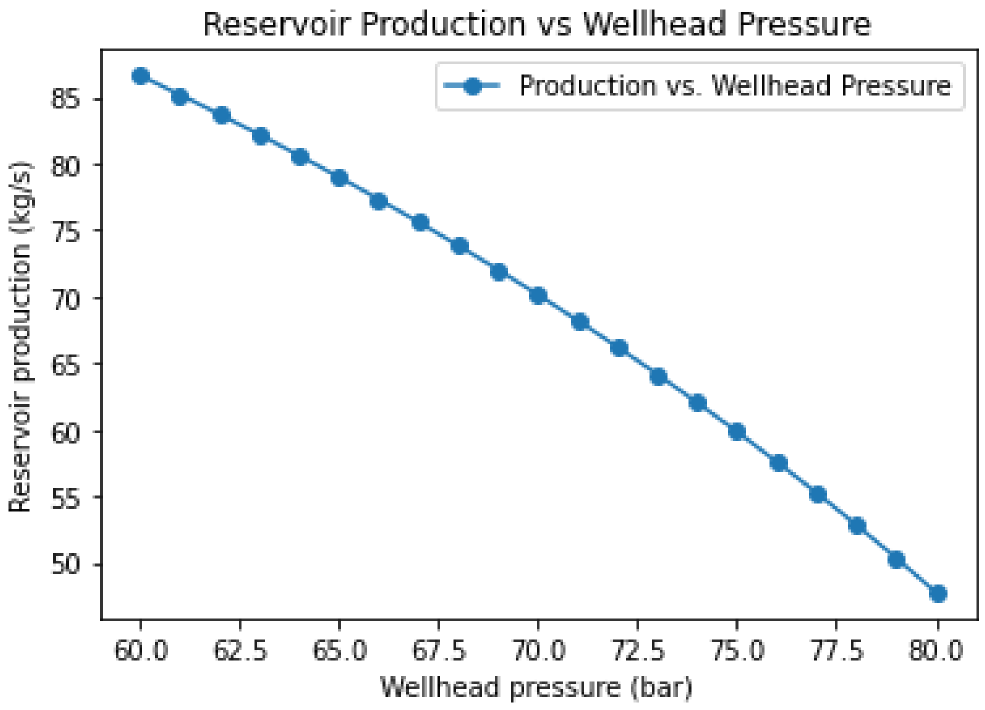

Figure 18: Reservoir production against wellhead pressures

4.3 Results Analysis

4.3.1 Wellhead Pressure Analysis

Wellhead Pressure	Maximum Gas Fraction	Maximum Mixture Velocity	Bottomhole Well Pressure
60 Bar	35%	13 m/s	225 Bar
70 Bar	30%	9 m/s	227.5 Bar
80 Bar	25%	5 m/s	232 Bar

Table 2: Wellhead Pressure results

The key results from Chapter 4.2 are presented in Table 2, the trends in wellhead pressure, gas fraction, mixture velocity, and bottomhole well pressure highlight key characteristics of multiphase flow behavior. Note that the analysis will focus on the plots from the final iteration, as this is the point at which the relative error, denoted as Relerr, reaches its minimum value from Equation 18, ensuring that the data used are at their highest accuracy. This is evident in the right plot of figure 17 for a wellhead pressure of 80 bar, and similarly applies to pressures of 60 and 70 bar. Values in Table 2 are approximations.

At a wellhead pressure of 60 bar, the maximum gas fraction is approximately 35%, with a corresponding maximum mixture velocity of around 13 m/s and a bottomhole well pressure of about 225 bar. As the wellhead pressure increases to 70 bar, the maximum gas fraction decreases to 30%, while the maximum mixture velocity declines to around 9 m/s, with a bottomhole well pressure of approximately 227.5 bar.

Further increasing the wellhead pressure to 80 bar results in a maximum gas fraction of around 25%, a maximum mixture velocity of approximately 5 m/s, and a bottomhole well pressure of about 232 bar.

These observations indicate that as the wellhead pressure increases, the gas fraction and mixture velocity decrease, while the bottomhole well pressure increases. This trend reflects the fluid phase behavior's dependency on pressure, with higher wellhead pressures delaying gas breakout and maintaining the fluid in a liquid state over a more extended section of the wellbore. It is also noted that there appears to be no pressure loss in the horizontal sections, as seen in figure 16 for a wellhead pressure of 80 bar, which is consistent with observations at 60 and 70 bar. In real-life scenarios, we would typically expect to observe pressure losses in horizontal sections due to factors like friction. This discrepancy suggests areas for future work, potentially involving more detailed modeling of frictional losses in the horizontal sections to enhance the model's predictive accuracy.

4.3.2 Reservoir production

Figure 18 elucidates the effect of wellhead pressure on reservoir production rates, revealing a distinct downward trend. As the wellhead pressure increases from 60 to 80 bar, production rate notably decreases from approximately 85 kg/s to around 50 kg/s. This pattern may be logical due to the decrease in mixture velocity, as we increase wellhead pressure seen in Table 2.

Lower wellhead pressures enhance the pressure differential between the reservoir and the wellhead, thus increasing the available driving pressure for the flow of reservoir fluids, which facilitates higher production rates. Conversely, higher wellhead pressures reduce this differential, diminishing the driving force necessary for fluid movement and thereby decreasing production rates.

This inverse relationship is further supported by the phenomenon of delayed gas separation at higher pressures within the vertical sections of the well. The predominance of the denser liquid phase means that the hydrostatic pressure within the wellbore is increased by using Equation 17, However, it's crucial to note that this hydrostatic pressure gradient does not constitute resistance to fluid flow in the same way as friction or viscosity does. Instead, it reflects the static pressure due to fluid weight, which is distinct from the flow resistance caused by the fluid's viscous properties

Our analysis corroborates the findings from earlier figures detailing wellbore fluid dynamics and presents a coherent narrative of how wellhead pressure impacts production from the reservoir to the surface. The trend demonstrated in Figure 18 aligns with the independent research by Adeyemi, who reports an "There exist an inverse relationship between wellhead pressure and gas production rate, that is, gas production rate increases as the wellhead pressure decreases" [29] echoing the behavior captured in our surface performance curve. This concordance not only validates our physical model but also reinforces the analysis's reliability.

4.4 Model Validation for single-phase Flow

To validate the model for conditions of single-phase flow, a hypothetical scenario is considered where the wellhead is completely shut, effectively stopping oil production. To guarantee that the oil remains in a liquid state and does not boil, the boiling pressure P_b is set to a value below the minimum pressure expected in the well. It is assumed $P_b = 1$ bar, ensuring that as long as the computed wellhead pressure is above this value, the liquid will not boil. This creates a safe margin as the operating pressures within the well will always be significantly greater than P_b, maintaining single-phase flow conditions. With only changing P_b from 240 bar to 1 bar from Table 1, the bottom hole pressure is calculated using the hydrostatic pressure gradient as defined by Equation 17 with the following values listed under:

- Oil density $(\rho_O) = 800\ \text{kg/m}^3$
- Gravitational acceleration $(g) = 9.81\ \text{m/s}^2$
- Height of the vertical well section $(h) = 2000$ m

By applying Equation 17, the pressure difference (ΔP_{hyd}) due to the hydrostatic pressure is computed:

$$\Delta P_{\text{hyd}} = \rho_O \cdot g \cdot h = 800\ \text{kg/m}^3 \cdot 9.81\ \text{m/s}^2 \cdot 2000\ \text{m} = 15,696,000\ \text{Pa} \tag{19}$$

To determine the hydrostatic pressure at the wellhead when the fluid is at rest, the hydrostatic pressure is subtracted from the reservoir pressure obtained from Table 1:

$$P_{\text{wh},0} = \left(\frac{240 \times 10^5\ \text{Pa} - 15,696,000\ \text{Pa}}{10^5\ \text{Pa/bar}} \right) \text{bar} = 83.04\ \text{bar} \tag{20}$$

According to theory, setting the wellhead pressure to 83.04 bar would result in a static condition with no production, consistent with the closed-wellhead scenario. Then, iterating the wellhead pressures from 80 to 84 bar produces the following plot.

