AN INTEGRATED HORIZONTAL AND VERTICAL FLOW SIMULATION WITH APPLICATION TO WAX PRECIPITATION

SAMEENA TRINA

AN INTEGRATED HORIZONTAL AND VERTICAL FLOW SIMULATION WITH APPLICATION TO WAX PRECIPITATION

By
Sameena Trina

A thesis submitted to the School of Graduate Studies in partial fulfilment of the requirements for the degree of Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland
St. John's, Newfoundland

December 2010

ABSTRACT

This research explored a few opportunities of improving the simulations available to reservoir engineers in the oil and gas industry. Three very specific simulation models were used in this thesis. Firstly, improvements were made to an inflow model for a horizontal well by making it possible to run the model for different fluids easily. Secondly, a vertical flow model was developed by combining a well-known, multi-phase flow correlation with a multi-phase temperature model. A novel approach was developed to solve these two models in sequence. Thirdly, this thesis scoped out the application of two different wax crystallization models. It was the first time that these wax models were tested using a flow simulator. The results obtained from all three simulation models were in par with theory and expectations. It was concluded that these models together would be a very useful tool for both the industry and for further research work.

ACKNOWLEDGEMENTS

It is a pleasure to thank those who made this thesis possible. First and foremost, I would like convey the most heartfelt thanks to my research supervisor Dr. T. Johansen. His enthusiasm, knowledge and guidance played the most vital role from the start to the finish of this project. Many thanks to both Dr. Johansen and the School of Graduate Studies for funding my research work. I am endlessly grateful to my colleagues Marjan Hashem and Justin Skinner for their support and encouragement every step of the way. The two years of toiling at the lab would have felt much longer and dreadful without them. I am thoroughly indebted to my father for introducing me to beautiful St. John's and opening the doors for me to pursue my M.Eng. His support and encouragement has always been my source of inspiration. Words cannot convey how incredibly thankful I am to my mother for always providing me with a solid blanket of affection for my well-being. I am honoured to have a sister and cousins who have always been there for me. I am also thankful to my numerous friends for putting up with my hyper activity and for keeping me entertained. Without their intervention, my Masters' years would have been devoid of adventure and high spirits. I am also ever grateful to my friends all over the world for their constant encouragement. I owe a lot to the many family friends in St. John's for being there for me as a powerful support network. Last but not the least, I am thankful to God for inundating me with countless blessings and much happiness during my tenure as a Master's student.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Chapter 3-2

K	Rock permeability
k_{rg}	Relative permeability of gas
k_{ro}	Relative permeability of oil
L	Reservoir length
PI	Productivity Index
$p_{annulus}$	Pressure in the annulus
$p_{reservoir}$	Pressure in the reservoir
$q_{\inf low}$	Fluid inflow rate
r_e	Radius of reservoir external boundary
r_{well}	Radius of wellbore
S	Skin Factor
S_o	Oil Saturation
S_g	Gas Saturation
μ_g	Gas viscosity
μ_o	Oil viscosity

Chapter 3-3

B_g	Gas formation volume factor
B_o	Oil formation volume factor
m_{in}	Mass of fluid coming into a node
m_{out}	Mass of fluid going out of a node
q_{in}	Fluid flow rate coming into a node
q_{out}	Fluid flow rate going out of a node
R_s	Solution gas-oil ratio
α_{in}	Liquid fraction in the fluid coming into a node
α_{out}	Liquid fraction in the fluid going out of a node
ρ_o^{RC}	Density of oil at reservoir conditions

Chapter 3-4

A	Cross-sectional area of pipe
c	Nozzle coefficient
D_h	Hydraulic diameter of pipe
f	Friction factor
L	Length of pipe
$\dot{m}$	Mass flow rate
p_{in}	Pressure at the inlet side of nozzle
p_{out}	Pressure at the outlet side of nozzle
$\frac{dp}{dz}$	Pressure drop over the length of pipe
q	Fluid flow rate
v	Fluid velocity
$\frac{\delta v}{\delta z}$	Velocity drop over the length of pipe
ρ	Fluid density
$\overline{\rho}$	Average fluid density
τ_w	Stress tensor at pipe wall

Chapter 3-5

F	Non-liner function
$\vec{F}$	Matrix of non-linear functions
J	Jacobian matrix
n	Number of functions, number of variables
x	Variables
$\vec{x}$	Matrix of variables
y	Parameter
ε	Threshold level

Chapter 4-1

A_t	Tubing cross-sectional area [ft^2]
BB	Gas velocity ratio parameter
B_g	Gas formation volume factor [unit less]
B_o	Oil formation volume factor [unit less]
B_w	Water formation volume factor [unit less]

CN_L	Coefficient of viscosity number
D	Tubing diameter [in]
dh	Differential change in well depth [ft]
dp	Differential change in well pressure [psi]
dW_e	Differential change in external work done by flowing fluid [$lb_f ft/lb_f$]
dW_f	Differential change in work done due to friction [$lb_f ft/lb_f$]
f	Friction factor
g	Gravitational constant [ft/sec^2]
g_c	Unit conversion factor [32.2 $lb_m ft/lb_f/sec^2$]
GLR	Gas liquid ratio [ft^3/bbl]
H_L	Liquid hold up [unit less]
L_B	Limits of boundary line for bubble flow [dimensionless less]
M	Mass associated with 1 bbl of fluid (includes water, oil, gas) [lb_m/bbl]
N_D	Diameter number
N_{GV}	Gas velocity number
N_{LV}	Liquid velocity number
N_L	Viscosity number
$\overline{p}$	Average pressure between point 1 and 2 [psi]
p_{r1}	Critical pressure at point 1
q_L	Mass flow rate of liquid (oil, water) [lb_m/day]
q_o	Mass flow rate of oil [lb_m/day]
q_w	Mass flow rate of water [lb_m/day]
Re_{TP}	Two phase Reynold's number [dimensionless less]
R_s	Solution gas-oil ratio [unit less]
$\overline{T}$	Average temperature between point 1 and 2 [°F]
T_{r1}	Critical temperature at point 1
V	Volume [ft^3/lb_m]
v	Velocity [ft/sec]
v_m	Mixture velocity [ft/sec]
v_{m1}	Mixture velocity at point 1 [ft/sec]
v_{m2}	Mixture velocity at point 2 [ft/sec]
v_{SG}	Superficial gas velocity [ft/sec]
v_{SL}	Superficial liquid velocity [ft/sec]
v_{SG1}	Superficial gas velocity at point 1 [ft/sec]
v_{SL1}	Superficial liquid velocity at point 1 [ft/sec]
WOR	Water-oil ratio [unit less]
z	Gas compressibility factor [unit less]
$\overline{z}$	Average gas compressibility factor [unit less]

Δp	Pressure difference between point 1 and 2 [psi]
Δh	Depth difference between point 1 and 2 [ft]
$\overline{\rho}_m$	Average mixture density [lb_m/ft^3]
$\overline{\rho}_g$	Average gas density [lb_m/ft^3]
$\overline{\rho}_L$	Average liquid density [lb_m/ft^3]
γ_g	Specific gravity of gas [unit less]
γ_o	Specific gravity of oil [unit less]
γ_w	Specific gravity of water [unit less]
μ_g	Viscosity of gas [cp]
μ_o	Viscosity of oil [cp]
μ_w	Viscosity of water [cp]
σ_o	Interfacial tension between oil and gas [dynes/cm]
σ_w	Interfacial tension between water and gas [dynes/cm]
ϕ_{HL}	Hold up correlation
ϕ_s	Secondary correlation
$\frac{H_L}{\Psi}$	Hold up factor
ε	Absolute roughness [ft]

Chapter 4-2

c	Compressibility [/psi]
c_e	Formation heat capacity [$Btu/lb_m/^oF$]
C_J	Joule-Thomson coefficient [^oF/psi]
c_p	Specific heat capacity [$Btu/lb_m/^oF$]
g	Acceleration due to gravity [ft/sec^2]
g_c	Unit conversion factor [32.2 $lb_m ft/lb_f/sec^2$]
H_1	Enthalpy at point 1 [Btu/lb_m]
H_2	Enthalpy at point 2 [Btu/lb_m]
h_1	Depth at point 1 [ft]
h_2	Depth at point 2 [ft]
J	Unit conversion factor [$Btu/lb_f ft$]
k_e	Heat conductivity of formation [$Btu/hr/ft/^oF$]
L_R	Relaxation distance parameter [1/ft]
Q	Heat flow rate [Btu/hr/ft]
Q_{out}	Heat flow rate into the formation [Btu/hr/ft]
$Q_{friction}$	Heat loss rate due to friction [Btu/hr/ft]

r	Radius [ft]
r_{wb}	Well bore radius [ft]
r_{to}	Tubing outside radius [ft]
t	Production time [hr]
t_D	Dimensionless time [dimensionless]
T_D	Dimensionless temperature [dimensionless]
T_e	Temperature at the external boundary of the well [°F]
T_f	Fluid temperature in the tubing [°F]
T_{wb}	Wellbore temperature [°F]
U_{tot}	Overall heat transfer coefficient of completion wall [Btu/hr/ft^2/°F]
v_1	Velocity at location 1 [ft/sec]
v_2	Velocity at location 2 [ft/sec]
ρ_e	Density of formation [lb$_m$/ft^3]
μ	Oil viscosity [cp]
ϕ	Formation porosity [unit less]

Chapter 5

C_1, C_2, C_{ia}, C_{ib}	Constants
$K(T)$	Ozawa crystallization rate function
n	Ozawa exponent
T	Temperature
ΔT	Temperature differential
X_r	Relative crystallinity
λ	Cooling rate

Chapter 5-1

$C_1^{'}, C_2^{'}$	Constants
$C_{1n}^{'}, C_{2n}^{'}, C_{3n}^{'}$	Constants
$K(\theta)$	Normalized crystallization rate function
n	Ozawa exponent
T	Temperature
$T_{\max}$	Temperature at which all paraffins and waxes are melted or dissolved
$T_{\min}$	Temperature below which crystal growth ceases
X_r	Relative crystallinity
θ	Normalized temperature
φ	Normalized cooling rate

λ	Cooling rate
λ_{eff}	Highest cooling rate at which wax the earliest wax particles appear

Chapter 5-2

C	Ozawa Constant [unit less]
$T_d(z)$	Temperature of the inside of the tubing wall
X_r	Relative crystallinity
$WAT(z)$	Wax appearance temperature as a function of well depth
$v(z)$	Velocity as a function of well depth
$\frac{\delta T_d(z)}{\delta z}$	Axial temperature gradient of the inside of the tubing wall

Chapter 6-4

c_p	Specific heat capacity
D	Pipe diameter
q_L	Mass flow rate
T_{amb}	Ambient temperature (tubing head temperature)
T_{in}	Fluid temperature at inlet (bottom-hole temperature)
$T(x)$	Fluid temperature at depth x
U_{tot}	Overall heat transfer coefficient
x	Depth

Appendix B

A_r	Heat transfer surface area
d	Diameter
$\hat{C}_p$	Specific heat capacity
$\dot{e}$	Energy flux
$\dot{E}$	Energy flow rate
$\hat{H}$	Enthalpy per unit mass
h	Convective heat transfer coefficient
K_{JT}	Joule-Thomson coefficient
k	Thermal conductivity
Nu	Nusselt number
P	Pressure

Pr	Prandtl number
Q	Heat transfer
q	Flow rate
r	Radius (variable)
R	Radius of tubing (when there is so subscript)
Re	Reynolds number
T	Temperature
U	Overall heat transfer coefficient
u	Specific internal energy
v	Velocity
β	Coefficient of expansion
γ	Fraction of area covered by open slots
δ	Unit vector in a certain direction
μ	Viscosity
π'	Tensor stress
ρ	Density
τ	Shear stress

CHAPTER 1

INTRODUCTION

1-1 Background

The use of oil and gas has had a big effect on our societies. The dense source of energy has provided mankind with many luxuries as well as necessities, such as easy access to clean water, food preservation, disease control, etc. (Ezzati et al. 2004). Oil demand in 2004 in Canada alone was 2.3 million barrels per day for its 32 million people (CBC News 2005, Fourth quarter 2006). Worldwide oil demand in 2009 was 85 million barrels per day (International Energy Agency 2010), and this number is expected to rise as countries develop and populations grow, even while infrastructures develop for other sources of energy (Levant 2010). As a result, there is immense pressure to have secure oil supply sources.

During the oil embargo set by the OAPEC countries in 1973, it became very clear that oil was an extremely important driving force of world economies, and that controlling its supply could be a powerful political tool (Essley 1974). This experience forced new means of diplomacy and cooperation to ensure oil supply would be steady. Oil supply also controls the market oil price, which in turn determines the amount of funds available to companies for well exploration and research projects. Such projects are essential to keep up with the growing

demand - technological advances allow new wells to be drilled in extreme conditions, such as offshore in deep water and in the arctic. It is also important to counter act the declining productivity of aging wells by using enhanced production, optimizing production plan, etc. to recover as much oil as possible. The use of such new technologies in the industry is calling for improved simulation models to plan well design and operations.

Simulation programs are complex mathematical models that are able to calculate useful parameters. In the case of a producing well in an oil field, the useful operating parameters are pressure and flow rates along the length of the well. If inappropriate operating pressure and/or flow rates are used, it could be a safety concern (e.g. causing blowouts), or it could reduce the life of the well (e.g. early gas /water breakthrough could occur if there is a sharp drop in pressure along the well, or, flow assurance problems could occur if timely cleanup operations are not performed, etc.). Thus having the proper simulation tools could result in big rewards - both for the performance of the operator, as well as for the optimal use of available natural resources. This is warranted by the fact that companies are willing to make big investments to obtain and support the development of such software.

There are a number of simulation software packages that are commonly used in the industry. These software packages are tools to the reservoir engineers to accurately plan and design well operating conditions. They incorporate many

models that can perform calculations on complex oil reservoir situations, such as aquifer support, gas/water injection, etc. Over the years, these packages have been improved and new calculation schemes have been added. This thesis is a step in that direction, proposing a new method of calculating well operating conditions that could be suitable for horizontal well applications. The research work also calculates temperature distribution, which can have many applications, such as investigation of flow assurance problems due to wax deposition in the well. The complete body of work can be summarized by the research objectives outlined in the next section.

1-2 Research Objectives

1. To make a vertical flow model that calculates the pressure, flow rate and temperature profiles
2. To apply proper fluid characterization to horizontal and vertical flow model
3. To calculate operating pressure and flow rate and temperature
4. To investigate the effect of different completions on operating parameters
5. To compare analytical and numerical temperature models
6. To evaluate conditions when wax precipitation is expected

1-3 Relevance of this Research

In producing wells, the oil from the reservoir flows into the perforated section of the tubing. This section is designed to optimize production as well as maximize the useful life of the well. It is increasingly common for this section to be horizontal, such that there is more contact with the oil in the reservoir and hence higher rate of production (Figure 1-3-1).

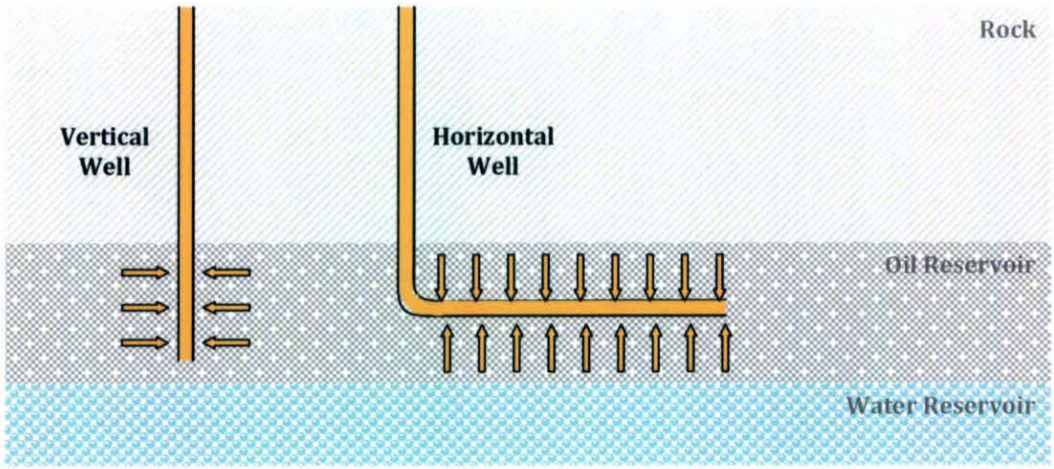


Figure 1-3-1 Types of Oil Wells

In this study, a theoretical model is applied to calculate the pressure, flow rate and temperature profiles across the horizontal section. This model is referred to as the "horizontal flow model". It can be categorized as a "hard wired" model, i.e. all the parameters are part of a single non-linear mathematical unit, and are all solved at the same time. This approach makes the model very stable during the iterative solving process. Following is a short list of the importance of applying a good horizontal flow model.

- It is imperative to maintain the pressure above bubble point in the reservoir to ensure that fluid is produced in liquid form. If fluid sits below bubble point, the dissolved gas in the fluid is liberated in the reservoir pores, which could cause flow restrictions and lower oil production. Pressure profile calculations in the horizontal well help to investigate such possibilities by allowing the reservoir engineer to see where the lowest pressures occur, how it could affect production performance, if subsurface pressure support is required or not, etc.
- Flow rate and pressure calculations are essential to optimize production. In a producing well, the quickest way of controlling the operation is by changing the surface flow rate using the "choke". Calculating flow rate profile along the horizontal part of the well would be the first step to find out how the choke should be controlled to provide the desired pressure profile for optimal production.
- The expected pressure profile, together with reservoir geology, can indicate if there are risks for gas breakthrough, increased water production, etc. - factors that affect the life of the well. This in turn could dictate the design of the completion.
- When the calculated horizontal temperature profile is matched with collected temperature data along the horizontal completion, detailed information about the "skin" values (i.e. the extent of flow restrictions in the near well reservoir) can be obtained. This information can be used by reservoir engineers for production planning as the well ages.

Therefore, there are numerous motivations to calculate the flow rate, pressure and temperature using the horizontal flow model. By applying the mathematical model for a given reservoir and a given completion, one can find out crucial information about operating the specific reservoir. The results presented in this study will touch on the first three points noted above.