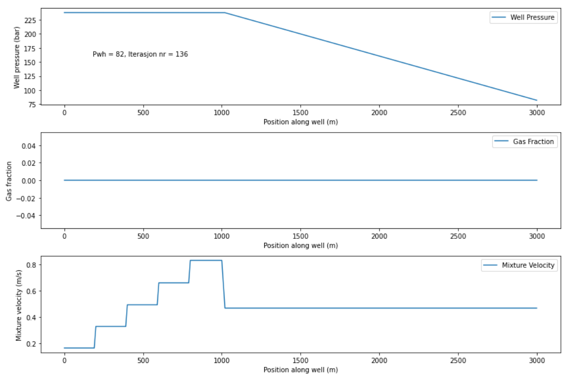

Figure 19: Plot proving single-phase flow for $P_{\text{wh}} = 82$ bar

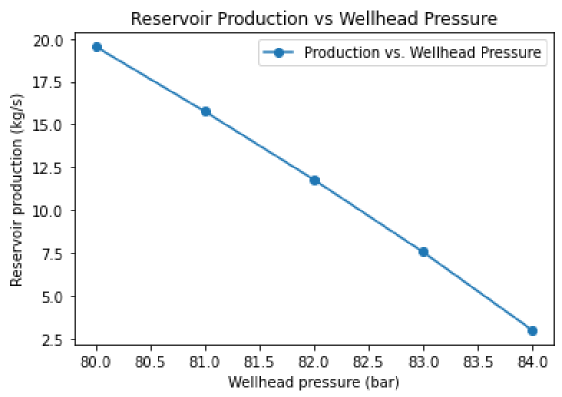

Figure 20: Reservoir production with Dlv=20 with P_{wh} iterating from 80 to 84 bar

From Figure 19, it is observed that the flow is single phase, as evidenced by the gas fraction remaining constantly at 0. This indicates that the flow consists solely of liquid, a crucial condition for the application of Equation 17, which assumes single-phase flow.

In Figure 20 it is observed that the model predicts a production rate of approximately 7.5 kg/s at a wellhead pressure of 83 bar. Theoretically, no oil production is expected under these conditions, since the wellhead pressure equals the calculated hydrostatic pressure at the top of the well. This discrepancy between the model and theory can be attributed to the choice of $D_{lv} = 20$, which defines the discretization interval of the vertical section of the well.

Setting $D_{lv} = 20$ means that the vertical section of the well is divided into segments of 20 meters each. The discretization of a continuous system into a finite number of segments for computational modeling purposes inherently introduces an approximation. Specifically:

- When the interval D_{lv} is too large, the model may not accurately capture the incremental changes in pressure due to the hydrostatic effect along the well's depth. This is because the hydrostatic pressure, which should vary continuously with depth, is instead calculated at discrete intervals, assuming a uniform pressure within each segment.

- As a result, there could be an underestimation of the cumulative hydrostatic pressure when aggregating the contributions from each segment. This underestimation may lead to a lower predicted pressure at the bottom hole than the actual value, thereby suggesting the possibility of production where there should be none.

- In essence, the model's resolution is a key factor in accurately capturing the pressure profile within the well. A higher resolution (smaller D_{lv}) would lead to a finer approximation and potentially more accurate predictions, aligning closer with the theoretical expectation of no flow.

Thus, by choosing a smaller value for D_{lv}, such as 1 meter, the model can better approximate the continuous nature of the hydrostatic pressure gradient, resulting in a prediction that adheres more closely to the theoretical no-flow condition proved by the plot under:

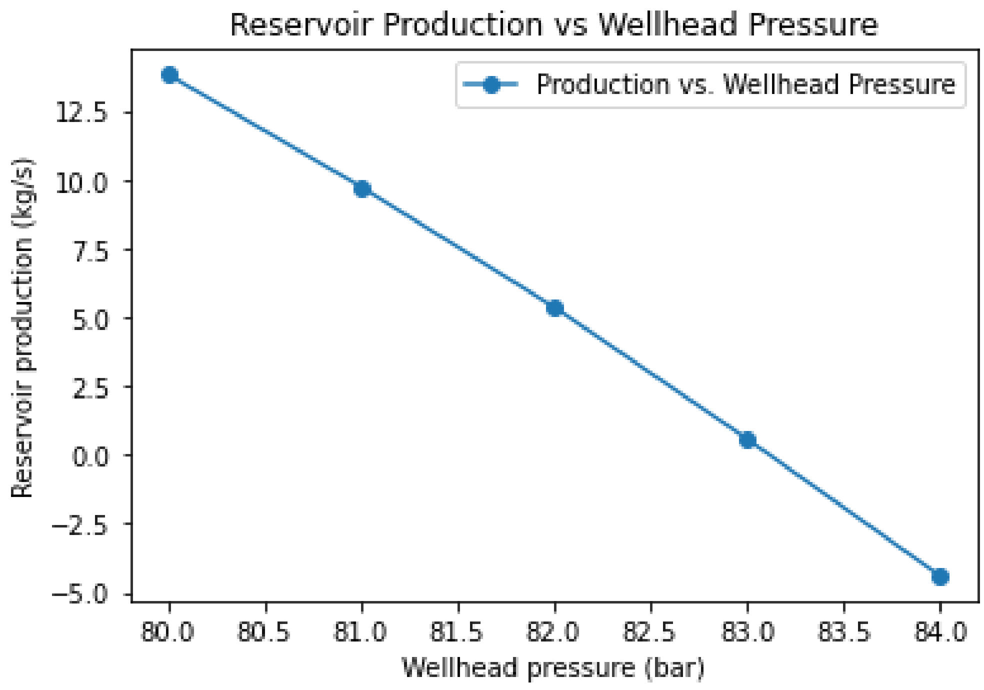

Figure 21: Reservoir production with Dlv=1

Observing Figure 21, it is noted that at wellhead pressure of 83 bar, the oil production is almost negligible, aligning with theoretical expectations. However, an important trend is observed where the model predicts negative production rates at wellhead pressures exceeding 83 bar. The maximum wellhead pressure that can be sustained by the natural reservoir drive is 83.04 bar; pressures above this threshold cause the well to cease production and remain static. If the wellhead pressure is decreased towards atmospheric levels, a substantial increase in flow rate is expected due to the enhanced pressure differential driving the flow. Conversely, increasing the wellhead pressure beyond this natural limit, which necessitates external pumping, converts the well from a producer to an injector.

The model's prediction of negative production rates at wellhead pressures beyond 83 bar is a computational artifact that suggests an inversion of flow direction, which is not physically plausible under normal production scenarios. In reality, increasing the wellhead pressure above the reservoir's natural drive would not result in oil production; instead, the well would cease to produce. Such pressures would require external forces, transforming the function of the well from extracting to potentially injecting fluids back into the reservoir, a situation contrary to typical production operations. Therefore, while the model indicates negative production rates, this outcome does not align with the actual physical behavior of a production well, underscoring the limitations and boundaries of the model's applicability.

This effect is further illustrated in Figure 22, which uses the script from Appendix B to model production changes under varying boiling pressures. This figure uses the constants from Table 1, where the boiling pressure (P_b) is now set as an array ranging from 30 to 250 bar with step-size of 10, and the wellhead pressure is fixed at 83.04 with Dlv equal to 1, all other parameters are constant as per Table 1.

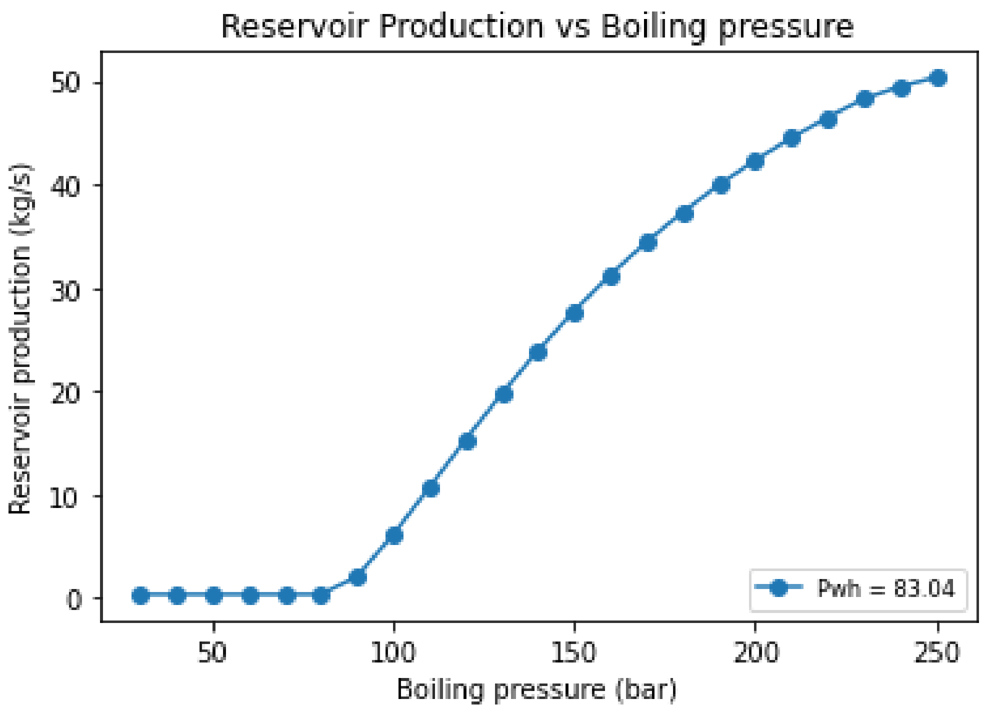

Figure 22: Plot for changing boiling pressures and reservoir production at fixed wellhead pressure = 83.04 bar

As depicted in Figure 22, an increase in boiling pressure corresponds to an increase in production. This trend is attributed to the density changes that occur as the liquid begins to vaporize into gas, which has a significantly lower density than the liquid phase. The oil density is given as $800\,\mathrm{kg/m^3}$ and the gas density at reference conditions is $1.5\,\mathrm{kg/m^3}$. As boiling progresses, the density of the produced mixture decreases due to the increasing proportion of gas, calculated using Equation 19. This reduction in mixture density leads to a decrease in hydrostatic pressure, which in turn results in a higher pressure at the wellhead, seen in Equation 20. Therefore, with a higher boiling pressure inducing faster vaporization, the production rate increases as a consequence of the declining hydrostatic pressure from a less dense fluid mixture. The mixture velocity discussed in chapter 4.3 also increases with higher boiling pressure. This is because there will be more gas displayed with increased boiling pressure, which is another explanation for why the production increases.