In the remaining section of the well, the fluid flows upwards from the reservoir to the surface. A different model is applied to this section, because, unlike the horizontal section, fluids do not enter this part of the well and the flow is against gravity. This is termed as the "vertical flow model" in this thesis. The important parameters calculated for this vertical section are the pressure, flow rate, liquid hold up (volume fraction of liquid) and temperature profiles. This model can be categorized as a "correlation" because its equations are adjusted using field data, as opposed to fully theoretical equations in the horizontal flow model. The correlation allows the use of one simplified method to calculate parameters for various flow regimes expected in the well, such as bubble flow, slug flow, etc. The output from this correlation can be useful in many ways, two of which will be discussed in this thesis. These are introduced below.

- The pressure and flow rate profile obtained from the vertical flow model connects the horizontal flow model outputs to surface control equipment measurements. The horizontal flow model is used to check that a workable pressure level is maintained at the reservoir depth. However, the only way

to control this pressure at the bottom of the hole is by using the choke at the surface of the well (tubing head). The vertical flow model calculates the flow rate that has to be maintained at the tubing head, such that it is possible to have the desired pressure at the bottom of the well.

- As fluids flow up the well, they experience big changes in pressure and temperature. This leads to phase changes of certain components, such as ashphaltenes and waxes. These newly formed solids may deposit along the tubing wall and cause flow assurance problems. Wax deposits can clog the tubing such that wells have to be shut down and production has to be abandoned. In the less extreme instances, expensive "pigging" methods have to be utilized for the wells to operate properly (Begatin et al 2008). By having a good idea of wax deposition issues that can occur, it may be possible to design and operate in a way that would address these issues. In this study, a thermodynamic correlation is used to calculate wax crystallization profile which utilizes the temperature change as the primary driving force for wax to change into solid phase. This temperature profile is generated from the vertical flow model.

Unlike the horizontal flow model where all the parameters are calculated together, the vertical flow model is solved by iterating between two different models – the momentum balance and the temperature balance. These models were developed separately and, therefore, it is possible for them to function without being coupled together. However, a temperature profile has to be

assumed to run the momentum balance independently, and a pressure profile has to be assumed to run the temperature balance independently. In this study, a method was developed to intertwine the two models in a way that the final output values would not have such underlying assumptions. By linking up two separate models, the simulation is very versatile because it allows the possibility of other such models to be linked in. Since the calculations are not done in tandem, there is also more flexibility regarding the sequence of calculations. For instance, in this study, the temperature balance calculations starts with the bottom hole fluid temperature and solves sequentially all the way to the final value of tubing head fluid temperature; however, the momentum balance solution starts at the tubing head pressure and solves sequentially down to the value of bottom hole pressure. Thus, the two solution methods run in opposite directions, which conveniently accounts for the fact that during production, it is the tubing head pressure and the bottom-hole temperature that are known.

The effect of integrating various mathematical models and solving them in creative ways is of interest in academic circles and to software developers. The practical application of this research work is of interest to the companies that operate oil fields. Thus, this thesis touches on both industrial and academic motivations of developing a simulation program for use in the upstream oil and gas business.

1-4 Scope of the Study

This research work is focused on making a program that combines a few existing, highly regarded models. These models are combined such that an optimum operating range is calculated, and predicts flow assurance problems from wax deposition. This would be the first time that the complete, elaborate forms of these three models are put together. In addition, specific reservoir and fluid properties will be used, which will allow the program to be as field specific as possible. The literature review shown in the next chapter will describe in details why these models were selected. The technique of solving and utilizing these models will be described in details, as well as the outputs will be shown. Results will be displayed for various reservoir properties, and for various well completions. Also, the model outputs will be compared with results from current a commercial software package, as well as field data to evaluate its performance. Using these findings, it will be possible to comment on further work that could be done to make valuable contributions to this research.

1-5 Organization of the Thesis

This thesis is organized into seven chapters. This current chapter introduced the topic and described its relevance in the oil and gas industry. Chapter 2 outlines the literature review that was performed to get the directions for this study. Chapter 3 describes the horizontal flow model used in this study. Chapter 4 covers the vertical lift correlation and temperature model. Chapter 5 outlines the

wax deposition models used in this thesis. In chapter 6, the models from chapters 3 to 5 are combined together and the results are shown. Operating pressure and flow rates are calculated for specific fluid and reservoir, for different well completions. Wax deposition profiles are also shown. Chapter 7 summarizes the conclusions from this research work, comments on its novelty and makes recommendations for further study. The MATLAB computer programs developed in this research are contained in the Appendix A. For ease of future referencing, all the parameters that describe the mathematical models are defined in the nomenclature section prior to Chapter 1.

CHAPTER 2

LITERATURE REVIEW

In this study, a computer program is proposed that combines a horizontal flow model and a vertical flow model to calculate the parameters in a well. Gilbert (1954) was the first to split the oil production system to calculate a separate inflow performance and a separate vertical lift performance. In the same decade, the use of computer systems saw its light in the oil and gas industry. Warren and Mueller (1957) were among the first to solve a common reservoir engineering problem using computer technology. This new technology quickly solved complex calculations over very fine integral steps, and hence provided useful information that was previously unavailable (McCarty and Peaceman 1957). Over the decades, a lot of research work has been done using computers to solve novel problems in the oil and gas industry. Brill and Arirachakaran (1992) classified the developments in multiphase flow modeling into three broad categories. At first, empirical models were used to approximate pressure and flow profiles. In the next stage, computers were used to do full calculations of the empirical models, which highlighted their shortcomings. Since the 1980's, better models have been developed with the aid of better testing and measuring instruments in completion. The following sections describe some of the models relevant to this research work.

2-1 Horizontal Model

Some of the first horizontal pipe flow models developed was for the purpose of fluid movement between surface equipments. These empirical models, such as the ones developed by Dukler et al. (1969) and by Beggs and Brill (1973), considered frictional pressure drop, flow regimes and liquid hold up during fluid flow along pipes only. Such models are different from the horizontal flow model used in this research work in that the horizontal flow pipe in this study is situated subsurface (in the oil reservoir itself) and there are multiple fluid entry locations situated along the horizontal section. As such, the model used in this study is better categorized as an inflow performance relation (IPR) for a horizontal well, which considers aspects of flow through a pipe and flow in the annulus (which will be further discussed in the next chapter).

The first IPR models were developed by Vogel (1968) and Fetkovich (1973). These were empirical models developed for wells with perforations in the vertical section of the well, and did not take into account rock damage zone due to drilling. Standing (1971), Dias-Couto and Golan (1982), and Lekia and Evans (1990) built on these empirical models for better prediction. However, since the 1980's, much attention was given to the use of wells with perforations in the horizontal section of the well (Nzekwu 1989). Due to improvements in drilling technologies, horizontal wells are commonly used at the present time. Therefore, availability of multi-phase IPR models for such wells is increasingly important.

In their paper, Kamkom and Zhu (2005) compiled a list of horizontal well IPR models available. They pointed out that there was a lack of good models for horizontal wells. Bendakhlia and Aziz (1989) proposed a model by improving on the work by Vogel (1968), which was further developed by Cheng (1990) to make the model specific to a bounded rectangular reservoir. Retnanto and Economides (1998) proposed the first two-phased IPR model for horizontal wells, and this is also base on the Vogel model (1968). Kabir (1992) combined these works and a solution of productivity index (such as by Joshi 1988 and Butler 1994) to propose a method to estimate the open-hole flow potential of a horizontal well. However, all these models are semi-analytical and empirical in nature since they all have their roots from the empirical Vogel model. Moreover, they are only able to provide flow rate corresponding to a certain well operating pressure. Additional models need to be used in conjunction with IPR models to get a pressure and flow-rate profile, which are useful parameters for well operations as noted before. The first of such models was proposed by Dikken (1989) for single phase, turbulent and steady-state flow. Novy (1995) used Dickken's model to determine an optimal length of the horizontal well so that frictional loses would still be insignificant. Joshi (1991) proposed a pressure drop model for single-phase flow through slotted liner. Sharma et al. (1995) incorporated Dikken's model to have well-defined reservoir inflow equations. The analytical model developed by Anklam and Wiggins (2005) provides a quick method to estimate pressure drop and flow rate profile along a horizontal well.

Even in recent times, there has been much attention on developing an accurate IPR and flow model for horizontal wells. Tabatebaei and Ghalambor (2009) pointed out the underperformance of the existing horizontal well models, and hence proposed a new semi-analytical model that incorporates multilateral wells for easy use by reservoir engineers. Jahanbani and Shadizadeh (2009) developed a method to accurately develop IPR using well test information. Ostrowski et al. (2010) outlined the complexity of modern horizontal completions, and hence proposed a model to incorporate the role of inflow control devices in the horizontal well model. Bryne et al. (2010) proposed a 3D model to accurately represent the fluid inflow equations into the horizontal well. Such continued development work affirms that there is a need for further work in developing a horizontal flow model. The most comprehensive horizontal, multi-phase model for advanced completions was developed by Johansen and Khoryakov (2007).

In this research work, a horizontal flow model is used to calculate pressure, flow rate and temperature profile for multi-phase fluid at steady state conditions. It was first developed for single phase flow by Johansen (2007). Thanyamanta et al. (2009) further developed this model to allow 2-phase flow calculations together with a temperature model. Liu (2009) incorporated 3-phase calculations into this model. This model is solved numerically by calculating at predetermined nodes throughout the length of the well iteratively. Although the calculation is complex, it can be easily performed using a computer. This complexity allows the user to define multiple entry points of reservoir fluids into the well, as well as define flow

directions as expected in the complex well completions with inflow control valves. Flow rate and pressure profile will be calculated for both inside the tubing and inside the annulus. Therefore, this model combines many of the desired characteristics that are desired in reservoir simulation software. In this study, the 2-phase model developed by Thanyamanta et al. (2009) was used. Improvements were made to it such that the model can be run using specific PVT data for characterized fluids, as well as plot IPR curves from its calculations.

2-2 Vertical Model

The first vertical flow models were developed at the same time as the first horizontal flow models for surface pipes. Poetmann and Carpenter (1952) proposed a model for vertical flow in a pipe by fitting with experimental data. It was commonly accepted that an empirical model was necessary to capture the complex effects of various flow patterns. While formulating their own vertical flow model, Duns and Ros (1963) did extensive experimentation to propose a method to define the various flow regimes. Even today, after many decades of further work on this subject, Duns and Ros' model produces good results for bubble flow, slug flow and froth region. However, in a comparative study of empirical models done by Rao (1998), it was recommended that the Hagedorn and Brown (1965) and Orkizewski (1967) models produce superior results. Hagedorn and Brown developed their model by fitting to field data, as opposed to data from laboratory experimentation. Additionally, it does not distinguish

between the various flow regimes. Orkiezewski proposed a model that combined Hagedorn and Brown, Duns and Ros, Griffith (1962), and Griffith and Wallis (1961) models. The Standard Handbook of Petroleum and Naturail Gas Engineering (Lyons and Plisga 2005) recommend that the Hagedorn and Brown method and Orkizewski method be used in conjunction with each other, because the latter made better predictions for extreme flow situations, such as annular and mist flows.

Further development has been done in multiphase, vertical flow models over the years. Taitel and Dukler (1976), Taitel et al. (1980) and Barnea (1987) proposed different methods of calculating pressure drop based on flow regimes. Many mechanistic models have also been proposed. However, Ansari et al. (1990) and Hasan and Kabir (1990) did studies to compare empirical and mechanistic models, and concluded that there was no significant improvement in pressure drop predictions using the complex mechanistic models. Falcone et al. (2007) noted that mechanistic models are not able to handle intermittent flows in multi-phase flows very well and hence proposed to look more carefully at experimental flow loop designs. It is difficult to classify a model as fully mechanistic or fully empirical. This is because even the mechanistic models still use empirical parameters, such as friction factor, and the empirical models still use momentum balance as the basic starting point. Similar deductions about performance can be made from the results of the numeric model proposed by Cazarez-Candia and Vasquez-Cruz (2004).

In this thesis, the Hagedorn and Brown correlation is used to predict the vertical flow parameters. This model is chosen because it is still one of the most highly regarded models, and it is able to predict pressure, flow rate and liquid hold up easily. It also works well with multi-phase systems. The analysis done in the results chapter (Chapter 6) ensure that the vertical flow regimes are fit for Hagedorn and Brown's method, and does not require Orkizewski's model to supplement for certain parts. However, the Hagedorn and Brown method requires that the temperature profile of the fluid in the vertical well be known. In this study, temperature is one of the parameters that is calculated in the vertical flow model. This is done by intertwining a temperature model in the solving process.

One of the very well known temperature models in the Oil and Gas industry was developed by Ramey (1961). He took into account the conduction of heat through the wall of the vertical well completion and into the layers of rock structure. The model also takes into account the vertical transfer for heat by the fluid itself. The fluid temperature is solved over small incremental sections of the well depth. Ramey applied the concept of an overall heat transfer coefficient from Moss and White (1959). Because this model applied the fundamentals of heat transfer mechanism, there are few methods that completely deviate from Ramey's theory. Lindeloff and Krejberg (2002) used a simplified, single-phase, analytical form of Ramey's model since it is widely accepted to produce superior results. Hagoort (2004) resonated the same message, but proposed a different method of

approximating the solution at early time periods to better match field data. Wu and Pruess (1990) proposed an analytical method of solving for temperature profile, but Pruess and Zhang (2005) later proposed it would be better to have a semi-analytical method, much like Ramey's model. Ali (1981) and Wooley (1980) have proposed numerical methods to solve for temperature balance to get a better idea of the bottom-hole temperatures. However, these models are far more complex, and would depend of highly accurate field data to have significant differences from Ramey's model.

In this study, Ramey's model is used, with the added complexity of using properties from fluid characterization and simultaneous solution of the Hagedorn and Brown method. This provides the flow and pressure information along vertical wells.

2-3 Wax Deposition Model

Wax deposition in pipelines is considered one of the worst flow assurance problems encountered in the oil and gas industry (Misra et al. 1995), however, the phenomenon causing waxes to build up on pipes is still not fully understood (Merino-Garcia et al. 2007). Bidmus and Mehrotra (2004) found that wax deposition was not encountered in liquids unless there was a temperature gradient; this was the case even if the liquid contained wax crystals suspended in it. Therefore, oil field production pipelines provide excellent provisions for wax

build up, since the cylindrical coordinates for heat transfer (from the pipeline fluid to the surrounding rocks) ensures that a thermal gradient will always be present. It is also widely known that wax precipitates only after the fluid temperature goes below a certain value, known as the Wax Appearance Temperature (WAT) or Pour Point Temperature (PPT). Over the years, much experimental work has been done to improve the method of estimating this value (Erickson et al. 1993, Calange et al. 1997). Merino-Garcia et al. (2007) formulated a set of thermodynamic calculations to estimate WAT. All in all, the importance of temperature and heat transfer rate in wax deposition means that a good temperature prediction model is a prerequisite for a good wax deposition model.

Much work has also been done to develop a wax deposition model. As early as in 1988, Weingarten and Euchner (1988) had proposed a wax deposition model through experimentation. They looked at diffusion of wax molecules from the bulk fluid to the tube wall and shear dispersion as the phenomenon dictating wax deposition. Over the years, further research work has been done to improve the mathematical models and back them up with experimental data, since it is not possible to find out the extent of deposition along actual wells. A comprehensive list of these developments is noted by Nazar et al. (2001). However, a big draw back to this method is that the diffusion constant and other constants (whose values have a physical significance) are completely unknown. They are only determined in a trial an error method by fitting with experimental data. Therefore, there is no guarantee that this set of phenomena actually occurs. On

the contrary, it has been long known that wax solidification occurs because crystals form below the WAT (Holder and Winkler 1965). A new trend in wax precipitation modeling is, therefore, to look at a crystallization model as opposed to a diffusion model. Moreover, a turbulent flow regime is expected in the oil wells, where is it unlikely for wax molecules to be able to diffuse in the radial direction to the tubing wall. Even in the case of laminar flow, wax molecules may travel in the axial direction due to viscous forces from the laminar velocity profile, as opposed to diffusion forces (which require a concentration difference as the driving force). Also, when diffusion models are applied to turbulent flow situations, the effect of turbulent flow is considered to play a role only in the shear removal of deposits (Hsu et al. 1994, Hsu and Brubaker 1995)).

Some of the most widely used crystallization equations used for wax precipitation are the Ozawa equation (1971) and the Avrami equation (1940). Correra et al. (n.d.) and Fasano and Primicero (n.d.) have done extensive model development to propose a crystallization model for wax deposition. Begatin et al. (2008) noted that this model was still under development and that this approach needs to be investigated thoroughly, since the current software packages on wax deposition are not performing up to the mark. A similar model is recommended by Zougari and Sopkow (2007). In this thesis, these two models are applied to a complete simulation study for the first time.

CHAPTER 3

HORIZONTAL MODEL

In this study, the horizontal model is referred to as the Inflow Performance Relationship model for a horizontal well. The model that is used in the programming was first developed by Johansen (2007) as a single phase, hard-wired model, which uses numerical methods to solve for parameters at "nodes" defined by a grid representation of the well. Thanyamanta (2009) and Liu (2009) did additional work to allow the use of two-phase and three-phase fluids in the model. Thanyamanta's (2009) code was used as the starting point for the research work outlined in this thesis. to calculate the pressure and flow rate profiles in the horizontal well. This chapter describes the mathematical basis of the model and the additional work done on it in detail.

3-1 Horizontal Well Grid

A horizontal well generally consists of a tubing section and an annular section. Fluid flows occur in each of these sections, as well as between them. The path and direction of the flows are determined by the well completion. In this horizontal model, a grid is used to define the fluid flow. Figure 3-1-1 superimposes the network grid used in the horizontal model on a generic horizontal well. It can be seen that the grid suffices for flow everywhere in the horizontal well.

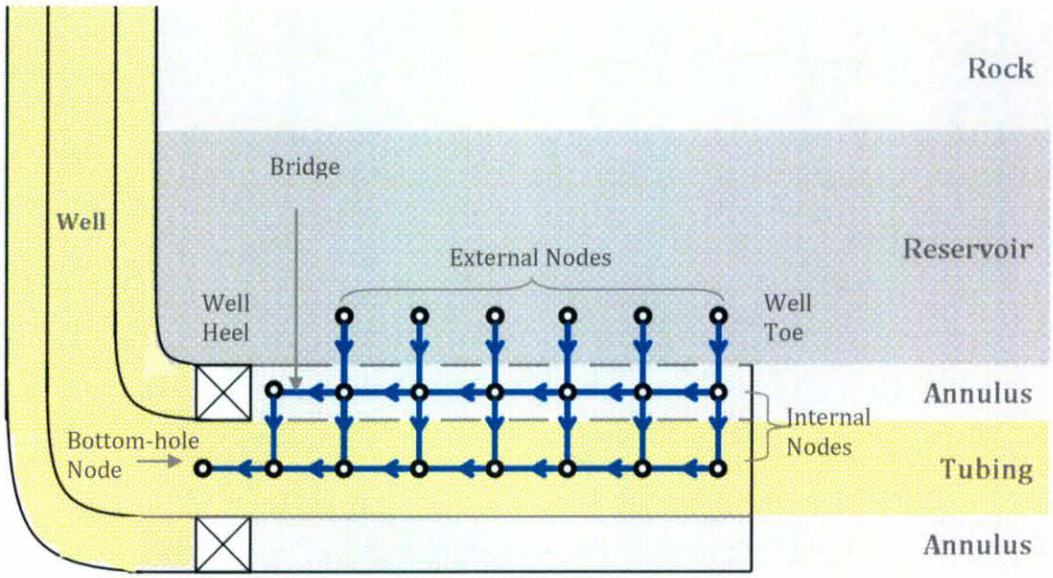
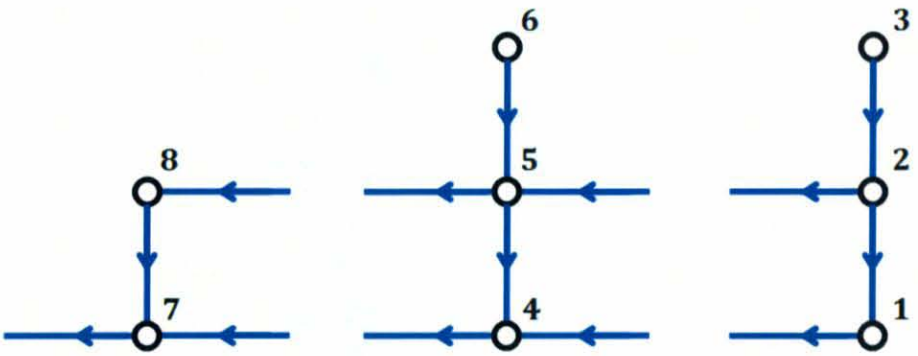


Figure 3-1-1 Horizontal Well Grid in a Completion

The users have the ability to make the grid as fine as they like. The points where the grid lines meet are called "nodes". These are the locations where most of the calculations are conducted – inflow equations and mass balances. There are two different types of nodes depending on their location – the external nodes are in the reservoir and bottom-hole, and the internal nodes are in the annulus and some in the tubing.

The boundary conditions are specified at the external nodes. The reservoir pressure and fluid saturations are set at the reservoir nodes. These parameters are used to define the inflow equations (also known as the productivity equations). The boundary condition at the bottom-hole node is the bottom-hole pressure. These boundary conditions dictate the amount of fluid that enters the well from the reservoir.

The internal nodes are found in the annulus and in the tubing section of the well completion. The nodes in the annulus combine the flow from the reservoir and the flow from the previous annular node (if applicable). The tubing nodes combine the flow from the annulus and the flow from the node where the fluid is coming from (if applicable). There is a special situation for the annular and tubing nodes at the well toe and heel – they only receive flow from the reservoir and annulus respectively, and hence do not have to account for the fluids coming from a previous node of the same type. This is demonstrated in Figure 3-1-2.



(a) Heel segment (b) Middle segment (c) Toe Segment

Figure 3-1-2 Grid at the various segments of the well

The grid lines themselves are called "bridges". The momentum balances for this system is calculated across these bridges. Thanyamanta's (2009) model also proposed a method to calculate temperature distribution along the horizontal well, by carrying out a temperature balance across the bridges. Although the temperature balance is used in the calculation process to generate the results of

this thesis, it is not the focus of the thesis. That is why the calculated temperature profile has not been investigated and commented on in this work. Therefore, the temperature balance is only described in Appendix B.

These bridges connect the various nodes, and hence dictate the direction of flow. This direction can be easily adjusted accordingly to define the effects of the inflow control valves in well completion by adjusting the "bridge index" value to be +1, 0 or -1. Figure 3-1-3 demonstrates this idea with the example of a stinger completion. A value of 0 means there is no flow in that direction, a value of +1 means flow is towards the well hell, a value of -1 means flow is away from the well toe. There are many different types of completion that would require such adjustments. Figure 3-1-4 shows the gird for a completion that has the annulus partially packed off.

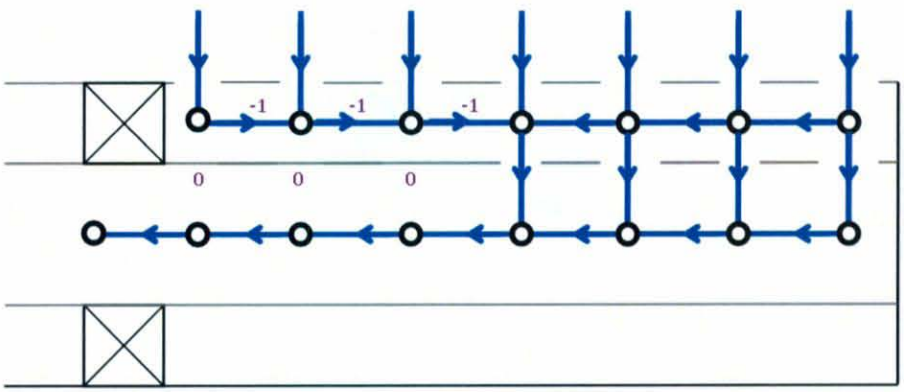


Figure 3-1-3 "Stinger" Completion Grid

This Completion has Different Bridge Indexes at six locations (values in purple)

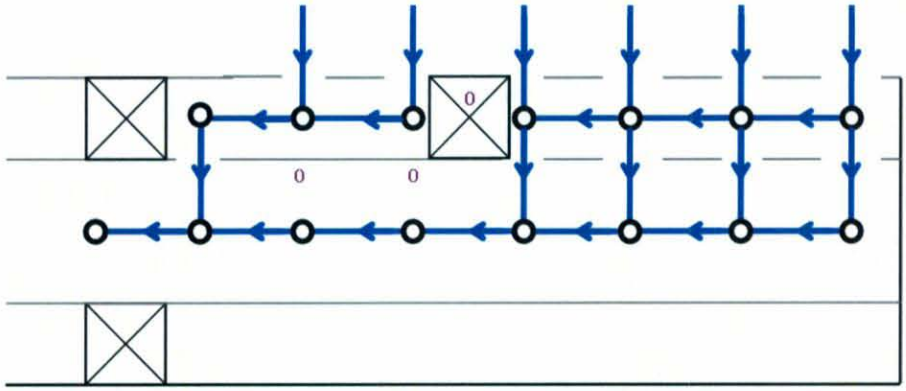
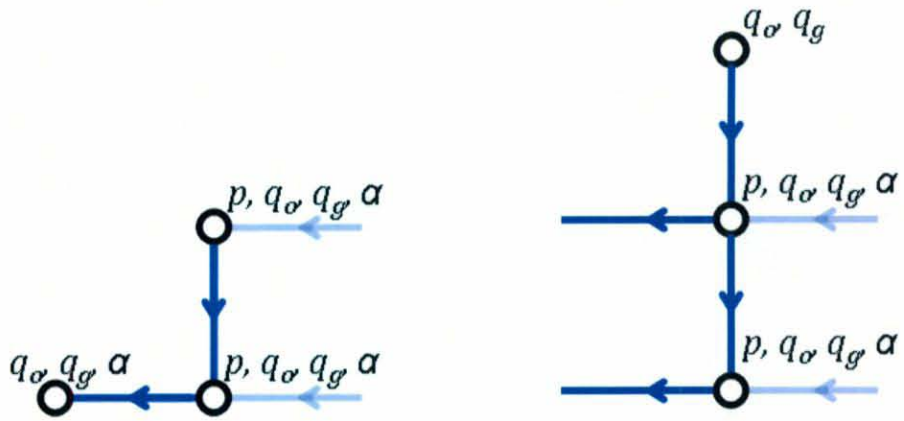


Figure 3-1-4 Completion Grid when the Annulus is Packed-Off

This Completion has Different Bridge Indexes at three locations (values in purple)

Figure 3-1-5 shows a segment of the grid network and labels the variables would need to be calculated. There are a total of 10 unknown variables, and hence 10 equations are needed to find a specific solution set. At the heel of the well, only 8 equations are needed for the 8 unknowns. The following sections lay out the equations used in the model.



(a) Heel segment (b) Middle/Toe segment

Figure 3-1-5 Grid Segment Showing the Unknown Values

3-2 Productivity Equation

A productivity equation is applied to a reservoir node. It dictates the amount of fluid flow from the reservoir to the well because the reservoir pressure and fluid saturations are specified as boundary conditions. Both the oil and gas saturations were defined, since this is a 2-phase model. The principle behind the productivity equation is the Darcy's law, which predicts flow through a porous medium due to a pressure differential, which acts as the driving force. Therefore, the well operating pressures are always less than the reservoir pressure. Following is the equation used in the model.

Eq. 3-2-1

$$q_{\inf low} = PI\left(\frac{k_{ro}(S_o)}{\mu_o} + \frac{k_{rg}(S_g)}{\mu_g}\right)\left(p_{reservoir} - p_{annulus}\right)$$

Where,

Eq. 3-2-2

$$PI = \frac{2\pi KL}{\ln\left(\frac{r_e}{r_{well}}\right) + S}$$

Eq. 3-2-1 applies to flow from each of the reservoir nodes in the well grid. Eq. 3-2-2 applies for a homogeneous, isotropic reservoir section in the vicinity of a well segment. However, it is possible to have a different set of values for the properties and parameters at each segment. This functionality allows the simulation to have a very precise picture of the reservoir conditions, since it is common to find different values for permeability, skin, etc. along the reservoir length.

Eq. 3-2-1 describes the factors that affect the inflow of fluid into the well – a higher permeability and pressure difference will increase fluid production, while a higher viscosity and skin value will do otherwise.

For the segment shown in Figure 3-1-5 (a), there is no inflow equation, since the heel of the well does not have flow coming in from the reservoir. The segment in Figure 3-1-5(b), one inflow equation applies, since there is one opening that allows fluids to flow in from the reservoir.

3-3 Mass Balance

Mass or material balance equations are applied at each internal node in the annulus and tubing. This is because at these nodes, fluids from different directions meet and then split to travel through other bridges defined by the grid. Mass balance equations are applied to each phase to ensure that the law of conservation of mass is applied to the calculations that will produce the pressure and flow rate profiles. The following set of equations shows how the mass balance equations are developed individually for oil and gas under stock tank conditions.

For each component at each internal node,

$$\Sigma m_{in} - \Sigma m_{out} = 0$$

$$\rho_{o,in}^{RC} q_{in} \alpha_{in} - \rho_{o,out}^{RC} q_{out} \alpha_{out} = 0$$

For oil component, this can be written as,

$$\frac{q_{in}\alpha_{in}}{B_o} - \frac{q_{out}\alpha_{out}}{B_o} = 0$$

And similarly for gas component,

$$\left[\frac{q_{in}(1-\alpha_{in})}{B_g} + \frac{q_{in}\alpha_{in}R_s}{B_g}\right] - \left[\frac{q_{out}(1-\alpha_{out})}{B_g} + \frac{q_{out}\alpha_{out}R_s}{B_g}\right] = 0$$

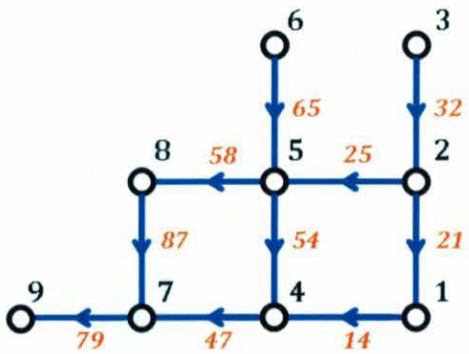


Figure 3-3-1 A Short Example Grid

The mass balance equations look different depending on the number of bridges involved in fluid passage through the associated internal node. The above mass balance equations apply for node 8 and node 1 shown in Figure 3-3-1. At these nodes, one bridge brings fluid in and one bridge takes fluid out. The mass balances for node 8 are the following equations.

$$\frac{q_{58}\alpha_5}{B_o} - \frac{q_{87}\alpha_8}{B_o} = 0$$

$$\left[\frac{q_{58}(1-\alpha_5)}{B_g} + \frac{q_{58}\alpha_8 R_s}{B_g}\right] - \left[\frac{q_{87}(1-\alpha_8)}{B_g} + \frac{q_{87}\alpha_8 R_s}{B_g}\right] = 0$$

In the cases where there is one inflow bridge and two outflow bridges at a node, as in the case of node 2 in Figure 3-3-1, the mass balance equations are as follows.

$$\frac{q_{32}\alpha_3}{B_o} - \frac{q_{25}\alpha_2}{B_o} - \frac{q_{21}\alpha_2}{B_o} = 0$$

$$\left[\frac{q_{32}(1-\alpha_3)}{B_g} + \frac{q_{32}\alpha_3 R_s}{B_g}\right] - \left[\frac{q_{25}(1-\alpha_2)}{B_g} + \frac{q_{25}\alpha_2 R_s}{B_g}\right] - \left[\frac{q_{21}(1-\alpha_2)}{B_g} + \frac{q_{21}\alpha_2 R_s}{B_g}\right] = 0$$

In the case of nodes 4 and 7 (Figure 3-3-1), there are two inflow bridges and one outflow bridges. This yields the following set of equations for node 4.

$$\frac{q_{54}\alpha_5}{B_o} + \frac{q_{14}\alpha_1}{B_o} - \frac{q_{41}\alpha_4}{B_o} = 0$$

$$\left[\frac{q_{54}(1-\alpha_5)}{B_g} + \frac{q_{54}\alpha_5 R_s}{B_g}\right] + \left[\frac{q_{14}(1-\alpha_1)}{B_g} + \frac{q_{14}\alpha_1 R_s}{B_g}\right] - \left[\frac{q_{47}(1-\alpha_4)}{B_g} + \frac{q_{47}\alpha_4 R_s}{B_g}\right] = 0$$

The combination of two inflow and two outflow bridges at node 5 (Figure 3-3-1) requires the following mass balance equations.

$$\frac{q_{65}\alpha_6}{B_o} + \frac{q_{25}\alpha_2}{B_o} - \frac{q_{58}\alpha_5}{B_o} - \frac{q_{54}\alpha_5}{B_o} = 0$$

$$\left[\frac{q_{65}(1-\alpha_6)}{B_g} + \frac{q_{65}\alpha_6 R_s}{B_g}\right] + \left[\frac{q_{25}(1-\alpha_2)}{B_g} + \frac{q_{25}\alpha_2 R_s}{B_g}\right] - \left[\frac{q_{58}(1-\alpha_5)}{B_g} + \frac{q_{58}\alpha_5 R_s}{B_g}\right]$$

$$- \left[\frac{q_{54}(1-\alpha_5)}{B_g} + \frac{q_{54}\alpha_4 R_s}{B_g}\right] = 0$$

Another aspect of mass balance that is relevant is the value of liquid fraction at the exit bridges from each node. It is assumed that the fluids get well mixed at the nodes, and hence the exit streams from each node are assumed to have to the same value for liquid fractions. This is also termed as the "split equation", and is noted below for node 5 in Figure 3-3-1.

$$\alpha_{58} = \alpha_{54}$$

Therefore, for a toe or middle segment (shown in Figure 3-1-5(b)), there are 3 nodes, one of which is part of the inflow equation. Of the remaining 2 modes, each node will have three mass balance equations, and hence 6 equations are generated for each segment. For the heel segment shown in Figure 3-1-5(a), there are 2 nodes, and hence that segment will have 6 mass balance equations.

3-4 Momentum Balance

Another factor that causes changes in pressure of a flowing fluid is the momentum balance. Gravity, fluid acceleration and frictional losses are some of the factors that are considered in this balance that considers the conservation of momentum, i.e. Newton's second law. The following equation describes this phenomenon.

$$\frac{dp}{dz} = \left[\frac{\dot{m}}{A}\frac{\delta v}{\delta z}\right] - \left[\frac{\tau_w p}{A}\right] - [\rho g \sin\theta]$$

Since the fluid flow is not against gravity in a horizontal well, the momentum loss due to gravitational pull is not relevant. The third term can therefore be dropped for the equation system. Moreover, it is understood that the contribution of acceleration to momentum is only about 10% at the maximum. Therefore, for simplicity purposes, the first term can also be ignored. The following is the equation we are left with to account to momentum loss.

$$\frac{dp}{dz} = -\tau_w \frac{p}{A} = -\tau_w \frac{4}{D_h}$$

The above form of this equation cannot be used to calculate pressure, since shear stress values cannot be estimated. However, the workable equation for pressure loss due to friction can be expressed as follows.

$$\frac{dp}{dz} = \frac{f\bar{\rho}v^2}{2D_h}$$

The Blasius friction factor, f, below for turbulent flow is used in this work. This is because the high flow rates in wells provide an environment for turbulent flow, and hence laminar flow can be ignored.

$$f = \frac{0.3164}{\sqrt[4]{\mathrm{Re}_{D_h}}} \qquad \text{where,} \quad \mathrm{Re} = \frac{\rho v D_h}{\mu}$$

The above set of equations account for pressure loss for flow through a pipe only. Another area in the well where pressure loss is expected is when fluids flow from

the annulus to the tubing due to the convergence of flow through a small opening. In this case, the equation characterizing flow through nozzle is incorporated.

$$p_{out} - p_{in} = c\rho v^2$$

These momentum balance equations are calculated across all the bridges. Therefore, in each segment, there are three momentum balances. Following is the equation for momentum balance in the annulus (node 2 to 5 in Figure 3-3-1).

$$\frac{p_5 - p_2}{L} = \frac{f\overline{\rho}}{2D_h}\left(\frac{q}{A}\right)^2$$

The equation below is the momentum balance in a tubing bridge (node 1 to 4 in Figure 3-3-1).

$$\frac{p_4 - p_1}{L} = \frac{f\overline{\rho}}{2D_h}\left(\frac{q}{A}\right)^2$$

The momentum balance of flow from the annulus node to the tubing node is as follows (flow from node 3 to 2 in Figure 3-3-1).

$$p_2 - p_3 = \frac{q^2 c\overline{\rho}}{A^2}$$

Thus, momentum balances contribute to 2 more equations to the heel segment shown in Figure 3-1-5(a), since the segment has 2 bridges. The middle and toe segments shown in Figure 3-1-5(b) shows 4 bridges, however one of these

bridges denote the inflow equation. Therefore, the segment has 3 momentum balances from the remaining 3 bridges.