4.5 Boiling pressure analysis

This chapter aims to analyze how varying boiling pressures in conjunction with changing wellhead pressures impact reservoir production. Table 3 below lists the constants used for the Python script presented in Appendix B.

From Table 3, boiling pressures were varied from 30 to 80 with a step size of 5, while wellhead pressures were varied simultaneously. A plot will be generated with reservoir production on the y-axis and boiling pressures on the x-axis, featuring curves that correspond to different wellhead pressures. The results showcased below reveal complex trends.

Table 3: Constants and Initial Data for the Numerical Model from Appendix B

Constant	Value with Units
Gravitational acceleration	$g = 9.81\,\mathrm{m/s^2}$
Reservoir pressure	$P_r = 240\,\mathrm{bar}$
Wellhead pressures to iterate over	$P_{WH} = [30, 35, 40, \dots, 80]$
Length of horizontal section	$L_h = 1000\,\mathrm{m}$
Height of vertical section	$L_v = 2000\,\mathrm{m}$
Segment length horizontal	$\Delta L_h = 10\,\mathrm{m}$
Segment length vertical	$\Delta L_v = 1\,\mathrm{m}$
Perforation inflow constant	$K = 1.3 \times 10^{-3}\,\mathrm{m^3/s \cdot Pa}$
Oil density	$\rho_o = 800\,\mathrm{kg/m^3}$
Gas density at reference condition	$\rho_{g0} = 1.5\,\mathrm{kg/m^3}$
Dynamic liquid viscosity	$\mu_L = 2 \times 10^{-3}\,\mathrm{Pa\ s}$
Kinematic gas viscosity	$\nu_G = 2 \times 10^{-5}\,\mathrm{m^2/s}$
Diameter of horizontal section	$D_{hor} = 0.15\,\mathrm{m}$
Diameter of vertical section	$D_{ver} = 0.20\,\mathrm{m}$
Boiling point pressures to iterate over	$P_{WH} = [30, 40, 50, \dots, 270]$

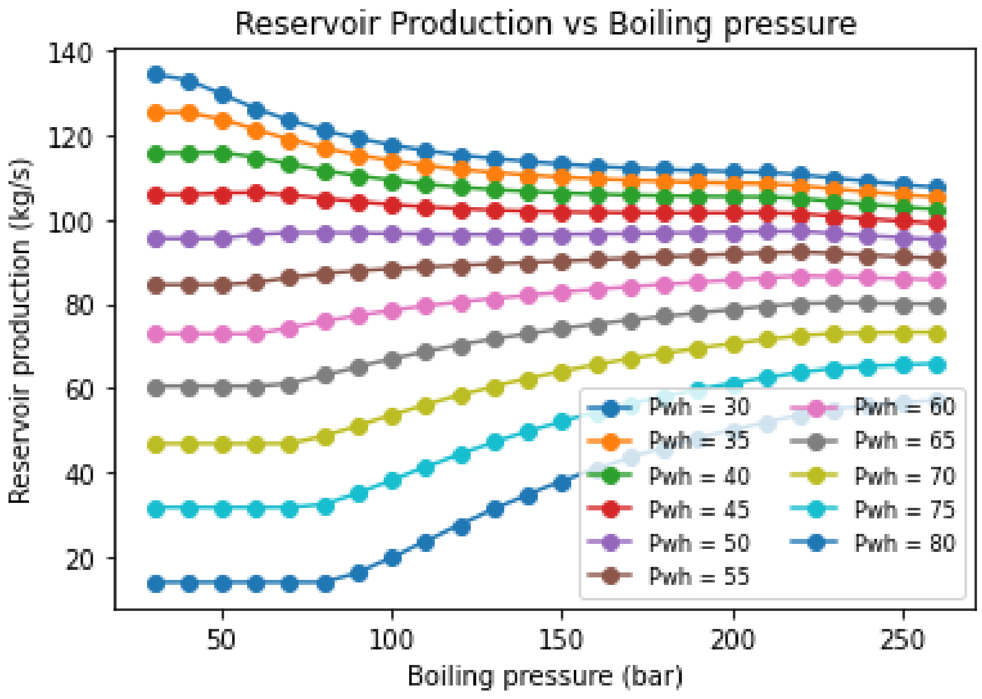

Figure 23: Production plot with varying wellhead pressure curves over different boiling pressures obatined from Appendix B

From Figure 23, the brown curve representing a wellhead pressure of 55 bar corresponds to a production rate of approximately 80 kg/s at a boiling pressure of 50 bar.At a lower wellhead pressure of 45 bar (red curve), while maintaining a constant boiling pressure of 50 bar, the production rate increases to around 110 kg/s. This suggests that lower wellhead pressures result in higher production rates when the boiling pressure is kept constant. This inverse relationship between wellhead pressure and gas production rate has also been highlighted by Adeyemi [29]. However, Figure 23 reveals intriguing results when examining the trend of increasing boiling pressures. Chapter 4.4 of the report discusses how increased boiling pressures lead to higher gas fractions in the wellbore, as oil transitions to gas more rapidly. This was supported by calculations using Equation 19, which indicated that higher boiling pressures reduce hydrostatic pressure, potentially increasing oil production. Contrary to these expectations, for wellhead pressures above 45 bar, an opposite trend is observed: as boiling pressure increases, production decreases. This discrepancy suggests that factors beyond hydrostatic pressure, discussed in Section 4.4, significantly influence production outcomes. Further analysis on this phenomenon will be explored in the next chapter.

4.6 Third script: Hydrostatic and friction gradient

This chapter will explore the reasons behind wellhead pressures causing production rates to decrease with increasing boiling pressure. In order to explain this, this report will use the code from Appendix C to explain this.

Table 4: Constants and Initial Data for the Numerical Model from Appendix C

Constant	Value with Units
Gravitational acceleration	$g = 9.81\,\mathrm{m/s^2}$
Diameter of vertical section	$D_{ver} = 0.20\,\mathrm{m}$
Oil density	$\rho_o = 800\,\mathrm{kg/m^3}$
Gas density at reference condition	$\rho_{g0} = 1.5\,\mathrm{kg/m^3}$
Dynamic liquid viscosity	$\mu_L = 2 \times 10^{-3}\,\mathrm{Pa\ s}$
Kinematic gas viscosity	$\nu_G = 2 \times 10^{-5}\,\mathrm{m^2/s}$
Reservoir pressure	$P_r = 240\,\mathrm{bar}$

Utilizing the constants listed in Table 4, the following plot is generated and shown in Figure 24.

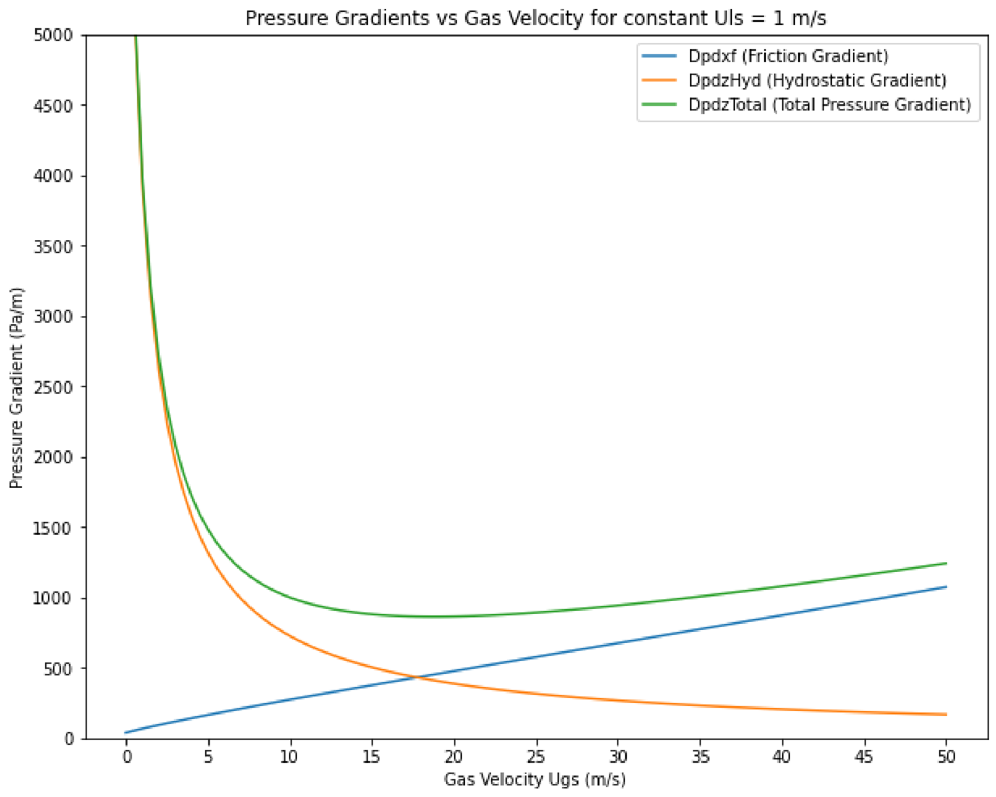

Figure 24: Pressure gradients against increasing gas velocity

Observing Figure 24, it's apparent that the hydrostatic pressure gradient ($DpdxHyd$) decreases as gas velocity increases. This suggests that with increased gas presence in the wellbore (i.e., a higher gas fraction), the hydrostatic pressure decreases, thereby resulting in a higher production rate. However, this occurs when the friction gradient ($Dpdxf$) is small enough to not significantly impact the total pressure gradient ($DpdxTotal$), which can be seen for gas velocities below 17 m/s.

When the gas fraction increases, which happens at lower wellhead pressures as explained in Table 2, the friction gradient also increases. This explains the opposite trend, where increasing boiling pressures result in decreased reservoir production, as seen in Figure 23.

In Figure 24, the friction gradient surpasses the hydrostatic pressure gradient at a gas velocity of approximately 17 m/s, marking the point where the friction gradient becomes the dominant contributor to the total pressure gradient. The total pressure gradient increases with rising gas velocity, potentially explaining the inverse relationship observed: increasing the boiling pressure leads to decreased reservoir production.

Figure 24 also indicates an optimal point where the total pressure gradient is at its lowest, which occurs where the friction gradient and hydrostatic pressure gradient intersect. This point, at a gas velocity of approximately 17 m/s, represents the lowest total hydrostatic pressure, potentially providing the highest lift, assuming the wellhead pressure remains unchanged. For this scenario with constant liquid velocity, this intersection will be crucial in optimizing oil production as it represents where the total pressure gradient is at its lowest.

5 Conclusion

The objective of this thesis was to develop a Python script for predicting reservoir production in petroleum wells, considering parameters such as wellhead pressure, gas fraction, mixture velocity, and boiling pressures. The study utilized three distinct Python scripts to evaluate these factors. The results were:

- **Validation Against single phase Flow:** The Python model aligned with theoretical expectations for single phase flow, accurately simulating fluid dynamics in wells. This validated the model's reliability. Using theories of hydrostatic pressure and assuming a closed wellhead, the computed wellhead pressure of 83.04 bar, as shown in Table 1. The findings showed that the model predicted zero oil production. However, when wellhead pressures are higher than 83.04 bar, negative values are obatined, which is not plausible in real-life scenarios. Therefore, for further work, modifications would be needed in this model to prevent negative values.

- **Influence of Wellhead Pressures:** The study showed that higher wellhead pressures led to lower gas fractions, reduced mixture velocities, higher bottomhole well pressures and reduced oil production, which is consistent with fluid behavior in multiphase flow.

- **Impact of Boiling Pressures:** The relationship between boiling pressures and reservoir production exhibited complex trends. Initially, increased boiling pressure enhanced production due to reduced hydrostatic pressure, but at lower wellhead pressures, frictional pressure became dominant, thereby diminishing production.

Overall, the thesis established a robust framework for understanding gas-liquid flow dynamics in petroleum wells. The developed Python model demonstrated effectiveness in digital modeling within the energy sector. This work underscores the importance of precise computational models in petroleum engineering. The findings of this report reveal how wellhead pressures, along with boiling pressures, affect production, which will help in optimizing oil production.

In future studies, several areas could be explored to further enhance the capabilities and applicability of the developed Python model:

- **Incorporating Temperature Variations:** Currently, the model assumes constant temperature changes. Future work could involve incorporating temperature variations into the simulation, as temperature fluctuations can significantly impact fluid properties and flow behavior in petroleum wells.

- **Enhancing Horizontal Section Modeling:** The simplification of the horizontal section may limit the model's accuracy, especially in horizontal well configurations. Future research could focus on incorporating appropriate models and physics to better simulate fluid flow in horizontal section.

- **Validation Against Experimental Data:** While the model has been validated against theoretical expectations, further validation against experimental data from real-world petroleum wells would strengthen its reliability and applicability. Future work could involve conducting experimental studies or collaborating with industry partners to gather relevant data for validation after improving the model.

By addressing these aspects in future work, the model can be further refined and expanded to better simulate real-world gas-liquid flow dynamics in petroleum wells, ultimately enhancing its utility for reservoir management and optimization.