The following table summarizes the equation counts for each type of segments.

Table 3-4-1 Equation Count for Each Type of Segments in the Grid

	Heel Segment	**Middle Segment**	**Toe Segment**
Unknowns (Figure 3-1-5)	11	10	10
Inflow Equation	0	1	1
Mass Balance	9	6	6
Momentum Balance	2	3	3

3-5 Fluid Properties

As can be seen from the sets of equations above, it is important to know certain fluid properties and black-oil properties in order to use the derived equations. These are R_s , B_o , B_g , μ_o and μ_g . Moreover, these values need to be known over a range of temperature and pressure values, since these properties change over the length of the well. There are many different ways of estimating these properties. The most reliable information would come from doing extensive laboratory analysis of samples taken from the reservoir – this is the method used by oil field operators to perform the most reliable simulation calculations. In the past, correlations have been developed to estimate these properties

corresponding to a specific pressure, temperature and fluid API value. Such correlations were used by Liu (2009) to propose a 3-phase calculation scheme for the horizontal model described above. Thanyamanta (2009) used the software package PVTsim™ (Calsep Inc.) to generate properties for characterized fluids, and performed regression on the data to generate an equation of state. In this study, a table of fluid properties are generated in the same way as Thanyamanta (2009) using PVTsim™ (Calsep Inc.). However, a code for a double liner interpolation is developed to calculate the properties using the table at the required pressures and temperatures. This method will easily allow different fluids to be used in the simulation. This is because the time consuming and potentially not very accurate process of regression to generate equations of state will not have to be performed. Data collected from laboratory analysis could also be used if available. Consequently, this method will generate far more specific information than using correlations. This method of calculating fluid properties is also used for the vertical model described in the next chapter.

3-6 Solution Method

As mentioned earlier, the horizontal non-linear model used in this thesis is a hard-wired model, which is solved iteratively using the Newton Raphson method. In other words, all the unknown parameters are solved simultaneously, as opposed to solving for one parameter using one scheme of calculations followed

by another in a sequential manner. The system of non-linear equations was written in the following format.

$$F_1(x_1, x_2, \ldots\ldots x_n) = 0$$

$$F_2(x_1, x_2, \ldots\ldots x_n) = 0$$

$$\vdots$$

$$F_n(x_1, x_2, \ldots\ldots x_n) = 0$$

The solution to this system was obtained by solving the following equation.

$$\vec{F}(\vec{x}) = 0$$

where,

$$\vec{F} = [F_1, F_2, \ldots\ldots F_n]$$

$$\vec{x} = [x_1, x_2, \ldots\ldots x_n]$$

The Jacobian matrix, J, of matrix $\vec{F}$ was used to solve the system. This is shown as follows.

$$J\left(\vec{x^n}\right) \cdot \vec{x^{n+1}} = \vec{x^n} - \vec{F}\left(\vec{x^n}\right)$$

In order to calculate the parameters for the first time, a set of values have to be assumed. When new values are calculated, they are tallied against the values assumed at the beginning. If these values are within the threshold limit, then these values are accepted to be correct. This can be represented by the following equation.

$$\frac{\left|y_i^{new} - y_i^{old}\right|}{\left|y_i^{old}\right|} < \varepsilon$$

If the differences between the calculated and the assumed values are big, then the newly calculated values are used as the guessed values, and the calculations are carried out again. The flowchart in Figure 3-6-1 explains this process.

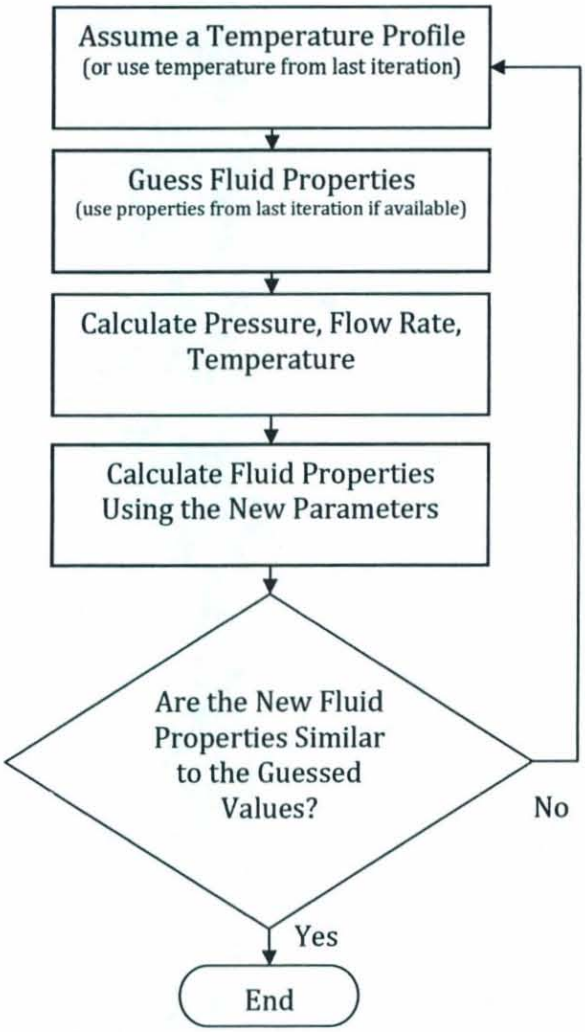


Figure 3-6-1 Horizontal Model Solution Scheme

As can be seen, Thanyamanta's model also had a temperature calculation, the details of which can be found in Appendix B. It is not described in details in this thesis, since the temperature profile will not be used in the results and discussion section.

This model calculates a pressure and flow rate profile across the horizontal well. Additionally, a set of bottom-hole pressures are given to calculate different operating flow rates. This data is then used to generate Inflow Performance Relationship (IPR) for the well.

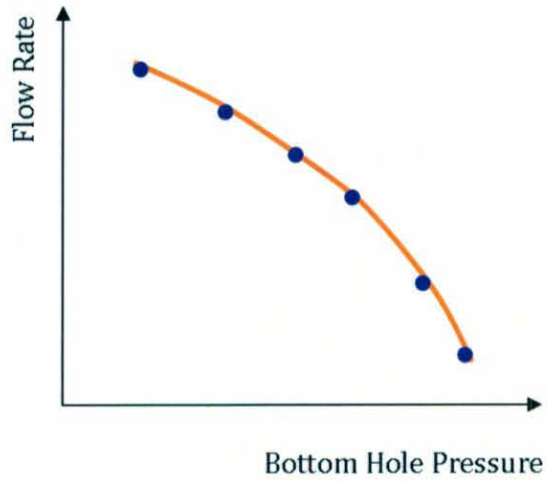


Figure 3-6-2 Typical IPR Curve

Figure 3-6-2 shows a typical IPR curve at fixed values of water-cut, gas-oil-ratio, tubing head pressure, etc. Each of the data points in the plot are generated by running the horizontal model once, each time using a different bottom-hole

pressure to calculate a flow rate. The plot demonstrates what is expected: as bottom-hole pressure is decreased, the pressure differential increases, hence there is a greater driving force for more fluids to come into the well (i.e. higher flow rate). However, the increase in flow rate is not proportional to the increase in pressure differential; this is because resistance due to friction increases at higher flow rates. IPR curve produced using the computer program quoted in this thesis is shown in the results section.

CHAPTER 4

VERTICAL MODEL

In the previous chapter, a horizontal well model was described, which generates the IPR correlation for an oil well. However, that model could only provide information about the bottom-hole conditions. The vertical model described in this chapter will calculate the pressure, temperature, flow-rate and liquid fraction profiles from the bottom-hole to the tubing head location. In order to achieve this, Hagedorn and Brown (1965) correlations are used to calculate pressure, flow rate and liquid fraction profiles, coupled with Ramey's model (1961) to calculate the temperature profile. These two models are run in series, thus all the parameters are not calculated at the same time. This is the basis of the modular nature of this part of the programming. The final solution is obtained by iterating between the two models until the estimated and calculated values converge. The detailed framework of equations is discussed in the following sections.

4-1 Pressure, Flow Rate and Liquid Hold-Up Correlation

The Hagedorn and Brown correlation proposes a series of calculations to calculate the pressure, flow rate and liquid fraction of the fluid raising up from the bottom-hole to the tubing head. The fundamental flow equation this correlation is based on is as follows. It is presented in field units, since that is how the

correlation was developed. The list of actual units used in this research work is outlined in the "Nomenclature" section of this thesis on page ix. The momentum balance equation used in this correlation is as follows.

$$144\frac{g_c}{g}Vdp + dh + \frac{vdv}{g} + dW_f + dW_e = 0$$

This equation assumes steady state flow, and the gas-liquid mixture is assumed to be a homogeneous fluid with combined properties. Expanding the frictional losses (using two phase friction factor) and assuming no external work done by the fluid, the above equation can be rearranged after fitting with field data to obtain the following correlation.

Eq. 4-1-1

$$144\frac{\Delta p}{\Delta h} = \overline{\rho_m} + \frac{fq_L^2 M^2}{2.9652\times10^{11} D^5 \overline{\rho_m}} + \overline{\rho_m}\frac{\Delta\left(\frac{v_m^2}{2g_c}\right)}{\Delta h}$$

where,

$$\overline{\rho_m} = \overline{\rho_L} H_L + \overline{\rho_g}(1 - H_L)$$

In order to use this equation, a set of pre-calculations need to be performed in series. At first, the mass of one barrel of liquid is calculated using the following correlation.

$$M = 350\gamma_o\left(\frac{1}{1+WOR} + 350\gamma_w\right)\left(\frac{WOR}{1+WOR}\right) + (0.0764)(GLR)(\gamma_g)$$

The volumetric flow rate of the liquid components (q_w, q_o) are then converted to mass flow rate q_L using the next equation.

$$q_L = M(q_w + q_o)$$

The parameters M and q_L appear in Eq. 4-1-1. In order to determine the other parameters that are present in the flow equation 4-1-1, it is important to understand the progression of calculations. Two sets of pressure and temperature values were selected, and denoted as p_1, T_1 and p_2, T_2. The values for p_1, T_1 are known at the tubing head, which will be at the boundary conditions. The guessed values at the next point are p_2, T_2. This is demonstrated in Figure 4-1-1.

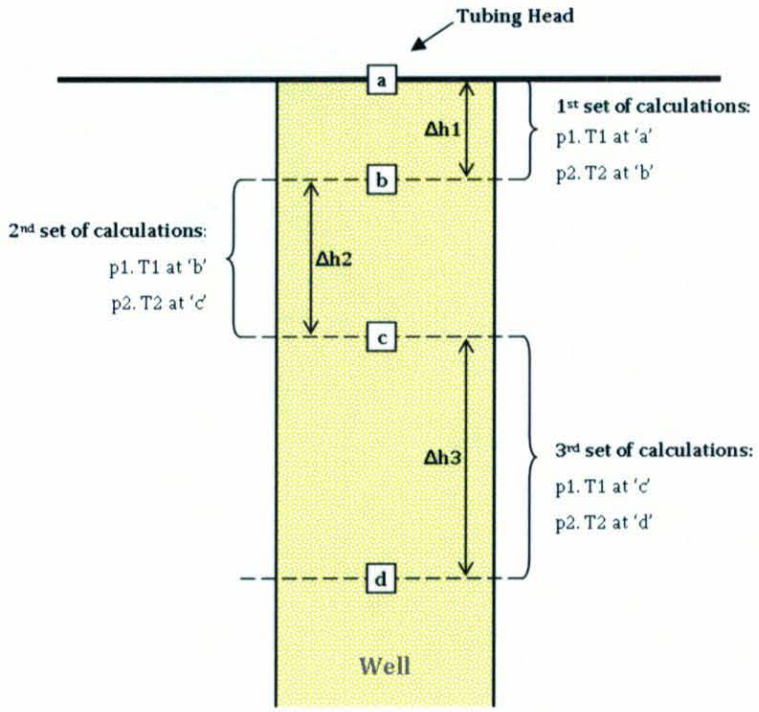


Figure 4-1-1 Vertical Model Calculation Progression

By iterative method, the correct values of p_2, T_2 will be calculated, and the location in the tubing depth where these values occur. In the next set of calculations, these known values of p_2, T_2 will become the p_1, T_1 for the next incremental calculations. However, for the initial guess, p_2 and T_2 are both assumed to be 10% greater than p_1 and T_1. This is demonstrated by the flowchart below (Figure 4-1-2).

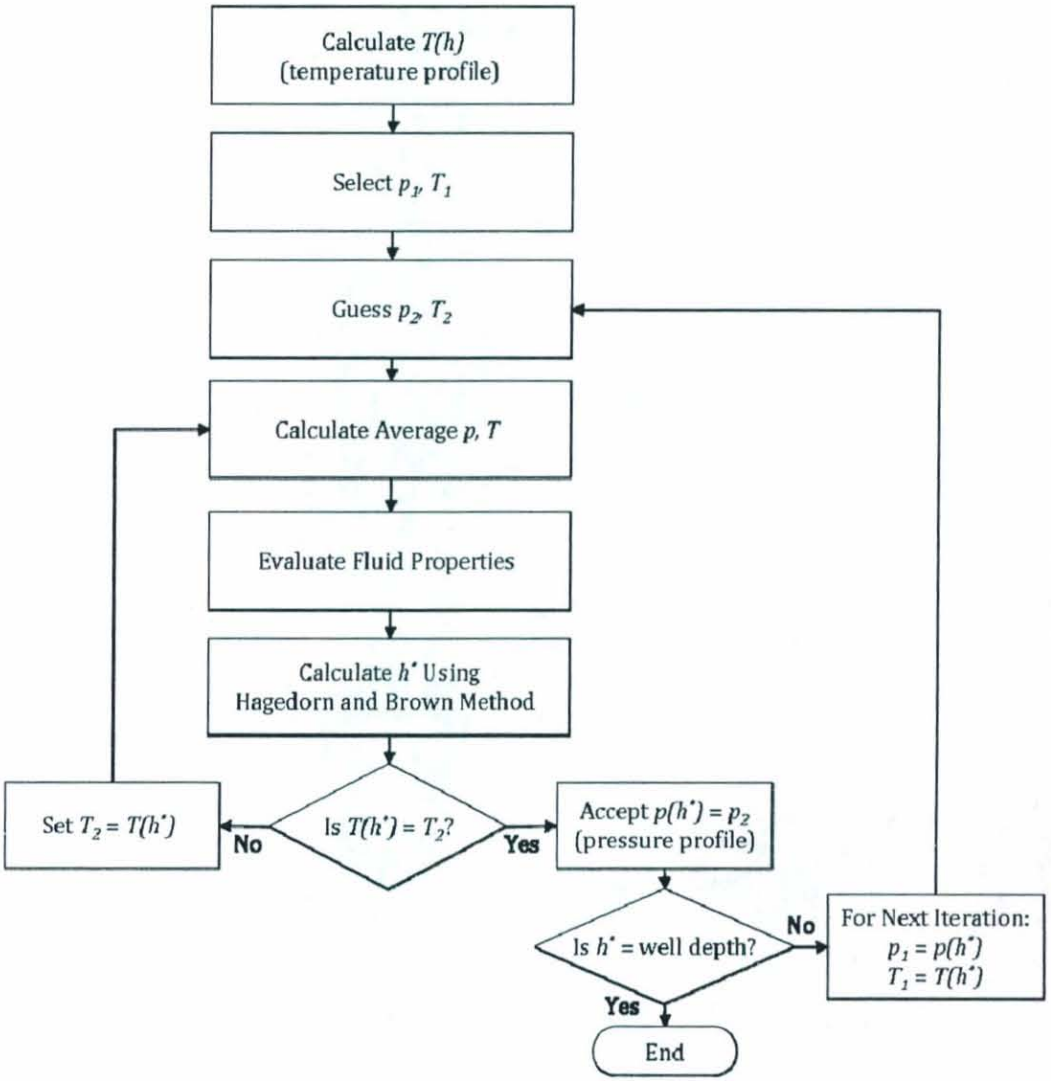


Figure 4-1-2 Hagedorn and Brown Calculation Scheme

As can be seen from the flow chart, the temperature profile needs to be known for this method to provide the pressure profile, with the corresponding flow rate and liquid fractions. The details of the temperature calculations are described in the next sections.