6 Appendix

A Main Python code: Wellhead pressures

The script presented below is annotated with comments, utilizing the same notation conventions as described in chapter 3.6. The multiple hashtags (#) signify commentary lines intended to elucidate the accompanying code segments.

```
# This code takes some time to run

import numpy as np
import matplotlib.pyplot as plt

# Main code with Table 1 constants

############## INITIALIZATION OF CONSTANTS AND PARAMETERS ################
g = 9.81  # gravitational acceleration (m/s^2)
Pr = 240  # Reservoir pressure (bar)
Pb = 175  # Boiling point pressure (bar)
Lh = 1000  # Length of horizontal well section (m)
Lv = 2000  # Height of vertical section (m)
Dlh = 10  # Segment length horizontal section (m)
Dlv = 20  # Segment length vertical section (m)
K = 1.3e-3  # Perforation inflow constant
Rol = 800  # Oil density (kg/m^3)
Rog0 = 1.5  # Gas density at reference condition (kg/m^3)
Myl = 2e-3  # Liquid viscosity (dynamic Pa*s)
Nyg = 2e-5  # Gas viscosity (kinematic m^2/s)
Dhor = 0.15  # Diameter of horizontal section
Dver = 0.20  # Diameter of vertical section
Ahor = np.pi * Dhor**2 / 4  # Area of horizontal section
Aver = np.pi * Dver**2 / 4  # Area of vertical section
Perfor = np.array([0, 200, 400, 600, 800])  # Perforation locations (m from "toe")
Nper = len(Perfor)
Nhor = 1 + Lh // Dlh  # Number of cells in horizontal section
Nver = Lv // Dlv  # Number of cells in vertical section
PWHI = np.arange(60, 81, 1)  # Wellhead pressures to iterate over
S = 1 # Initial slip ratio

############## PREPROCESSING ###############

# Generating Lpos for horizontal and vertical sections combined
# Positions in the horizontal section
Lpos_hor = np.arange(0, Lh + Dlh, Dlh)

# Positions in the vertical section
Lpos_ver = np.arange(Lpos_hor[-1] + Dlv, Lpos_hor[-1] + Lv + Dlv, Dlv)

# Combine the two sections
Lpos = np.concatenate((Lpos_hor, Lpos_ver))

Kxi = np.zeros(Nhor)

# Set perforation flags in the Kxi array
for j in range(Nhor):
    Lx = j * Dlh  # Position in horizontal section starting from toe
    if Lx in Perfor:
        Kxi[j] = K  # Set inflow constant at perforations

############## MAIN ITERATION OVER WELLHEAD PRESSURES ################
MtotWH = np.zeros(len(PWHI))
NITER_list = []  # List to store the number of iterations for each Pwh

for ipwh, Pwh in enumerate(PWHI):
    PWSTART = [247]
    NITER = 1
    Relstore = [1]
    NITER_list =[1] # Allocate
    Pwi = np.full(Nhor + Nver, 247.0)  # Initial pressure in all cells
    Relerr = 1.0
    Errlimit = 1e-5
    Corec = 5

    while abs(Relerr) > Errlimit:
        Ugsis = [] # create empty list to store Ugsi values
        Ulsis = [] # create empty list to store Ulsi values
        Epsgis = [] # create empty list to store Epsgi values
        Umixed_list = [] # create empty list to store Ulsi + Ugsi values
        NITER = NITER + 1
        NITER_list.append(NITER)

  # Adjust first cell pressure based on error
        Pwi[0] = Pwi[0] - Corec * Relerr
        PWSTART.append(Pwi[0])

        Ulsi = np.zeros(Nhor + Lv // Dlv)
        Umix = np.zeros_like(Ulsi)
        Epsgi = np.zeros_like(Ulsi)
        Mgi = np.zeros_like(Ulsi)
        Mli = np.zeros_like(Ulsi)
        Ugsi = np.zeros_like(Ulsi)

        Mtot = 0  # Total mass inflow reset for each iteration
        for i in range(Nhor + Nver):
            # Area changes between horizontal and vertical sections
            Areal = Ahor if i < Nhor else Aver
```

```
# Diameter changes between horizontal and vertical sections
D = Dhor if i < Nhor else Dver
# Check for perforation in the horizontal section
if i < Nhor and Kxi[i] > 0:
    # Mass inflow through perforation
    Minn = Rol * K * (Pr - Pwi[i])
    Mtot += Minn  # Update total mass inflow

# Fluid properties and flow calculations
Rom = Rol  # Default density is liquid density
Mym = Myl  # Default viscosity is liquid viscosity
Ulsi = Mtot / (Rol * Areal)  # Liquid velocity
Umix = Ulsi  # Mixture velocity
Epsgi = 0  # Gas fraction initialized to 0
# Check for boiling
if Pwi[i] < Pb:
    Gfrak = 0.6 * (Pb - Pwi[i]) / (Pb - 1)  # Gas fraction
    Mgi = Gfrak * Mtot  # Gas mass flow
    Mli = Mtot - Mgi  # Liquid mass flow
    Rog = Pwi[i] * Rog0  # Gas density
    Ugsi = Mgi / (Rog * Areal)  # Gas velocity
    Ulsi = Mli / (Rol * Areal)  # Liquid velocity
    Umix = Ulsi + Ugsi  # Updated mixture velocity
    Epsnoslip  = Ugsi / Umix # Gas fraction without slip
    S  = 1 + 10*Epsnoslip # Slip calculation
    S_old = S # Update S_old to be the current S
    Epsgi = Ugsi / (Ugsi + S * Ulsi) # Gas fraction with slip
    Myg = Nyg * Rog  # Gas dynamic viscosity
    Rom = Rog * Epsgi + Rol * (1 - Epsgi)  # Mixture density
    Mym = Myg * Epsgi + Myl * (1 - Epsgi)  # Mixture viscosity
# END boiling test

Ugsis.append(Ugsi) # store gas velocity to list ugsis
Ulsis.append(Ulsi) # store liquid velocity to list ulsis
Epsgis.append(Epsgi) # store gas fraction with slip to list Epsgis
Umixed_list.append(Umix) # Ugsi + Ulsi stored in umix list.

############## PRESSURE DROP CALCULATIONS ##############
# Reynolds number
Reyn = Rom * Umix * D / Mym

# Friction factor
Frik = 0.046 * Reyn ** (-0.2) if Reyn >= 4000 else 64 / Reyn

# Pressure gradient due to friction
Dpdxf = (4 / D) * Frik * 0.5 * Rom * Umix ** 2
if i >= Nhor:  # Include hydrostatic pressure gradient in vertical section
    Dpdxf += Rom * g

# Update pressure for next cell
if i < Nhor + Nver - 1:
    Pwi[i + 1] = Pwi[i] - Dpdxf * (Dlh if i < Nhor else Dlv) * 1e-5
```

```
        # Relative error = calculated - actual wellhead pressure
        Relerr = 0.8 * (Pwi[Nhor + Nver - 1] - Pwh) / Pwh
        print(f"Iteration {NITER}: Pwh = {Pwh}")
        Relstore.append(Relerr)
        # END loop from toe to top in well

        ############# VISUALIZATION #############

        # Plots are placed here to plot for every iteration due to being inside the loop
        trykk=str(Pwh)
        iterasjon=str(NITER)
        strengliste = 'Pwh = {}, Iterasjon nr = {}'.format(trykk, iterasjon)

        plt.figure(figsize=(12, 8))
        plt.subplot(3, 1, 1)
        plt.plot(Lpos, Pwi, label='Well Pressure')
        plt.xlabel('Position along well (m)')
        plt.ylabel('Well pressure (bar)')
        plt.legend()
        # This will place the text in the middle of the subplot
        plt.text(0.1, 0.5, strengliste, transform=plt.gca().transAxes)

        plt.subplot(3, 1, 2)
        plt.plot(Lpos, Epsgis, label='Gas Fraction')
        plt.xlabel('Position along well (m)')
        plt.ylabel('Gas fraction')
        plt.legend()

        plt.subplot(3, 1, 3)
        plt.plot(Lpos, Umixed_list, label='Mixture Velocity')
        plt.xlabel('Position along well (m)')
        plt.ylabel('Mixture velocity (m/s)')
        plt.legend()
        plt.tight_layout()

    #END loop over iteration

    MtotWH[ipwh] = Mtot  # Store total mass inflow for current wellhead pressure

    plt.figure(figsize=(12, 4))
    plt.subplot(1, 2, 1)
    plt.plot(NITER_list, PWSTART, label='Initial Pressure at Toe')
    plt.xlabel('Iteration number')
    plt.ylabel('Bottomhole (toe) well pressure (bar)')
    # This will place the text in the middle of the subplot
    plt.text(0.5, 0.5, strengliste, transform=plt.gca().transAxes)
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(NITER_list, Relstore, label='Relative Error')
    plt.xlabel('Iteration number')
    plt.ylabel('Relative error')
    plt.legend()
    plt.tight_layout()
```

```
        print(f"*************** DONE WELLHEAD PRESSURE  Pwh = {Pwh} bar  **********")

        Relstore.clear()
        PWSTART.clear()
        NITER_list.clear()

    #END loop over Wellhead pressure

    # Plotting Results
    plt.figure(figsize=(6, 4))
    plt.plot(PWHI, MtotWH, '-o', label='Production vs. Wellhead Pressure')
    plt.xlabel('Wellhead pressure (bar)')
    plt.ylabel('Reservoir production (kg/s)')
    plt.title('Reservoir Production vs Wellhead Pressure')
    plt.legend()
    plt.show()
```