The fluid properties are evaluated by interpolating from data generated in PVTsim (as described in section 3-5). These properties include R_s, B_o, B_w, z, μ_o, μ_w, σ_o, σ_w, γ_o, γ_g and γ_w. The viscosity and the surface tension of the liquid phase (oil and water together) are averaged using the following equations.

$$\mu_L = \mu_o\left(\frac{1}{1+WOR}\right)+\mu_w\left(\frac{WOR}{1+WOR}\right)$$

$$\sigma_L = \sigma_o\left(\frac{1}{1+WOR}\right)+\sigma_w\left(\frac{WOR}{1+WOR}\right)$$

In the next step, the superficial liquid and gas velocities are calculated.

$$v_{SL} = \frac{5.615 q_L}{86400 A_t}\left[B_0\left(\frac{1}{1+WOR}\right)+B_w\left(\frac{WOR}{1+WOR}\right)\right]$$

$$v_{sg} = \frac{q_L\left[GLR - R_s\left(\frac{1}{1+WOR}\right)\right]}{86400 A_t}\left[\frac{14.6}{\overline{p}}\right]\left[\frac{\overline{T}+460}{520}\right]\overline{z}$$

The above values are used to calculate the L_B and BB parameters (defined below), which in turn are used to check if Hagedorn and Brown method is suited for this

calculation. The fluid examples shown in the results section were all checked to ensure that the Hagedorn and Brown method was suitable.

$$L_B = 1.071 - \frac{0.2218(v_{SL} + v_{SG})^2}{\left(\frac{D}{12}\right)}$$

$$BB = \frac{v_{SG}}{v_{SL} + v_{SG}}$$

This model was proposed such that L_B could only have a value of 0.13 or greater. If a value of less than 0.13 was calculated for this parameter, then the value 0.13 was used instead of the calculated value in the next step. If $(BB - L_B)$ yields a positive value, only then it is recommended that the Hagedorn and Brown method be used. Otherwise, the Orkiszewski method is preferred. This is how it was checked to ensure the suitability of the Hagedorn and Brown method. For all the calculations performed in this thesis, it was checked that the Hagedorn and Brown method was suitable for the purpose. Therefore, the Orkiszewski method is not used.

The correlation to calculate liquid holdup consists of the following set of calculations in series.

$$N_{LV} = 1.938 v_{SL} \left(\frac{\rho_L}{\sigma_L}\right)^{0.25}$$

$$N_{GV} = 1.938 v_{sg} \left(\frac{\rho_L}{\sigma_L}\right)^{0.25}$$

$$N_L = (0.15726)(\mu_L)\left(\frac{1}{\rho_L \sigma_L}\right)^{0.25}$$

$$N_D = 120.872\left(\frac{D}{12}\right)\left(\frac{\rho_L}{\sigma_L}\right)^{0.5}$$

The work of Hagedorn and Brown compiled a number of graphs that facilitate the calculation procedure. The graph of CN_L vs. N_L gives CN_L, which is used in the next set of calculations to get the holdup correlation function.

$$\phi_{HL} = \left[\frac{N_{LV}}{(N_{GV})^{0.575}}\right]\left[\frac{\overline{p}}{14.7}\right]^{0.1}\left[\frac{CN_L}{N_D}\right]$$

A different graph of $\frac{H_L}{\Psi}$ vs. ϕ_{HL} is then used to provide the corresponding value of $\frac{H_L}{\Psi}$. The secondary correlation factor is calculated next, as shown below.

$$\phi_S = \frac{N_{GV} N_L^{0.38}}{N_D^{2.14}}$$

This value is used to find out the value of Ψ from the graph of Ψ vs. ϕ_S. At this point, the liquid hold up value, i.e. the fraction of liquid in the fluid, can be calculated.

$$H_L = \left(\frac{H_L}{\Psi}\right)\Psi$$

The two phase density is then calculated using two different methods, as shown by the equations below.

$$\rho_m = \rho_L H_L + \rho_g (1 - H_L)$$

$$\rho_m = \frac{530\gamma_o + 0.0764\gamma_g GLR + 350\gamma_w WOR}{5.615B_o + 5.615WOR + \left[(GOR - R_s)\left(\frac{14.7}{\overline{p}}\right) + \left(\frac{\overline{T} + 460}{520}\right)\overline{z}\right]}$$

The bigger of the two values are used in the final flow equation (Eq. 4-1-1) to determine the depth and its corresponding pressure.

Friction factor is one of the parameters that appear in the final flow equation (Eq. 4-1-1). This is determined by first calculating the two phase Reynold's number, and then evaluating the friction factor, using the equations below.

$$\mathrm{Re}_{TP} = \frac{2.3 \times 10^{-2}}{\left(\frac{D}{12}\right)(\mu_L)^{H_L}(\mu_g)^{1-H_L}}$$

$$f = \left\{1.8\log_{10}\left[\frac{6.9}{\mathrm{Re}} + \left(\frac{\varepsilon}{3.7D}\right)^{\frac{10}{9}}\right]\right\}^{-2}$$

where, ε was assumed to be a constant value of 0.00015.

The last parameter that appears in Eq. 4-1-1 is the velocity parameter. It is calculated by doing a series of calculations, once at p_1, T_1 conditions, and then at p_2, T_2 conditions. The following are the equations to calculate at the p_1, T_1 location.

$$T_{r1} = \frac{T_1 + 460}{T_c}$$

$$p_{r1} = \frac{p_1}{p_c}$$

$$v_{SL1} = \frac{5.615 q_L}{86400 A_t}\left[B_0\left(\frac{1}{1+WOR}\right) + B_w\left(\frac{WOR}{1+WOR}\right)\right]$$

$$v_{SG1} = \frac{q_L\left[GLR - R_s\left(\frac{1}{1+WOR}\right)\right]}{86400 A_t}\left[\frac{14.6}{\overline{p}}\right]\left[\frac{\overline{T}+460}{520}\right]\overline{z}$$

Once the superficial liquid and gas velocities at p_1, T_1 and p_2, T_2 are known, then the final parameter can be calculated as follows.

$$v_{m1} = v_{SL1} + v_{SG1}$$

$$v_{m2} = v_{SL2} + v_{SG2}$$

$$\Delta(v_m)^2 = (v_{m1})^2 - (v_{m2})^2$$

The incremental depth between location of p_1, T_1 and p_2, T_2 is then calculated by rearranging the original flow equation (Eq. 4-1-1). Figure 4-1-1 demonstrates where the physical significance of where this incremental depth appear. Equation 4-1-1 can be rearranged to solve for the incremental depth as follows.

$$\Delta h = \frac{144\Delta p - \overline{\rho_m}\Delta\left(\frac{v_m}{2g_c}\right)^2}{\overline{\rho_m} + \frac{f q_L^2 M^2}{2.9652\times 10^{11}\overline{\rho_m} D^5}}$$

As can be seen from the calculations, this method calculates the pressure, flow rate and liquid hold up profiles of the vertical section of the well.

4-2 Temperature Balance

We know from the previous section, it is imperative to know the temperature profile to be able to determine pressure profile in the vertical section, which in turn is essential to determine the operating conditions of the well. In this section, the equations governing the temperature profile from Ramey's method (Ramey 1961) are described.

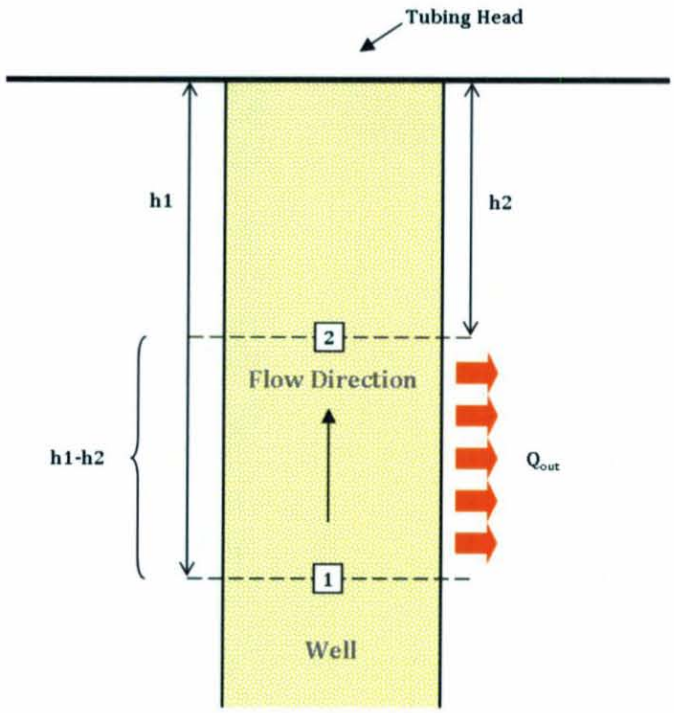


Figure 4-2-1 Temperature Balance Schematic

Ramey developed an energy balance between two points along the well tubing. This is demonstrated in Figure 4-2-1 by locations "1" and "2". The energy at each point was evaluated in terms of enthalpy, potential energy and kinetic energy of the fluid. As fluid travelled from point 1 to point 2, heat is generated due to friction. Some heat is transferred to the surrounding rock (since the fluid is warmer). The following equation compiles this information.

$$q_L H_1 + q_L h_1 \frac{g}{Jg_c} + \frac{1}{2} q_L \frac{v_1^2}{Jg_c} = q_L H_2 + q_L h_2 \frac{g}{Jg_c} + \frac{1}{2} q_L \frac{v_2^2}{Jg_c} - Q_{out} + Q_{friction}$$

This equation can be rearranged as follows.

$$\frac{\Delta H}{\Delta h} + \frac{g}{Jg_c} + \frac{1}{2} \frac{\Delta v^2}{Jg_c} \frac{1}{\Delta h} = \frac{-Q_{out} + Q_{friction}}{q_L}$$

where,

$$\Delta H = c_p (T_1 - T_2) - C_J c_p (p_1 - p_2)$$
$$\Delta h = h_1 - h_2$$
$$\Delta v^2 = \Delta v_1^2 - \Delta v_2^2$$

In order to solve this equation, each of the parameters needs to be evaluated first. All the parameters can be evaluated once the fluid properties are known, and from the calculations done in the Hagedorn and Brown process, except for Q_{out} and $Q_{friction}$. Ramey proposed that $Q_{friction}$ can be ignored. Therefore, a method was devised to calculate Q_{out}.

It was assumed that heat is transferred from the fluid to the rock through conduction. This is because, from the point where the fluid touches the tubing wall (i.e. where the heat exchange occurs) and beyond, all the layers of material through which heat transfer takes place are solid. Heat conduction can be defined by Fourier's law, which states that the rate of heat transfer is proportional to the temperature differential. The heat transfer in the formation itself can be written in cylindrical coordinates as follows.

$$\frac{\delta^2 T_e}{\delta r^2} + \frac{1}{r}\frac{\delta T_e}{\delta r} = \frac{c_e \rho_e}{k_e}\frac{\delta T_e}{\delta t}$$

One needs to know the steady-state fluid rock temperature in order to calculate the steady state fluid temperature. However, the problem is that the rock temperature never reaches a steady-state value because of the cylindrical geometry of the direction of heat transfer (i.e. the solution to the above equation is not unique). As time progresses, the heat from the fluid will heat up the rock farther away; hence the temperature boundary in the rock will keep on moving farther away from the tubing. This is shown by the Basel function solution as follows.

$$T_D = -\frac{2\pi k_e}{Q}(T_{wb} - T_e)$$

where,

$$T_D = 0.4063 + \frac{1}{2}\ln t_D \text{ (at large times/semi-steady flow)}$$

$$t_D = \frac{k_e t}{\phi \mu c r_{wb}^2}$$

The conductive heat transfer from the wellbore to the formation can be written as follows.

$$Q = -\frac{2\pi k_e}{T_D}(T_{wb} - T_e)$$

The overall heat transfer value, U_{to}, is assumed to be a constant value of 17.61 Btu/(hr.ft^2.^{o}F), as calculated by Dawkrajai et al. (2005). The conductive heat transfer from the fluid to the wellbore (through the layers of completion) can be expressed by the equation below.

$$Q = -2\pi r_{to} U_{to}(T_f - T_{wb})$$

By equating the above two equations, the following equation is obtained.

$$Q_{out} = -L_R q_L c_p (T_f - T_e)$$

where,

$$L_R = \frac{2\pi}{c_p q_L}\left[\frac{r_{to} U_{to} k_e}{k_e + (r_{to} U_{to} T_D)}\right]$$

This represents the heat loss from the fluid to the formation, i.e. Q_{out}. T_D is calculated by assuming a constant time. Therefore, this model calculates the temperature profile at a specific time only. This allows the model to be versatile in the ability to recognize the drop in fluid temperatures at the tubing head as time progresses. This could be used to estimate the time when the fluid temperatures could go below the WAT.

Since now all the parameters of the tubing energy balance can be calculated, it can be written in the following form.

$$\left[c_p(T_1 - T_2) - C_J c_p(p_1 - p_2)\right] + \left[(h_1 - h_2)\frac{g}{Jg_c}\right] + \left[\frac{1}{2}\frac{(v_1^2 - v_2^2)}{Jg_c}\right]$$
$$= \left[L_R c_p (h_1 - h_2)T_2 - L_R c_p (h_1 - h_2)T_e\right]$$

When calculating the temperature profile, the starting point is the tubing head temperature. This is because the temperature at the bottom-hole is fixed at the reservoir temperature, and it does not change. Depending on the time, flow rate, and other factors, the temperature at the tubing head varies. Therefore, for the first step calculation (Figure 4-2-1), point 1 is at the bottom-hole and the model will calculate the temperature at point 2. The above equation can be rearranged to solve for T_2, which is the parameter of concern.

$$T_2 = \frac{c_p T_1 + L_R c_p (h_1 - h_2)T_e - C_J c_p (p_1 - p_2) + (h_1 - h_2)\frac{g}{Jg_c} + \frac{1}{2}\frac{(v_1^2 - v_2^2)}{Jg_c}}{L_R c_p (h_1 - h_2) + c_p}$$

These values calculated at point 2 will be considered as point 1 for the next step. This procedure is repeated until the tubing head temperature is calculated. All the temperature points together form the temperature profile of the vertical model.

It is clear from Ramey's calculations that many fluid properties need to be calculated to find the temperature profile. These properties and other values depend largely on the calculations done in the Hagedorn and Brown method.

However, as noted earlier, the Hagedorn and Brown method relies on a known temperature profile. The following section outlines how the Hagedorn and Brown calculations and the Ramey's temperature calculations are solved together in an iterative way, such that it overcomes the need of having a known temperature profile or a known pressure profile.

4-3 Solution Method

There are two reasons why special attention needs to be paid to how the Hagedorn and Brown method and Ramey's method are being solved together. Firstly, as outlined above, both models require the other model to be solved first. Secondly, one of the calculation procedure starts at tubing head and ends at the bottom-hole location, while the other runs in the opposite direction. Both these issues are addressed in the solving process proposed in this thesis. The flowchart in Figure 4-3-1 describes the process.

The two models are run in series, and the calculations loop until the values converge. This eliminates the problem of having to know either the pressure or the temperature profile in full beforehand. Moreover, because the complete pressure and complete temperature profiles were calculated separately, it was possible to calculate in their respective directions without any problems. Figure 4-3-2 is a visual representation of the outputs in the iterative process.

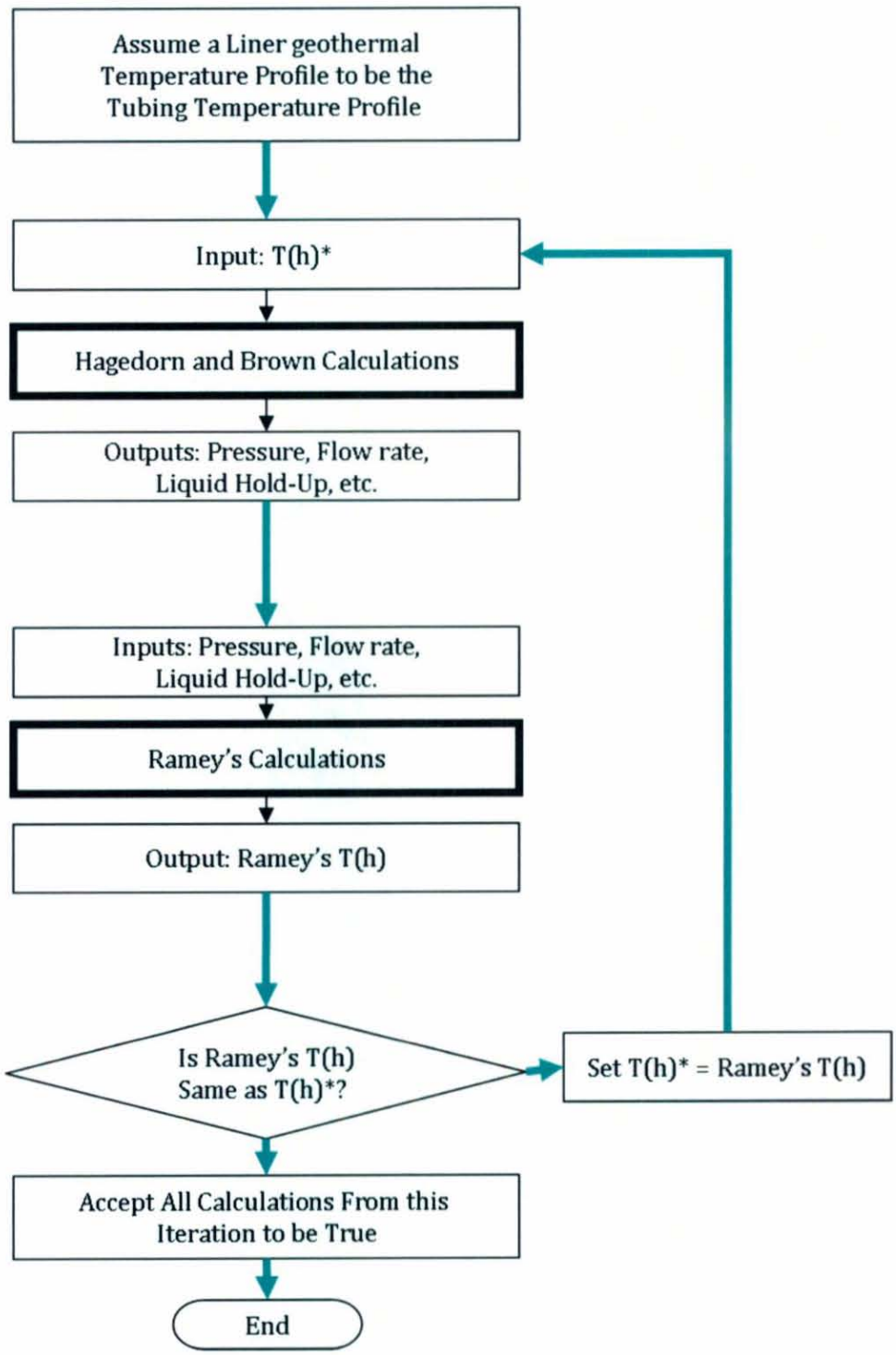


Figure 4-3-1 Solution Method for Vertical Model

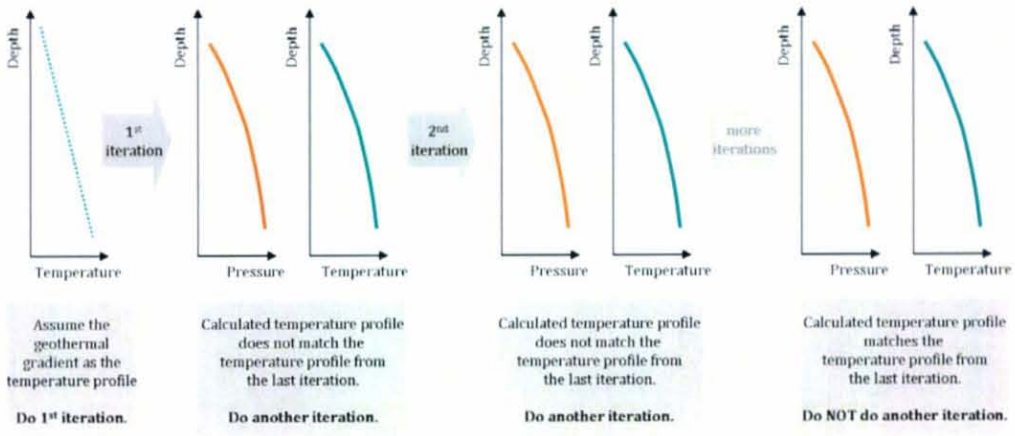


Figure 4-3-2 Results Progression of the Vertical Model

With the calculations done using the vertical models, it is possible to create lift curves (Figure 4-3-3), in the same way as IPR curves using the horizontal model. The single curve below is for a fixed value of tubing head pressure, gas-oil-ratio, water-oil-ratio, etc.