B Supplementary code: Boiling pressures

The following Python script has been modified from main code to iterate over different boiling pressures, analyzing their impact on reservoir production. The script is included to demonstrate the changes made to the simulation model and to provide the complete code for reference.

```
# Purpose of this script is to plot boiling pressures vs production
# This code takes some time to run, print statement is made as "loading" screen
# Constants are the same as Table 3

import numpy as np
import matplotlib.pyplot as plt

############## INITIALIZATION OF CONSTANTS AND PARAMETERS ################
g = 9.81  # gravitational acceleration (m/s^2)
Pr = 240  # Reservoir pressure (bar)
Pwh_values = np.arange(30, 85, 5)   # Wellhead pressures
Lh = 1000  # Length of horizontal well section (m)
Lv = 2000  # Height of vertical section (m)
Dlh = 10  # Segment length horizontal section (m)
Dlv = 1  # Segment length vertical section (m)
K = 1.3e-3  # Perforation inflow constant
Rol = 800  # Oil density (kg/m^3)
Rog0 = 1.5  # Gas density at reference condition (kg/m^3)
Myl = 2e-3  # Liquid viscosity (dynamic Pa*s)
Nyg = 2e-5  # Gas viscosity (kinematic m^2/s)
Dhor = 0.15  # Diameter of horizontal section
Dver = 0.20  # Diameter of vertical section
Ahor = np.pi * Dhor**2 / 4  # Cross-sectional area of horizontal section
Aver = np.pi * Dver**2 / 4  # Cross-sectional area of vertical section
```

```
Perfor = np.array([0, 200, 400, 600, 800])  # Perforation locations (m from "toe")
Nper = len(Perfor)
Nhor = 1 + Lh // Dlh  # Number of cells in horizontal section
Nver = Lv // Dlv  # Number of cells in vertical section
PB = np.arange(30, 270, 10)  # Boiling pressures to iterate over
S = 1 # Initial slip ratio

##############  INITIALIZATION OF CONSTANTS AND PARAMETERS ################
# Generating Lpos for horizontal and vertical sections combined
Lpos_hor = np.arange(0, Lh + Dlh, Dlh)  # Positions in the horizontal section
Lpos_ver = np.arange(Lpos_hor[-1] + Dlv, Lpos_hor[-1] + Lv + Dlv, Dlv)
# Positions in the vertical section

# Combine the two sections
Lpos = np.concatenate((Lpos_hor, Lpos_ver))

Kxi = np.zeros(Nhor)

# Set perforation flags in the Kxi array
for j in range(Nhor):
    Lx = j * Dlh  # Position in horizontal section starting from toe
    if Lx in Perfor:
        Kxi[j] = K  # Set inflow constant at perforations

############## MAIN ITERATION OVER BOILING PRESSURES ################
MtotWH = np.zeros(len(PB))
NITER_list = []  # List to store the number of iterations for each Pb

# This is a added loop in order to iterate over different wellhead presures
for Pwh in Pwh_values:
    # Re-initialize variables that depend on Pwh for each iteration
    Pwi = np.full(Nhor + Nver, Pr)  # Reset initial pressure for each Pwh
    MtotWH = np.zeros(len(PB))  # Reset total mass inflow array for new Pwh

# Loop over boiling pressures instead of wellhead pressures.
    for ipwh, Pb in enumerate(PB):
        PWSTART = [247]
        NITER = 1
        Relstore = [1]
        NITER_list =[1] # Allocate
        Pwi = np.full(Nhor + Nver, 247.0)  # Initial pressure in all cells
        Relerr = 1.0
        Errlimit = 1e-5
        Corec = 5
        print(f"Current Pwh: {Pwh}, Current Pb: {Pb}")
```

```
while abs(Relerr) > Errlimit:
    Ugsis = [] # create empty list to store Ugsi values
    Ulsis = [] # create empty list to store Ulsi values
    Epsgis = [] # create empty list to store Epsgi values
    Umixed_list = [] # create empty list to store Ulsi + Ugsi values
    NITER = NITER + 1
    NITER_list.append(NITER)

    # Adjust first cell pressure based on error
    Pwi[0] = Pwi[0] - Corec * Relerr
    PWSTART.append(Pwi[0])

    Ulsi = np.zeros(Nhor + Lv // Dlv)
    Umix = np.zeros_like(Ulsi)
    Epsgi = np.zeros_like(Ulsi)
    Mgi = np.zeros_like(Ulsi)
    Mli = np.zeros_like(Ulsi)
    Ugsi = np.zeros_like(Ulsi)

    Mtot = 0  # Total mass inflow reset for each iteration
    for i in range(Nhor + Nver):
    # Area changes between horizontal and vertical sections
        Areal = Ahor if i < Nhor else Aver
    # Diameter changes between horizontal and vertical sections
        D = Dhor if i < Nhor else Dver
    # Check for perforation in the horizontal section
        if i < Nhor and Kxi[i] > 0:
        # Mass inflow through perforation
            Minn = Rol * K * (Pr - Pwi[i])
            Mtot += Minn  # Update total mass inflow

        # Fluid properties and flow calculations
        Rom = Rol  # Default density is liquid density
        Mym = Myl  # Default viscosity is liquid viscosity
        Ulsi = Mtot / (Rol * Areal)  # Liquid velocity
        Umix = Ulsi  # Mixture velocity
        Epsgi = 0  # Gas fraction initialized to 0
        # Check for boiling
        if Pwi[i] < Pb:
            Gfrak = 0.6 * (Pb - Pwi[i]) / (Pb - 1)  # Gas fraction
            Mgi = Gfrak * Mtot  # Gas mass flow
            Mli = Mtot - Mgi  # Liquid mass flow
            Rog = Pwi[i] * Rog0  # Gas density
            Ugsi = Mgi / (Rog * Areal)  # Gas velocity
            Ulsi = Mli / (Rol * Areal)  # Liquid velocity
            Umix = Ulsi + Ugsi  # Updated mixture velocity
            Epsnoslip  = Ugsi / Umix # Gas fraction without slip
```

```
                    S  = 1 + 10*Epsnoslip # Slip calculation
                    S_old = S # Update S_old to be the current S
                    Epsgi = Ugsi / (Ugsi + S * Ulsi) # Gas fraction with slip
                    Myg = Nyg * Rog  # Gas dynamic viscosity
                    Rom = Rog * Epsgi + Rol * (1 - Epsgi)  # Mixture density
                    Mym = Myg * Epsgi + Myl * (1 - Epsgi)  # Mixture viscosity
                # END boiling test

                Ugsis.append(Ugsi) # store gas velocity
                Ulsis.append(Ulsi) # store liquid velocity
                Epsgis.append(Epsgi) # store gas fraction with slip
                Umixed_list.append(Umix) # Ugsi + Ulsi stored in umix list

                ############## PRESSURE DROP CALCULATIONS ##############
                # Reynolds number
                Reyn = Rom * Umix * D / Mym

                # Friction factor
                Frik = 0.046 * Reyn ** (-0.2) if Reyn >= 4000 else 64 / Reyn

                # Pressure gradient due to friction
                Dpdxf = (4 / D) * Frik * 0.5 * Rom * Umix ** 2

                # Include hydrostatic pressure gradient in vertical section
                if i >= Nhor:
                    Dpdxf += Rom * g

                # Update pressure for next cell
                if i < Nhor + Nver - 1:
                    Pwi[i + 1] = Pwi[i] - Dpdxf * (Dlh if i < Nhor else Dlv)*1e-5


            # Update relative = calculated - actual wellhead pressure
            Relerr = 0.8 * (Pwi[Nhor + Nver - 1] - Pwh) / Pwh
            Relstore.append(Relerr)

            # Store total mass inflow for current boiling pressure
            MtotWH[ipwh] = Mtot
        # END loop from toe to top in well
        # Labeling the boiling pressure
    plt.plot(PB, MtotWH, '-o', label=f'Pwh = {Pwh}')


# Plotting Results
plt.xlabel('Boiling pressure (bar)')
plt.ylabel('Reservoir production (kg/s)')
plt.title('Reservoir Production vs Boiling pressure')
```

```
# Making the legend box small so it does not interupt with the plot
plt.legend(fontsize='small', loc='lower right', ncol=2)
plt.show()
```

C Supplementary code: Total pressure gradient

The python code under is used to visualize the relationship of how increasing gas velocity changes total pressure gradient by assuming a constant liquid velocity.

```
import numpy as np
import matplotlib.pyplot as plt

# Constants under should be the same as Table 3

# Constants
g = 9.81  # gravitational acceleration (m/s^2)
Dver = 0.20  # Diameter of vertical section (m)
Rho_o = 800  # Oil density (kg/m^3)
Rho_g0 = 1.5  # Gas density at reference condition (kg/m^3)
Myl = 2e-3  # Liquid viscosity (dynamic Pa*s)
Nyg = 2e-5  # Gas viscosity (kinematic m^2/s)
Pr = 240  # Reservoir pressure (bar)

# Convert reservoir pressure to Pascals
Pr_pascal = Pr * 1e5  # 1 bar = 1e5 pascals

# Cross-sectional area of vertical section
Aver = np.pi * Dver**2 / 4

# Initialize the Uls and Ugs ranges
Ulsi = 1.0  # Constant liquid velocity (m/s)
Ugsi_values = np.linspace(0, 50, 100)  # Varying gas velocities from 0 to 50 m/s

# Store results
friction_gradients = []
hydrostatic_gradients = []
total_gradients = []

for Ugsi in Ugsi_values:
# Calculate no-slip gas fraction
    Gas_fraction = Ugsi / (Ulsi + Ugsi)

# Calculate mixture density with no-slip gas fraction
    Rom = Rho_o * (1 - Gas_fraction) + Rho_g0  * Gas_fraction

# Mixture viscosity
    Mym = Nyg * Rho_g0  * Gas_fraction + Myl * (1 - Gas_fraction)
```

# Reynolds number
    Reyn = Rom * (Ulsi + Ugsi) * Dver / Mym

# Friction factor calculation
    if Reyn >= 4000:
        Frik = 0.046 * Reyn ** (-0.2) # Turbulent flow friction factor
    else:
        Frik = 64 / Reyn # Laminar flow friction factor

    # Friction pressure gradient
    Dpdxf = (4 / Dver) * Frik * 0.5 * Rom * (Ulsi + Ugsi) ** 2

    # Hydrostatic pressure gradient
    DpdxHyd = Rom * g

    # Store results
    friction_gradients.append(Dpdxf)
    hydrostatic_gradients.append(DpdxHyd)
    total_gradients.append(Dpdxf + DpdxHyd)

# Plotting the results on a single plot
plt.figure(figsize=(10, 8))
plt.plot(Ugsi_values, friction_gradients, label='Dpdxf (Friction Gradient)')
plt.plot(Ugsi_values, hydrostatic_gradients, label='DpdzHyd (Hydrostatic Gradient)')
plt.plot(Ugsi_values, total_gradients, label='DpdzTotal (Total Pressure Gradient)')

plt.xlabel('Gas Velocity Ugs (m/s)')
plt.ylabel('Pressure Gradient (Pa/m)')
plt.title('Pressure Gradients vs Gas Velocity for constant Uls = 1 m/s')
# Set x-axis ticks at intervals of 5
plt.xticks(np.arange(min(Ugsi_values), max(Ugsi_values) + 1, 5))
plt.ylim(0, 5000, 500) # Set y-axis to range from 0 to 5000
plt.yticks(np.arange(0, 5001, 500)) # Set y-axis ticks with steps of 500
plt.legend()
plt.show()