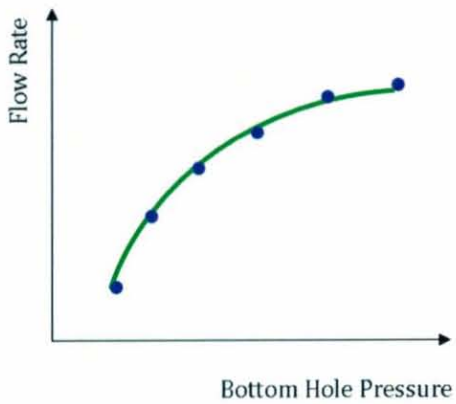


Figure 4-3-3 Lift Curve

The bottom-hole pressures are calculated for various operating flow rates, and these points are plotted on a graph to give the characteristic shape. As bottom hole pressure is increased, there is a greater pressure differential between the tubing head and bottom hole, which gives a greater driving force, allowing more fluids to flow up. A unit increase in the bottom-hole pressure is not match by a proportional increase in flow rate due to the higher frictional losses experienced at high flow rates. The curve in Figure 4-3-3 is obtained when the tubing head pressure and other parameters are fixed. A family of curves can be generated by varying these parameters, such as the tubing head pressure.

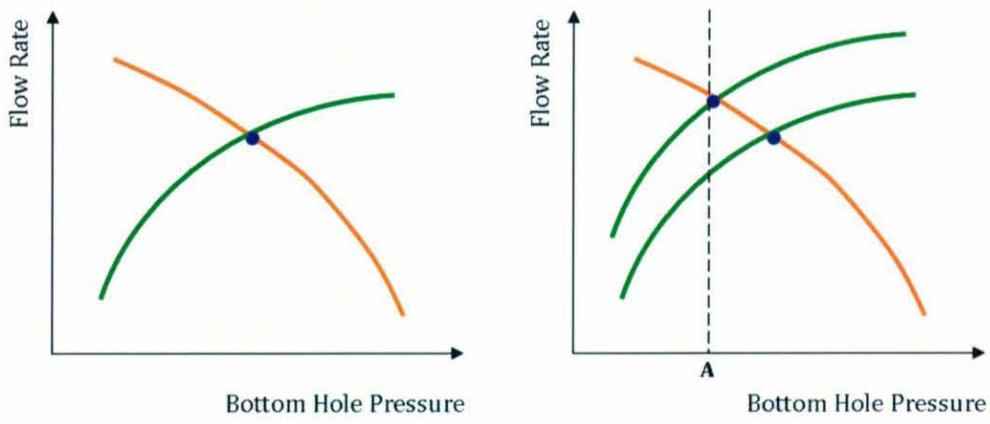


Figure 4-3-4 Well Operating Conditions

The well operating conditions are determined by allowing an IPR and a Lift Curve to intersect, as shown in Figure 4-3-4. Operating at the point of intersection signifies that the well is driving up the same amount of fluid that the well is able to collect from the reservoir. On the other hand, if the well was to be operated at

bottom-hole pressures at "A", a lot more fluid will be coming in from the reservoir than the well's ability to pull it out that that tubing head pressures. At that point, a lift curve using lower tubing head pressure is required to give the driving force to draw out more liquid up to the surface. This is the usual scenario if the production rate from that reservoir needs to be increased. The opposite applies if the production rate needs to be slowed down to ensure a higher recovery fraction.

Figure 4-3-4 demonstrates just one example of why the operating conditions may need to be changed, and in doing so, the program developed in this thesis would be a very important tool. However, there are many more reasons that could trigger the need to change the operation conditions. A further discussion about this topic is outlined in Reservoir Engineering books, such as by Johansen (2008).

CHAPTER 5

WAX MODEL

There are a number of professional software programs available to predict the wax precipitation in oil wells. However, all of them make use of deposition models that are based on wax diffusion. In such models, it is assumed that when the fluid temperatures reach and go below the WAT, the wax molecules diffuse to the tubing wall and become deposited as solids. This model may be applicable for surface or sub-sea pipelines, where the flow regime may be laminar. However, in recent times, attention had been drawn towards developing a model for the turbulent flow regime in the oil producing wells, in which it is not possible for diffusion to occur in radial direction. In this thesis, two different wax models are programmed with a complete well flow model for the first time and compared. These models assume that wax molecules crystallize at WAT, form gels that stick to the tubing, and harden over time. Both the models are not compositional, in an effort to keep the calculation simple enough for easy applicability.

The two models described in this thesis are by Zougari and Sopkow (2007) and Begatin et al. (2008). Both the models are based on the Ozawa (1971) kinetic model for crystal growth, which can be expressed as follows.

$$X_r(T,\lambda) = 1 - \exp\left(-\frac{K(T)}{\lambda^n}\right)$$

The above equation was expanded for non-isothermal conditions by Oliver and Calvert (1975), Turnbull and Spacpen (1978) and Hoffman (1985) for a single component.

$$X_r(T,\lambda) = 1 - \exp\left[-C_1 \Delta T \exp\left(-\frac{C_2}{T\Delta T^2} \right) \left(\frac{\Delta T}{\lambda} \right)^n \right]$$

For the case of multiple components, such as for crude oils, crystallization is considered as a cumulative process. Crystallization kinetics for m-components can be described by the following equation. This is also supported by some of the studies done by Hammami (1992).

$$X_r(T,\lambda) = 1 - \exp\left[-\sum_{i=1}^{m} C_{ia} \Delta T \exp\left(-\frac{\sum_{i=1}^{m} C_{ib}}{T\Delta T^2} \right) \left(\frac{\Delta T}{\lambda} \right)^n \right]$$

In the next sections, two different models are described, each applying the above equations in different manners. However, both these models calculate relative crystallinity, which basically describes the percentage of area under the microscope that is in some form of ordered or crystalline state. This is only the first step towards determining if a change of phase is expected to occur. Further calculations need to be developed to determine a relation between the relative crystallinity and phase change, as well as between phase change and deposition on the wall. One of the motivations for applying these developing wax models is to show the relevance of a good vertical temperature model to many applications.

5-1 Zougari and Sopkow Model

In their study, Zougari and Sopkow (2007) developed a kinetic model that was able to satisfactorily fit the experimental data from oils of five different wells, each from a different region of the world. They used the following equation for relative crystallinity, which is a normalized equation that has its beginnings from, but not exactly the same as, the Ozawa equation.

$$X_r(\theta,\varphi) = 1 - \exp\left(-\frac{K(\theta)}{\varphi^n}\right)$$

where,

$$\varphi = \frac{\lambda}{\lambda_{eff}}$$

$$K(\theta) = C_{1n}^{'} \exp\left(-\frac{C_{2n}^{'}}{(\theta + C_{3n}^{'})\theta^2}\right)\theta^{n+1}$$

$$\theta = \frac{T - T_{\min}}{T_{\max} - T_{\min}}$$

$$C_{1n}^{'} = C_1^{'}(T_{\max} - T_{\min})\left(\frac{T_{\max} - T_{\min}}{\lambda_{eff}}\right)^n$$

$$C_{2n}^{'} = \frac{C_2^{'}}{(T_{\max} - T_{\min})^3}$$

$$C_{3n}^{'} = \frac{T_{\min}}{(T_{\max} - T_{\min})^3}$$

In their study, the authors conducted laboratory testing by cooling stagnant (non-flowing) samples of crude oils at a known rate, and measuring the crystallinity. They fit their relative crystallinity equation to the experimental data by changing values of the constants. $T_{\max}$ is a temperature at which the waxes in the samples were still in

dissolved conditions; it was assumed to be a constant value of 80°C. T_{min} is the temperature at which crystal growth stops; this was also considered a constant with a value of -110°C. λ_{eff}, the effective cooling rate, was determined to be a constant of 0.1°C/min. The actual cooling rate, λ, was varied during experimentation. These recommended values were used to calculate relative crystallinity in this study as well. A value of 3, 1e-9 and -9.8e6 was used for the constants n, C_1' and C_2'.

5-2 Begatin et al. Model

In this model, Begatin et al. (2008) took a different approach to define the relative crystallinity function. They proposed an equation with just one fitting parameter C, which is the Ozawa constant. The single important variable that affects the crystallinity is $T_d(z)$ which is assumed to be the inside wall temperature. This is because the inside of the wall is the coldest surface that the well fluid comes into contact with, and hence provides a reason for nucleation and crystal growth to occur. The relation between $T_d(z)$ and relative crystallinity is shown in the equation below.

$$X_r(T) = 1 - \exp\left[\frac{-C[WAT(z) - T_d(z)]^2}{\left|\frac{\delta T_d(z)}{\delta z} v(z)\right|}\right]$$

The equation also assumes that velocity and wax appearance temperatures changes over the length of the horizontal well. This is to accommodate for the changing oil composition and tubing diameter respectively, as wax solids build up at various locations. Moreover, in the original Ozawa equation, there is a cooling rate term. In this model, the cooling rate is assumed to be related to the temperature gradient in the axial direction. Also, an Ozawa exponent of 1 is assumed to signify a rod morphology for the crystal structure, which is a phenomenon also noted by Zougari and Sopkow (2007). Holder and Winkler (1965) also observed that wax solidified into get only after a certain level of crystallinity was reached. This value was quoted to be as low as 0.5% for certain crude oils. Therefore, Begatin et al. (2008) proposed that wax deposition could be considered a possibility only when the relative crystallinity is calculated to be a value greater than 0.5%.

Therefore, this theoretical model can be applied with the help of a good temperature and flow model for the vertical well, as well as good experimental data for $WAT(z)$. Since this study aims to check the applicability of these models, only a simplified equation is used with assumed values for experimental data. $WAT(z)$ is assumed constant throughout the well, $v(z)$ is assumed to be the values calculated by the vertical flow model, and $T_d(z)$ is assumed to be same as the fluid temperature for simplicity purposes in this study.

CHAPTER 6

RESULTS

The previous chapters describe the mathematical basis of the models used in this study. In this chapter, the results obtained from these models are shown. At first, the results obtained from running only the horizontal model are outlined. This includes running simulations for a generic well completion and producing an IPR for the well. The results from running a complex well completion are also shown. In the next section, various profiles and lift curves are obtained from executing the vertical model alone are shown. The horizontal and vertical models were then combined to produce the operating parameters for a production well. A section is also dedicated to comparing the temperature profiles from the vertical model to the approximate, analytical temperature profiles commonly used in industry. This leads into the last section, where the temperature profile is used to calculate the wax crystallinity in the fluid.

The horizontal and vertical models used in this thesis produce results that are reservoir and fluid specific. There is provision for changing reservoir properties by making changes to the input files. For fluid properties, a table was generated using PVTsim™ (Calsep), which was then embedded as an input file to the program. This table could be regenerated for different fluids to produce fluid specific results. One of the pre-existing fluid compositions on PVTsim™ (Calsep)

was used to generate the results in this section. This fluid has a bubble point pressure of 265 bars. Further details could be seen in the input files outlined in Appendix A. Some of these detailed information, such as the length of the completion, can also be deduced from the values shown on the axes of the graphs. The following table shows the fluid composition.

Table 6-0-1 Fluid Composition

Component	Mol %	Mol wt (g/mol)
N2	0.56	28.014
CO2	3.55	44.01
C1	45.34	16.043
C2	5.48	30.07
C3	3.7	44.097
iC4	0.7	58.124
nC4	1.65	58.124
iC5	0.73	72.151
nC5	0.87	72.151
C6	1.33	86.178
C7	2.73	89.9
C8	3.26	103.2
C9	2.14	117.7
C10	1.94	133
C11	1.62	147
C12	1.47	160
C13	1.69	172
C14	1.62	186
C15	1.59	200
C16	1.3	213
C17	1.11	233
C18	1.26	247
C19	1.07	258
C20	13.32	421

6-1 Horizontal Model Results

The horizontal model applied in this thesis determines the flow rate, pressure and oil fraction profiles along the length of a horizontal well, for a maximum of 2-fluid phases. The simplest case that could be run in this simulation is for a single phase fluid using the generic completion shown in Figure 3-1-1. Following are the profiles obtained when the generic completion is operated above the bubble point pressure.

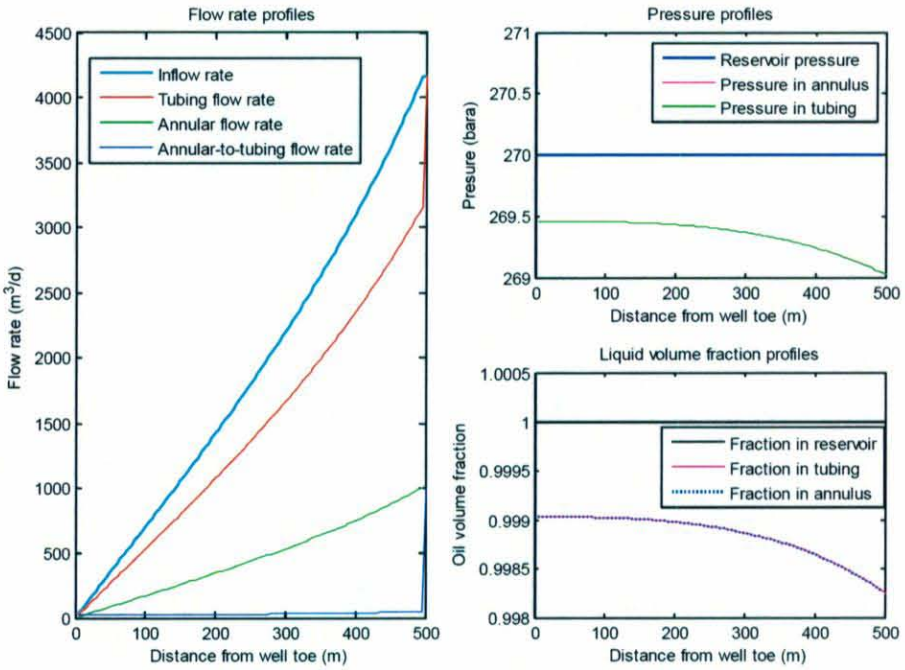


Figure 6-1-1 Generic Completion Operated Above the Bubble Point Pressure

The flow rate profiles show that inflow rate increases from the well toe to the well heel. This is because more inflow locations become available, which allows

more fluids to flow in. This is the main advantage of horizontal wells. All the fluid flowing into the completion first goes to the annular section. However, the annular section holds far less fluid at any distance than the inflow amount. This is because the fluids get transferred instantaneously to the tubing and to the next annular node. At any point in the completion, the total fluid in the annulus and tubing will equal the inflow rate at that location, as set by the mass conservation equations. It can be seen that the annular-to-tubing flow rate is constant throughout the length of the well. This is because the input parameters are set in a way such that there is equal distribution of slots along the body of the tubing, which facilitate this type of flow. Towards the heel, there is a big spike in the annular-to-tubing flow rate (which is matched by the spike in the tubing flow rate). This is because at the last annular node near the heel, all the fluid from the annulus flows into the tubing. Therefore, the generated flow rate profiles match the expected behaviour.

The above flow rate was achieved due to a pressure drawdown from the reservoir to the well. As indicated by the pressure profile, the reservoir pressure was assumed to be at 270 bars, and the bottom-hole pressure was set to 269 bars. As can be seen, a pressure differential exists throughout the length of the well, which is the driving force for the fluid to flow into the well. The pressure is lower at the heel of the well due to frictional pressure losses that occur over the length of the well. The tubing was assumed to be a smooth pipe in this case. Depending on the restrictions of flow in the completion, this pressure drop along the length of the

well would change. Therefore, the calculated pressure profile matches the expected behaviour as well.

The oil fraction profiles have a value of approximately 1, because the fluid is always subjected to pressures above the bubble point pressure. A slight drop in oil fraction is calculated due to pressure drawdown calculations.

In the next example, the same well is operated at pressures below the bubble point pressure.

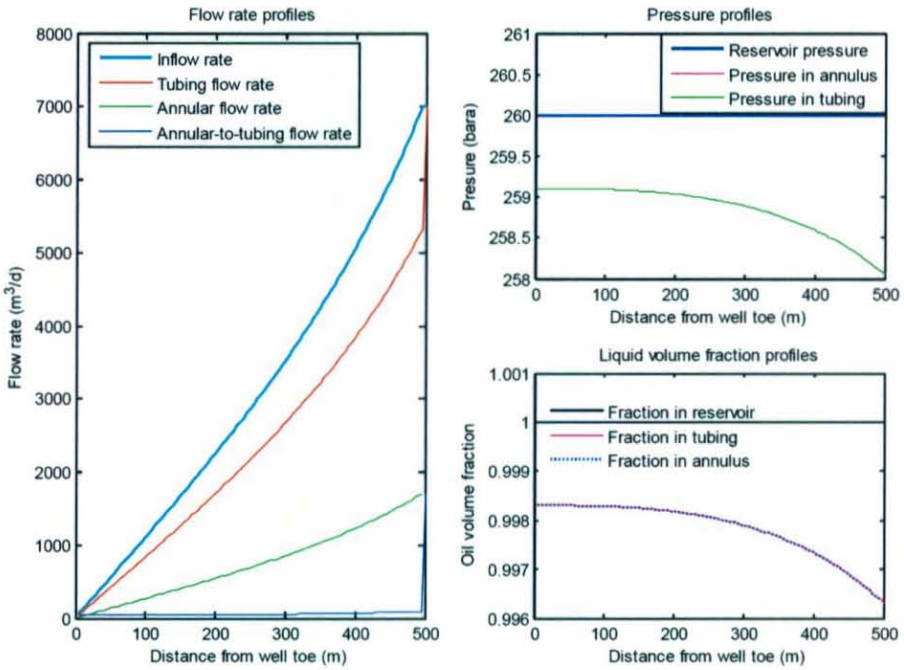


Figure 6-1-2 Generic Completion Operated Below the Bubble Point Pressure

As can be seen from the pressure profile in Figure 6-1-2, the reservoir pressure is assumed to be 260 bars, and the bottom-hole pressure was set to 258 bars. A similar pressure profile is obtained as the first example, because the same completion was utilized. The corresponding oil fraction profile shows a much greater amount of gas being produced than the last example because the pressure is below the bubble point pressure. However, because of lack of data, it was assumed that the oil fraction in the reservoir was 1. In reality, this value is expected to be less than 1 when the reservoir pressure is below bubble point pressure, and hence more gas is expected to be produced. In real applications, the reservoir oil fraction value can be obtained by PVT sampling and the data can be input into this program to give a realistic picture. The shape of the liquid fraction profile is also dependent on the pressure drawdown. A higher pressure drawdown would result in more gas liberation.

It can also be seen in this example that the rate of production is higher than that of the first example. This is mostly due to the fact that this example has a higher pressure draw down of 2 bars (compared to 1 bar in the first example).

In this next example, the profiles are calculated for a completion for which the last 250m section near the heel is packed off. A grid representation of this type of completion can be seen in Figure 3-1-4 (page 25). This completion has two locations where all the fluid in the annulus gets transferred into the tubing. This phenomenon is clearly represented by the flow rate profiles in Figure 6-1-3. At

the well heel and at 250m from the well heel, there are large jumps in the tubing flow rate because of the placement of the packers. The tubing flow rate remains constant for the last 250m section, since there is no annular-to-tubing flow.

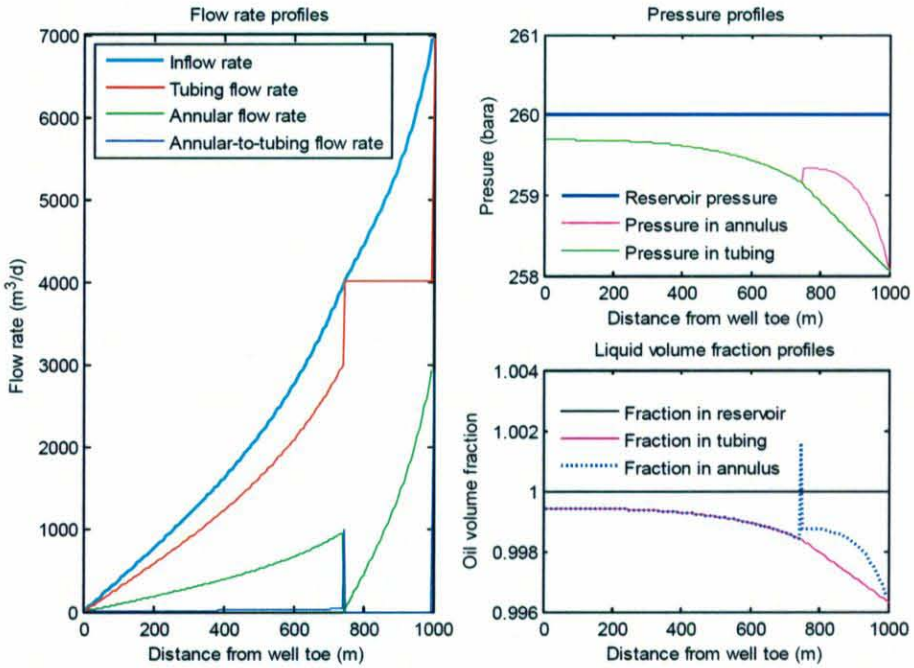


Figure 6-1-3 250m Packed-Off Completion

The last 250m of the annulus has higher pressure, because of the fluid build up that is only able to flow into the tubing from one location. This provides a more uniform pressure differential (between the reservoir and completion) throughout the length of the well. This is desired in the situations where a gas cap or water breakthrough is imminent. Similarly, different types of completions, such as the stinger completion in Figure 3-1-3 (page 24), can be utilized to tackle various production concerns. This horizontal flow model is able to provide the profiles

specific to the completion type, which is a very desirable attribute. It gives the reservoir engineer the opportunity to easily calculate the results of many possible solutions before investing into it.

As described in Chapters 3 and 4, it is necessary to make an IPR plot to determine the optimal operating pressure and flow rate for a given reservoir. Figure 6-1-4 is the IPR plot generated for the generic completion using this horizontal flow model. The IPR has the characteristic slant and curvature as expected – as the bottom-hole pressure is decreased (i.e. increase in pressure differential with the reservoir), more fluids flow into the well. Frictional effects at high flow rates explain the curvature of the plot. Section 6-3 shows how this IPR plot could be used in conjunction with lift curves to provide the operating conditions.

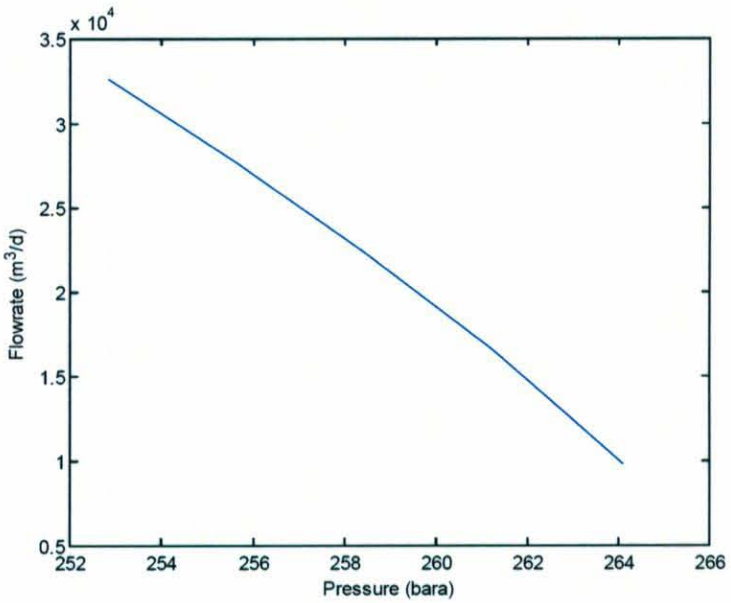


Figure 6-1-4 Inflow Performance Curve of Generic Completion

6-2 Vertical Model Results

The vertical flow model developed in this thesis models the flow of fluid from the well heel to the surface. During this voyage, the fluid experiences a much greater pressure and temperature drop, causing drastic changes in liquid hold-up (i.e. liquid fraction) and flow regime. The combination of Ramey's model and the Hagedorn and Brown model calculates temperature, pressure, liquid hold-up and flow rate profiles as shown in Figure 6-2-1 in the respective order.

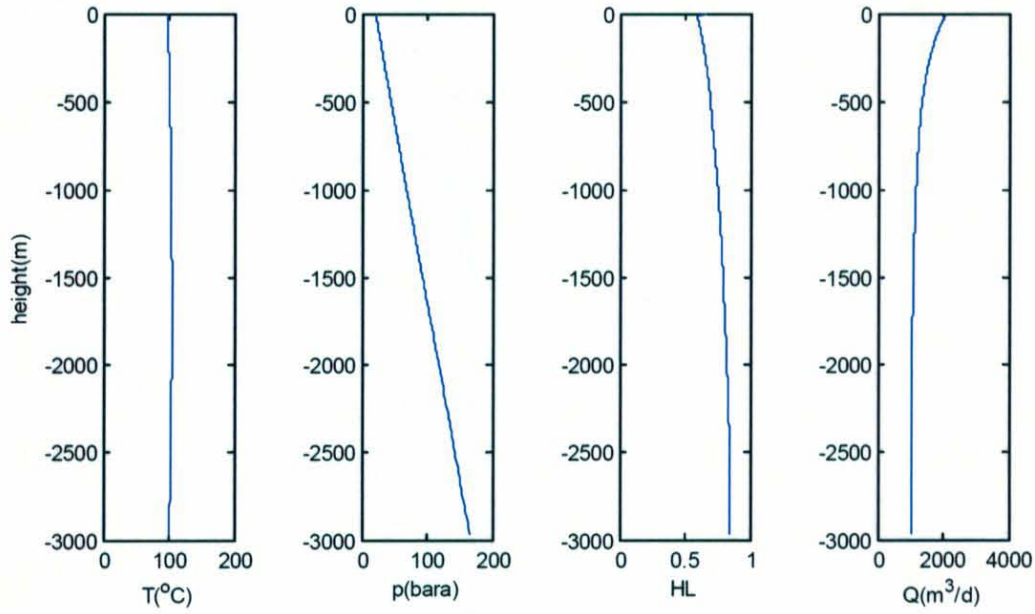


Figure 6-2-1 Profiles Calculated Using the Vertical Flow Model

Figure 6-2-1 was the results obtained when the selected fluid (Table 6-0-1, page 64) was run in the simulation at very high bottom-hole flow rate of about 1000 m^3/d. This was done because the flow rates calculated in the horizontal model

also had very high values, which is characteristic of horizontal models. The temperature profile was calculated starting with the bottom-hole temperature (i.e. reservoir temperature) as the boundary condition and moving upwards. It can be seen that the temperature slightly increases as it flows upwards, and the fluid temperature at the tubing head was calculated to be just slightly below the bottom-hole temperature. This is because the high fluid flow rate causes frictional heating in the piping. It will be shown in Section 6-3 how a lower flow rate produces a different type of temperature profile.

The pressure profile can be seen to change from over 160 bars at the bottom-hole to about 30 bars at the tubing head. This high pressure differential provides the force needed to naturally pull up such a big amount of fluid from the bottom-hole against gravity. It can be seen with the use of a ruler that the pressure profile is not a straight line; it is curved to account for the momentum loss due to friction. The loss of pressure causes gas to be liberated from the liquid. This is indicated by liquid hold up profile. The liberation of gas causes the flow rate to sharply increase, due to the low density of the gaseous phase.

Thus overall, the model results are in par with what is expected. These results were calculated through an iterative method. Figure 6-2-2 shows the iterative process that took place. It can be seen that the model converged in 2 iterations.

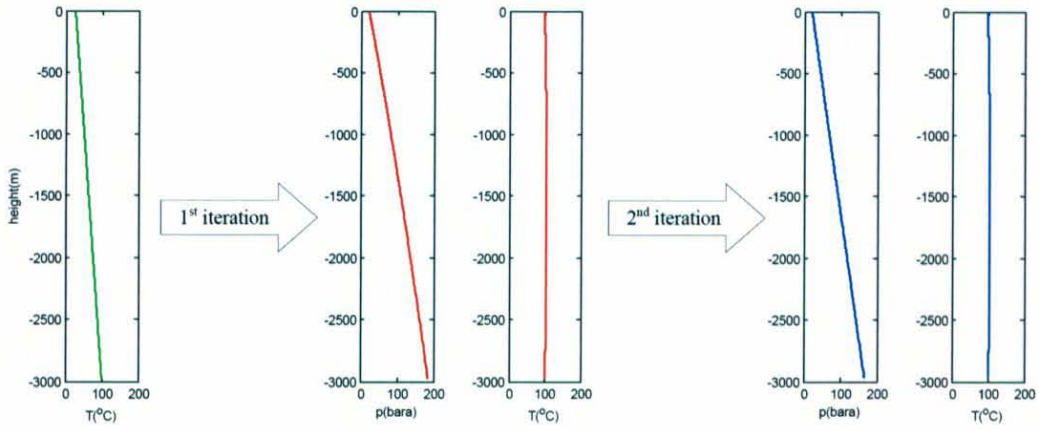


Figure 6-2-2 Calculation Progression for Vertical Model

Figure 6-2-2(a) shows the initial assumed temperature profile. This is the same as the geothermal profile, which was used to calculate the pressure profile and a new temperature profile using the pressure values. This is shown in Figure 6-2-2(b). The program then checked if the temperature profiles in Figure 6-2-2(a) matched the one in Figure 6-2-2(b). Since the temperature profiles are very different, the program went forward with the 2nd iteration. The pressure profile in Figure 6-2-2(c) was calculated using the temperature profile in Figure 6-2-2(b). At the same time, a new temperature profile is calculated in this 2nd iteration with the new pressure profiles. The computer then checked again to see if the difference between the temperature profiles in Figure 6-2-2(b) and Figure 6-2-2(c) were within the allowed tolerance. In this case, it was; this can also be seen from the similar shapes of the temperature profiles in the figure. Therefore, the program stops iterating after this step.

The vertical model is also used to plot the lift curves for the system. This was done by running the simulation for different values of bottom-hole flow rate. The model produced the corresponding values for bottom-hole pressures. The plot of these flow rate and pressure values is the lift curve, as shown in Figure 6-2-3. The curve has the characteristic shape of a lift curve.

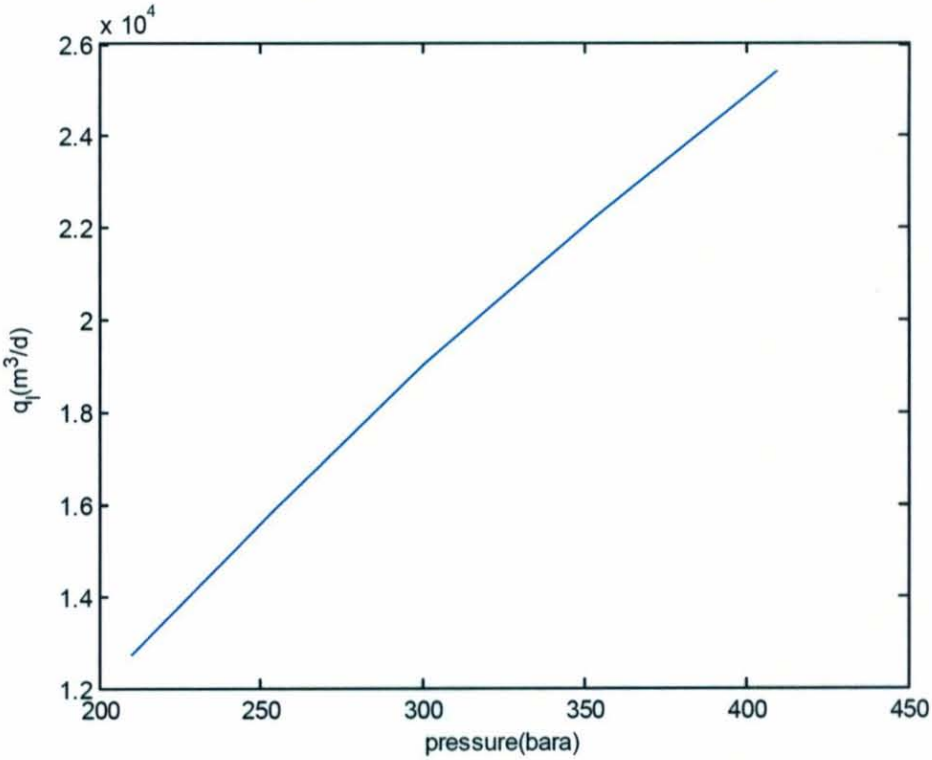


Figure 6-2-3 Lift Curve from Vertical Model

This lift curve, together with the IPR from the previous section, can be used to calculate the operating conditions. This is shown in the next section.

6-3 Determining Operating Parameters

In the previous sections, both an IPR and a Lift Curve were produced. The point of intersection of the IPR and Lift Curve dictates the operating parameters of the well. This is because, at this point, both the fluid coming into the bottom-hole (defined by the IPR) equals the fluid that is able to flow out of the well (defined by the Lift Curve). Figure 6-3-1 shows the plot when both Figure 6-1-4 (IPR) and Figure 6-2-3 (Lift Curve for a tubing head pressure of about 20 bars and a GLR of 300) are drawn on the same axis. It shows that if the well head is operated at 20 bars, about 16 000 m^3/day (at bottom-hole conditions) of fluid can be produced.