D MATLAB code

This code, provided by the supervisor, has been translated into the script outlined in Appendix A, thus it should produce identical results.

```
% ******************************************************************
% Calculate flow in a combined horisontal and vertical well with
% perforations.
% ASSUMPTIONS:
% - Homogeneous (noslip) model used - turbulent friction factor
% - Oil (single phase) reservoir
% - Isothermal flow conditions
% - Only momentum equation important for pressure drop
% - "Black oil" model for gas fraction calculation
% - Ideal gas
%
% *******************************************************************
% Tried to sum hydrostatic and frictional pressure drop and make these
% separate contributions to the total pressure drop. This is wrong.
% Insted friction should be the difference between Pwh and Pr muns the


clc
clear
format compact
starttid = tic
% *******************************************************************
% Data list:
g = 9.81              % gravitational acceleration (m/s^2)
Pr = 240              % Reservoir pressure (bar)
Pb = 175; 175          % Boiling point pressure (bar)
Pwh = 30              % Wellhead pressure (see PWHI)
Lh = 1000             % Length of horisontal well section (m)
Lv = 2000             % Height of vertical section (m)
Dlh = 10              % Segment length horisontal section (m)
Dlv = 20             % Segment length vertical section (m)
K = 1.3e-3            % Perforation inflow constant
%K = 5e-3             % Perforation inflow constant
Rol = 800             % Oil density (kg/m^)
Rog0 = 1.5            % Gas density at refernce condition (kg/m^)
Myl = 2e-3            % Liquid viscosity (dynamic Pa*s)
Nyg = 2e-5            % Gas viscosity (kinematic m^2/s)
Dhor = 0.15           % Diameter of horisontal section
Dver = 0.20           % Diameter of vertical section
Ahor = pi*Dhor^2/4    % Cross sectional area of horisontal section
Aver = pi*Dver^2/4    % Cross sectional area of vertical section

% ****************************************************************
% Pre analysis!
% Determine which cells in the horisontal section contain perforations
% ****************************************************************
Perfor = [0 200 400 600 800];  % perforation locations (m from "toe")
Nper = max(size(Perfor));
Nhor = 1+Lh/Dlh;             % Number of cells in horisontal section

Lpos(1)=0;
for j=2:Nhor+1               % Fill horisontal part of position vector
```

```
Lpos(j) = Lpos(j-1)+Dlh ;
disp(Lpos(j))
end

Kxi = zeros(100);
for j=1:Nhor              % Test every cell in horisontal well
Lx = (j-1)*Dlh;       % Position in horisontal section starting from toe
Lxi(j)=Lx;
for iper=1:Nper
if Lpos(j) == Perfor(iper)
Kxi(j)= K;     % Set perforation "flag": here inflow constant.
               % Could in principle be different for each perf.
end
end
end
% ***************************************

% *********** New extra loop over Boiling point pressure ***********
PBoil = 175:15:175% :110;
PBoilSize = length(PBoil);
for ipBP = 1:PBoilSize
Pb = PBoil(ipBP);

%  New loop over Pwh (use this)
PWHI = 60:1:80;
ipSize=length(PWHI)
for ipwh=1:ipSize % 11 - var med 30:40
Pwh=PWHI(ipwh);

Relerr=1;
Errlimit = 1e-5;
Pwi(1)= Pr-0.01; ;248;
NITER = 1;

PWSTART(NITER)=Pwi(1);
Corec=5;

while abs(Relerr) > Errlimit

NITER = NITER+1;

Pwi(1) = Pwi(1) - Corec*Relerr;    % Pressure in first cell (Assumption!)
disp(['Iteration ', num2str(NITER), ': Pwi(1) = ', num2str(Pwi(1)), ...
     ', Relerr = ', num2str(Relerr)]);
% disp(['Index ', num2str(1), ': Pwi(1) = ', num2str(Pwi(1))]);
PWSTART(NITER)=Pwi(1);
```

```
% ****************************************
% Run through horisontal well section
% Set total mass inflow initially to zero
% ****************************************
Ipos = 0;
Mtot = 0;
Areal = Ahor;
%            Sumfrik=0;
%            Sumhydro=0;
for i=1:Nhor
if Kxi(i)>0                % Test if perforation flag is set
    Minn=Rol*K*(Pr-Pwi(i)); % Mass  inflow through perforation
    % Use perhaps instead square root of DP as in nozzle flow
    Mtot=Mtot+Minn;          % Add up mass inflow from each perforation
    % disp(['Index ', num2str(i), ': Pwi(i) = ', num2str(Pwi(i))]);
end

% ************************************************
% Old "Finngass" SUB from Well98 added inside loop:
% ************************************************
Rom=Rol;
Mym=Myl;
Ulsi(i)=Mtot/(Rol*Areal);
Umix=Ulsi(i);
Epsgi(i)=0;
if Pwi(i)<Pb  % Detect if boiling takes place
          % Use linear dependence of gas fraction on
          % deviation from bubble point pressure
    Gfrak=.6*(Pb-Pwi(i))/(Pb-1);
    Mgi(i)=Gfrak*Mtot;
    Mli(i)=Mtot-Mgi(i);
    Rog=Pwi(i)*Rog0;
    Ugsi(i)=Mgi(i)/(Rog*Areal);
    Ulsi(i)=Mli(i)/(Rol*Areal);
    Umix=Ulsi(i)+Ugsi(i);
    % New gas fraction calculation, with slip included *****
    % Slip ratio increses with gas fraction
    Umix = Ulsi(i)+Ugsi(i);
    Epsnoslip=Ugsi(i)/Umix;
    S = 1+10*Epsnoslip;
    Epsgi(i)=Ugsi(i)/(Ugsi(i)+ S*Ulsi(i));
    % End new gas fractrion calculation ******************
    Myg=Nyg*Rog;
    Rom=Rog*Epsgi(i)+Rol*(1-Epsgi(i));
    Mym=Myg*Epsgi(i)+Myl*(1-Epsgi(i));
end
```

```
Reyn=Rom*Umix*Dhor/Mym;
if Reyn < 4000
    disp('Laminar flow')
end
Frik=.046*Reyn^(-.2);
Dpdxf=(4/Dhor)*Frik*.5*Rom*Umix^2;
%               Sumfrik=Sumfrik+Dpdxf;
Pwi(i+1)=Pwi(i)-Dpdxf*Dlh*1e-5;  %Calculate pressure in bar
end



% ****************************************
% Run through vertical well section
% Set total mass inflow initially to zero
% ****************************************
Areal = Aver;     % Cross sectional area in vertical well
Nver = Lv/Dlv;  % Number of cells in vertical section


for j=Nhor+1:Nhor + Nver % Fill vertical part of position vector
Lpos(j) = Lpos(j-1)+Dlv  ;
end

for i = Nhor+1:Nhor + Nver
Rom=Rol;
Mym=Myl;
Ulsi(i)=Mtot/(Rol*Areal);
Umix=Ulsi(i);
Epsgi(i)=0;
Ugsi(i)=0;
if Pwi(i)<Pb  % Detect if boiling takes place
              % Use linear dependence of gas fraction on
              % deviation from bubble point pressure
    Gfrak=.6*(Pb-Pwi(i))/(Pb-1);
    Mgi(i) = Gfrak*Mtot;
    Mli(i) = Mtot-Mgi(i);
    Rog = Pwi(i)*Rog0;
    Ugsi(i) = Mgi(i)/(Rog*Areal);
    Ulsi(i) = Mli(i)/(Rol*Areal);
    % New gas fraction calculation, with slip included *****
    % Slip ratio increses with gas fraction
    Umix = Ulsi(i)+Ugsi(i);
    Epsnoslip=Ugsi(i)/Umix;
    S = 1+10*Epsnoslip;
    Epsgi(i)=Ugsi(i)/(Ugsi(i)+ S*Ulsi(i));
    % End new gas fraction calculation ******************
    Myg = Nyg*Rog;
    Rom = Rog*Epsgi(i)+Rol*(1-Epsgi(i));
    Mym = Myg*Epsgi(i)+Myl*(1-Epsgi(i));
end
Reyn = Rom*Umix*Dver/Mym;
Frik = 0.046*Reyn^(-.2);
Dpdyf = (4/Dver)*Frik*.5*Rom*Umix^2;
%               Sumfrik=Sumfrik+Dpdyf;
Dpdyh = Rom*g;
Dpdy = Dpdyf + Dpdyh;
%               Sumhydro=Sumhydro+Dpdy;
Pwi(i+1) = Pwi(i)-Dpdy*Dlv*1e-5; %Calculate pressure in bar
end




Ntot=i;    % index of last cell pressure
Pwi = Pwi(1:Ntot); % Remove nonexisting cell
Relerr= 0.8*(Pwi(Ntot)-Pwh)/Pwh;
disp(['  Phw:  ',num2str(Pwh),'  Outlet pressure:  ',num2str(Pwi(Ntot))]);

figure(1)
subplot(3,1,1); plot(Lpos,Pwi)
xlabel(' Position along well (m)')
ylabel(' Well pressure (bar)')
text(100, 150, ['Pwh = ' num2str(Pwh) ' bar.  Pb = ' num2str(Pb) ' bar' ]);
% newline ...
%'Frictional pressure drop: '
% num2str(Sumfrik*1e-5) 'Hydrostatic: ' num2str(Sumhydro*1e-5) ])

subplot(3,1,2); plot(Lpos,Epsgi)
xlabel(' Position along well (m)')
ylabel(' Gas fraction ')

subplot(3,1,3); plot(Lpos, Ugsi + Ulsi)
xlabel(' Position along well (m)')
ylabel(' Mixture velocity (m/s) ')

NITER;
Relstore(NITER)=Relerr;
end % end of while loop

figure(2)
subplot(2,1,1);
plot(PWSTART)
```

```
xlabel(' Iteration number')
ylabel(' Bottomhole (toe) well pressure (bar)')
text(40, 240, ['Pwh = ' num2str(Pwh) ' bar'])

subplot(2,1,2);
plot(Relstore)
xlabel(' Iteration number')
ylabel(' Relative error')
text(40, 1, ['Pwh = ' num2str(Pwh) ' bar'])
MtotWH(ipwh)=Mtot;
AntIter(ipwh)=NITER;
%       SumFriksjon(ipwh)= Sumfrik*1e-5;
%       SumHydrostat(ipwh)=Sumhydro*1e-5;
%       Sumtrykk(ipwh) = (Sumfrik+Sumhydro)*1e-5;
disp(['Wellhead Pressure: ', num2str(Pwh), ' bar, Total Mass Flow Rate: ', ...
num2str(Mtot), ' kg/s']);
disp('Trykk en tast for aa fortsette til neste Pwh.')
disp('Trykk Ctrl-C inni Command window for avbryte og starte ny.')
% pause
end % End Pwh loop

figure(10)
plot(PWHI,MtotWH, '-o')
xlabel('Well head pressure (bar)')
ylabel('Reservoir production Oil (kg/s)')
title('Reservoir production vs Wellhead pressure')

figure(11)
plot(PWHI, AntIter, '-o')
xlabel('Well head pressure (bar)')
ylabel('Number of iterations required')
title('Number of iterations required - all Phw')

figure(12)
plot(PWHI,MtotWH, '-o')
xlabel('Well head pressure (bar)')
ylabel('Reservoir production Oil (kg/s)')
title('Reservoir production vs Wellhead pressure')
hold on

MtotPBWH(ipBP,:) = MtotWH; %Keep all production profiles for all Pb
end %PB  indeks ipBP
figure(14)
for i=1:ipBP
hold on
plot(PWHI,MtotPBWH(i,:), '-o')
end
```

```
% legend(hleg)%Denne er enkel, men primitiv - skriver bar 'data 1 data2 ...')
% legend(strcat('Pb = ',num2str(Pb')))
legend(strcat('Pb = ',num2str(PBoil')))
xlabel('Well head pressure (bar)')
ylabel('Reservoir production Oil (kg/s)')
title('Reservoir production vs Wellhead pressure - and Boiling point pressures')


Totaltid = toc(starttid)
```

7 References

References

[1] A. Michael, A. Srivastava, L. Garg, and R. Harkouss. "End-to-end digital transformation of petroleum engineering: The evolution of a revolution," TWA. (Jul. 19, 2020), [Online]. Available: https://jpt.spe.org/twa/end-end-digital-transformation-petroleum-engineering-evolution-revolution (visited on 03/20/2024).

[2] Deloitte. "Digital transformation in oil and gas." (), [Online]. Available: https://www.deloitte.com/global/en/Industries/energy/perspectives/digital-transformation-in-oil-and-gas.html (visited on 03/20/2024).

[3] S. Choi, K. Eloot, D. Lee, S. Liu, and K. on Laufenberg. "Building sustainability in manufacturing operations," McKinsey. (), [Online]. Available: https://www.mckinsey.com/capabilities/operations/our-insights/building-sustainability-into-operations (visited on 03/20/2024).

[4] Python. "Our community," Python.org. (), [Online]. Available: https://www.python.org/community/ (visited on 03/20/2024).

[5] R. Scarlett. "Why python keeps growing, explained," The GitHub Blog. (Mar. 2, 2023), [Online]. Available: https://github.blog/2023-03-02-why-python-keeps-growing-explained/ (visited on 03/20/2024).

[6] NIST, "SI units," Apr. 12, 2010, Last Modified: 2024-01-16T15:10-05:00. [Online]. Available: https://www.nist.gov/pml/owm/metric-si/si-units (visited on 03/21/2024).

[7] lumenlearning. "Density," Physics. (), [Online]. Available: https://courses.lumenlearning.com/suny-physics/chapter/11-2-density/ (visited on 03/21/2024).

[8] Y. Kassem. "Perforating oil and gas wells," AONG website. (Jun. 10, 2020), [Online]. Available: https://www.arab-oil-naturalgas.com/perforating-oil-and-gas-wells/ (visited on 05/05/2024).

[9] R. Time, *Two-phase flow flow in pipelines*, Jan. 8, 2017. (visited on 03/23/2024).

[10] Princeton University. “DEFINITION OF VISCOSITY.” (), [Online]. Available: https://www.princeton.edu/~gasdyn/Research/T-C_Research_Folder/Viscosity_def.html (visited on 03/21/2024).

[11] H. Levi. “What is the difference between dynamic and kinematic viscosity?” (), [Online]. Available: https://www.cscscientific.com/csc-scientific-blog/whats-the-difference-between-dynamic-and-kinematic-viscosity (visited on 03/21/2024).

[12] SimScale. “What is reynolds number?” SimScale. (Nov. 8, 2023), [Online]. Available: https://www.simscale.com/docs/simwiki/numerics-background/what-is-the-reynolds-number/ (visited on 04/13/2024).

[13] F. Elizalde. “Laminar flow vs. turbulent flow - what is laminar flow?” Cleatech Scientific. (Oct. 26, 2021), [Online]. Available: https://www.cleatech.com/laminar-flow-vs-turbulent-flow/ (visited on 03/23/2024).

[14] J. M. Mandhane, G. A. Gregory, and K. Aziz, “A flow pattern map for gas—liquid flow in horizontal pipes,” *International Journal of Multiphase Flow*, vol. 1, no. 4, pp. 537–553, 1974, ISSN: 0301-9322. DOI: https://doi.org/10.1016/0301-9322(74)90006-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0301932274900068.

[15] Y. Taitel and A. E. Dukler, “A model for predicting flow regime transitions in horizontal and near horizontal gas-liquid flow,” *AIChE Journal*, vol. 22, no. 1, pp. 47–55, 1976, _eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690220105. DOI: https://doi.org/10.1002/aic.690220105. [Online]. Available: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690220105.

[16] Y. Taitel, D. Barnea, and A. E. Dukler, “Modelling flow pattern transitions for steady upward gas-liquid flow in vertical tubes,” *AIChE Journal*, vol. 26, no. 3, pp. 345–354, 1980, _eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690260304. DOI: https://doi.org/10.1002/aic.690260304. [Online]. Available: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690260304.

[17] R. Time, *Personal communication about suggested slip ratio used in python*, Mar. 15, 2024.

[18] R. Time, *Personal communication about slip ratio suggestion for various flow regimes*, Mar. 10, 2024.

[19] H.-S. Dou, “Secret hidden in navier-stokes equations: Singularity and criterion of turbulent transition,” Aug. 30, 2014.

[20] D. Brkić, “Solution of the implicit colebrook equation for flow friction using excel,” *Spreadsheets in Education*, vol. 10, no. 2, Aug. 2017, Publisher: Bond University. [Online]. Available: https://hal.science/hal-01586190 (visited on 05/10/2024).

[21] python. “What is python? executive summary,” Python.org. (), [Online]. Available: https://www.python.org/doc/essays/blurb/ (visited on 04/01/2024).

[22] Simplilearn. “MATLAB vs python: Ultimate showdown of programming titans [2024],” Simplilearn.com. (Feb. 1, 2024), [Online]. Available: https://www.simplilearn.com/matlab-vs-python-article (visited on 04/01/2024).

[23] mathworks. "MATLAB vs. python: Which one is right for you?" (), [Online]. Available: `https://se.mathworks.com/products/matlab/matlab-vs-python.html` (visited on 04/01/2024).

[24] UiS. "Introduction to programming (DAT120)." (), [Online]. Available: `https://www.uis.no/en/course/DAT120_1` (visited on 04/01/2024).

[25] opautoclicker. "Automate mouse clicks." (), [Online]. Available: `https://www.opautoclicker.com/` (visited on 04/13/2024).

[26] Python. "The python standard library," Python documentation. (), [Online]. Available: `https://docs.python.org/3/library/index.html` (visited on 04/02/2024).

[27] NumPy. "NumPy: The absolute basics for beginners." (), [Online]. Available: `https://numpy.org/doc/stable/user/absolute_beginners.html` (visited on 04/02/2024).

[28] matplotlib. "Pyplot tutorial." (), [Online]. Available: `https://matplotlib.org/stable/tutorials/pyplot.html` (visited on 04/02/2024).

[29] T. Adeyemi, "A NOVEL APPROACH TO FORECASTING PRODUCTION RATE OF DRY GAS WELLS," *International Journal of Innovations in Engineering and Technology*, vol. 6, pp. 1–6, Mar. 1, 2021.