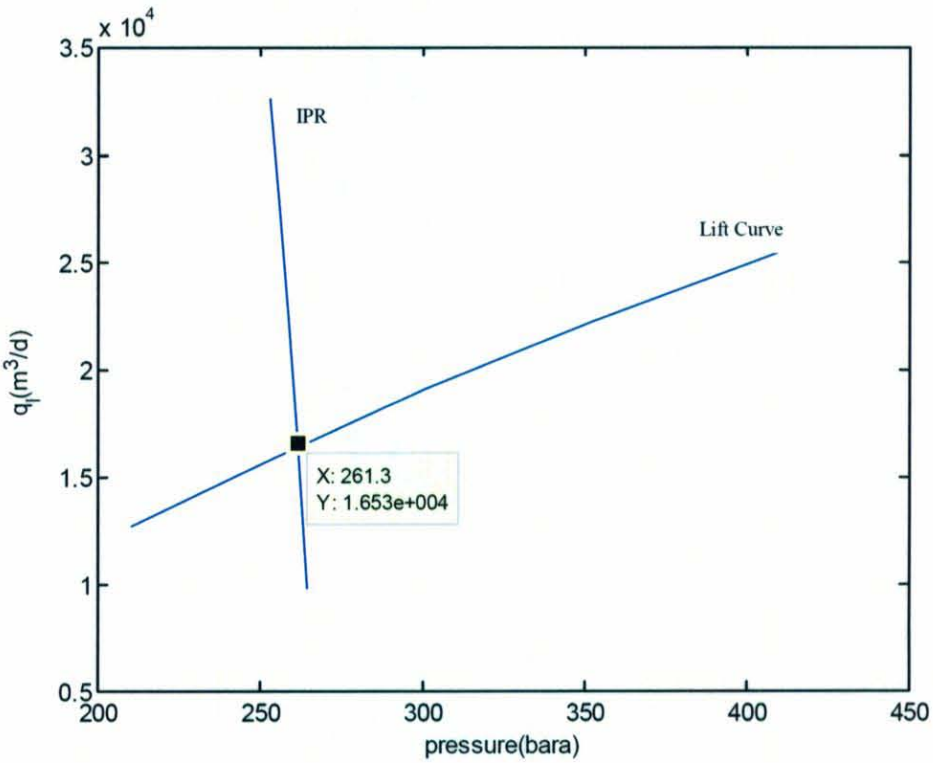


Figure 6-3-1 Operating Conditions

If for some reason it is desired to reduce the production rate, the tubing head pressure could be increased. Figure 6-3-2 shows the shift in Lift Curve when the tubing head pressure is changed from 20 bars to about 35 bars. This gives a new intersection point with the IRP, where the production rate at the bottom-hole conditions will be lower. Various other factors affect the Lift Curves, such as water cut, gas-liquid ratio, tubing radius, etc. These values could also be changed in the program developed in this thesis to give different families of lift curves. By having such quick simulation tools available to the reservoir engineer, it is possible to investigate various possible operating conditions and their effects.

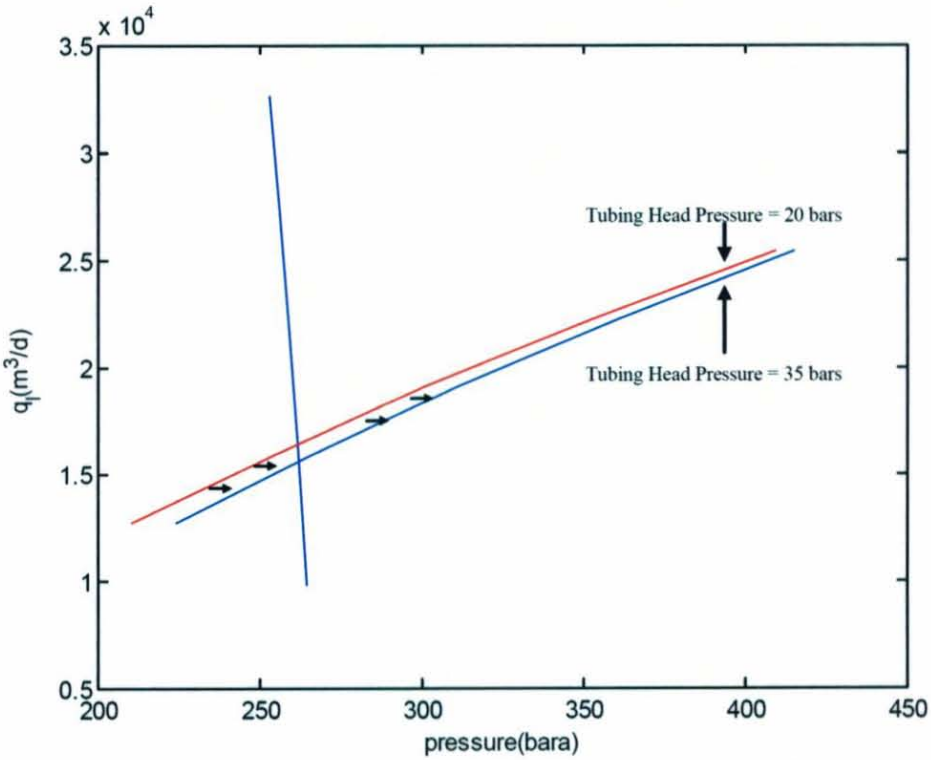


Figure 6-3-2 Operating Conditions at Different Tubing-Head Pressures

6-4 Temperature Profile Comparison

The temperature profile is an important factor for certain flow assurance issues. In this thesis, the direct relationship between the fluid temperature and wax crystallization has been discussed. Therefore, there is a need to have a good temperature model. A substantial amount of effort was invested to achieve this by applying a detailed temperature model for two phase flow, where only the reservoir temperature is known. This allows this model to be used for prediction purposes before the well is drilled. In this section, a few findings from running the temperature model are discussed.

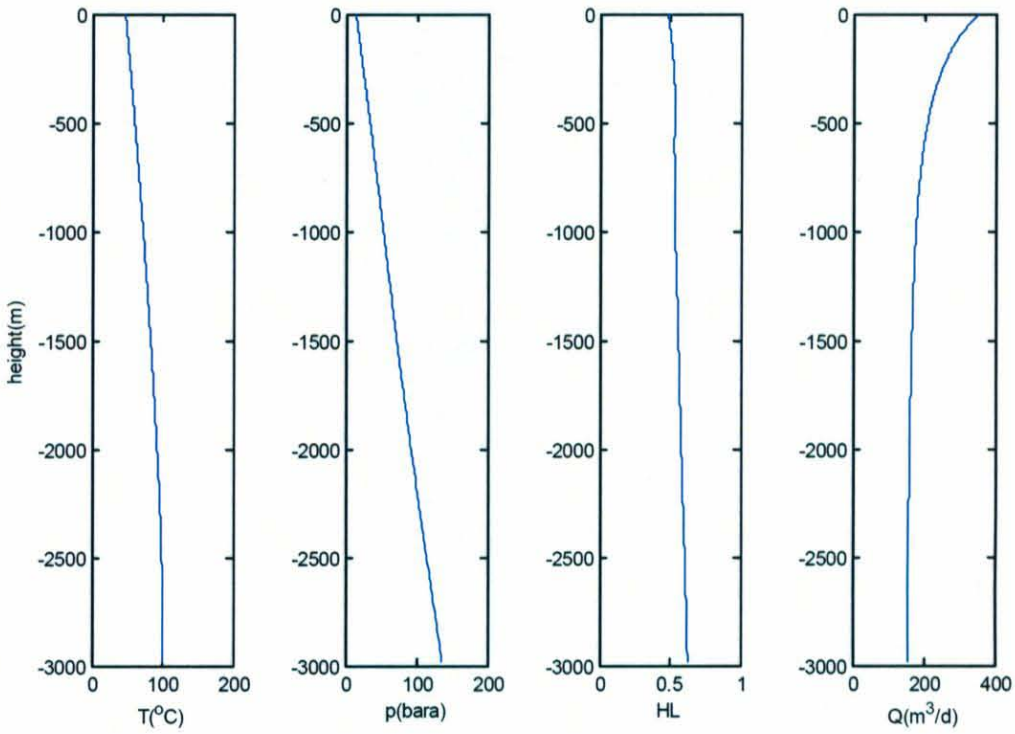


Figure 6-4-1 Vertical Model for a Low Flow Case

Figure 6-4-1 shows the profiles from a vertical model with bottom-hole flow rate of about 150m^3/d (much lower than the example from Section 6-2, page 71). Such flow rates are decent production rates for vertical wells. It can be seen that the tubing head temperature can very easily be as low as 50°C. At such temperatures, it is not unlikely to have crossed the WAT value for the fluid. In the next section, this temperature profile will be used to perform calculations regarding wax crystallinity.

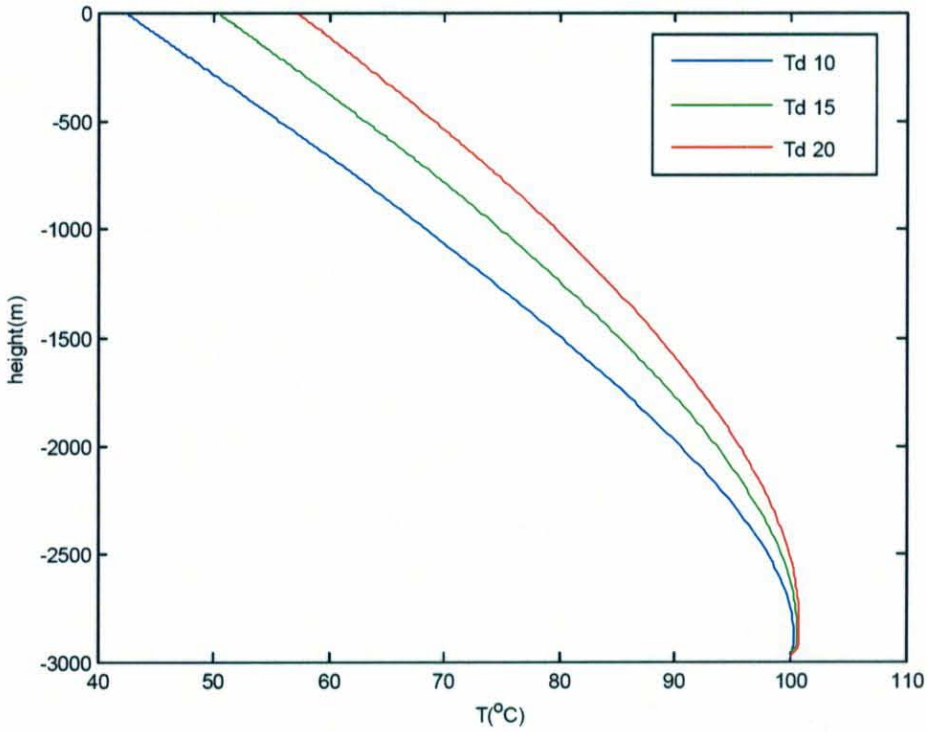


Figure 6-4-2 Wellbore Temperature Profiles over Time

There are certain advantages of using the temperature model used in this thesis. It acknowledges the fact that wellbore fluid temperature changes with time. Figure 6-4-2 shows how the temperature profile is different at different values of dimensionless time (Td). As the temperature of the surrounding rock rise over time (due to heat transfer from the fluid in the well), the temperature differential between the fluid and the rock decreases. Therefore, less heat is lost from the fluid, which results in the fluid to be warmer over time. By using this function, the reservoir engineers will be able to determine if certain concerns are expected to affect only on the short run or will it affect in the long run.

While doing literature review regarding wax deposition models, it was seen that an analytical model was used most frequently as the basis of the wax deposition model. This model is a single-phase simplification of Ramey's model. The motivation to select the analytical model was that it performed well for single-phase situations. The model can be written as follows.

$$T(x) = T_{amb} + (T_{in} - T_{amb}) \exp\left(\frac{-\pi D U_{tot}}{c_p q_L} x\right)$$

As can be seen, for this analytical model, both the tubing head and bottom-hole temperatures need to be known. Moreover, a constant value for c_p has to be used. In the model used in this thesis, all the properties used were evaluated for the temperature and pressure values at each specific location. Therefore, the analytical model is limited in its ability to predict a fluid specific profile before production starts.

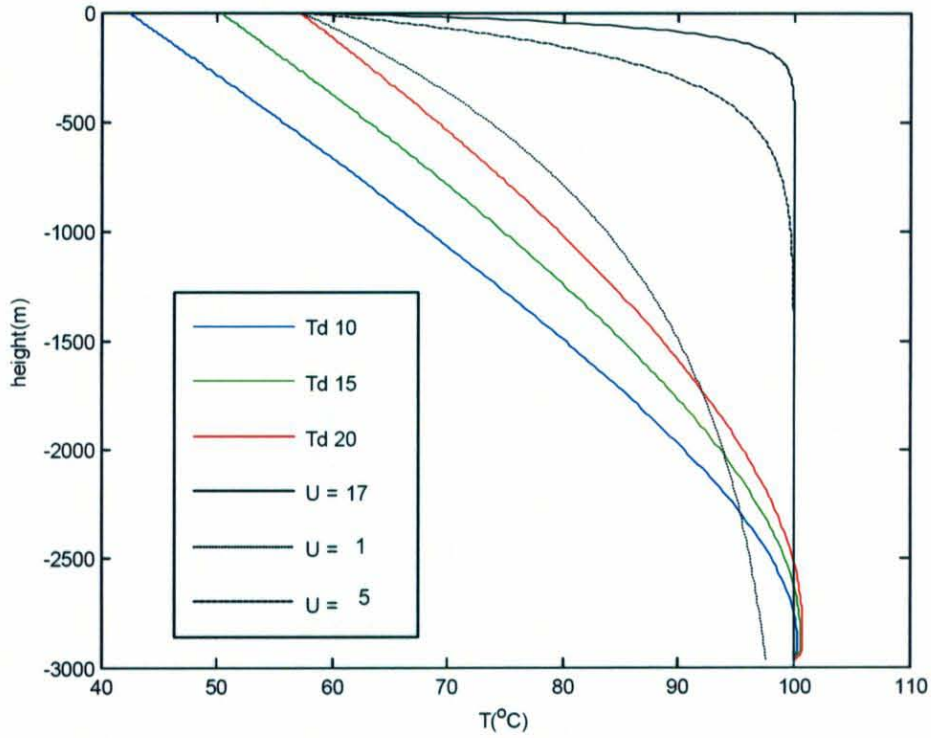


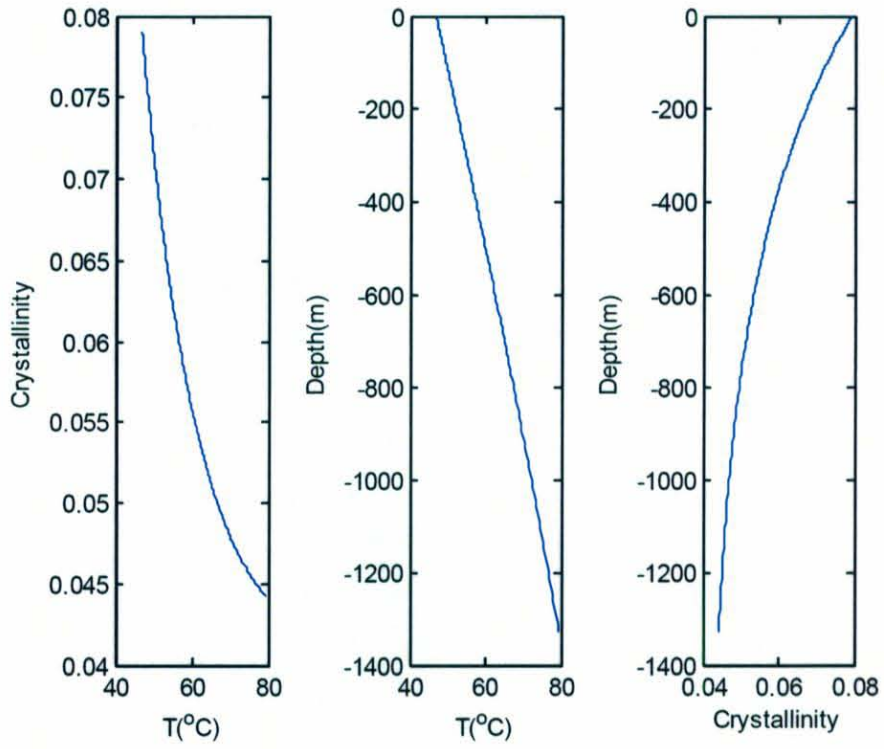
Figure 6-4-3 Numerical vs. Analytical Temperature Profiles

Figure 6-4-3 shows how this analytical model compares with the numerical model. The temperature profiles plotted in colour are the profiles produced using the numerical method coded in this thesis at different values of dimensional time. All these three profiles were plotted for an overall heat transfer (U) value of about 17 Btu/hr/ft^2/°F, as calculated by Dawkrajai et al. (2005). The temperature profiles in black colour are generated using the analytical solution described in this section (with the same values for diameter and overall heat transfer coefficient). It can be seen that the analytical model produced very different profiles compared to the numerical model. The analytical profile produces very steep changes for a

realistic overall heat transfer coefficient value. Only at very unrealistically low values of the overall heat transfer coefficient, it is possible to see a curvature similar to the numerical model. In most oil wells, it is common to have a two phase fluid at the tubing head. Hence, a numerical model, such as the one implemented in this thesis, may be worthwhile developing and investigating further into for accurate prediction. This is because the choice of temperature model will dictate the results of the wax model. For instance, if the fluid in Figure 6-4-3 had a WAT value of 80°C, the analytical temperature model would conclude that wax deposition is not a concern. However, the numerical temperature model would flag it as a concern for at least half of the depth of the well.

6-5 Wax Model Results

The temperature profiles generated in the vertical model could be applied to wax deposition models, since low temperature is the driving for wax crystallization. Two different wax crystallization models were discussed in this thesis. Although both the models were derived from the same concept, the final models have a number of differences. Figure 6-5-1 and Figure 6-5-2 show the results from applying these models. Both the models have different fitting parameters. The purpose of these parameters is to match the model to data. However, since field data were not available, the curve fitting was not possible. The graphs generated in this section only give an impression of the performance of these models. The models were then applied to the case generated in Figure 6-4-1 on page 77.



(a) (b) (c)

Figure 6-5-1 Zougari and Sopkow Model

Figure 6-5-1(a) shows the plot generated using the Zougari and Sopkow (2007) wax crystallization model. It shows that the wax crystallinity goes up and temperature goes down. It was assumed that the WAT was 80°C for this fluid, and therefore, the graph is only relevant for temperatures below 80°C. Figure 6-5-1(b) shows the temperature profile calculated in the vertical model in Section 6-4. For this temperature profile, the wax crystallinity profile in the well (calculated using the Zougari and Sopkow Model) is shown in Figure 6-5-1(c). It can be concluded that the wax crystallization and deposition is possible only for the first

1300 m of the well, since further below, the fluid temperature is above the WAT. It should be noted that the crystallinity values in the graphs are highly dependent on the choice of value for the fitting constants. Thus the crystallinity values, which would determine the amount of solid wax available to deposition on the tubing wall, would be relevant only if some fluid specific data were available.

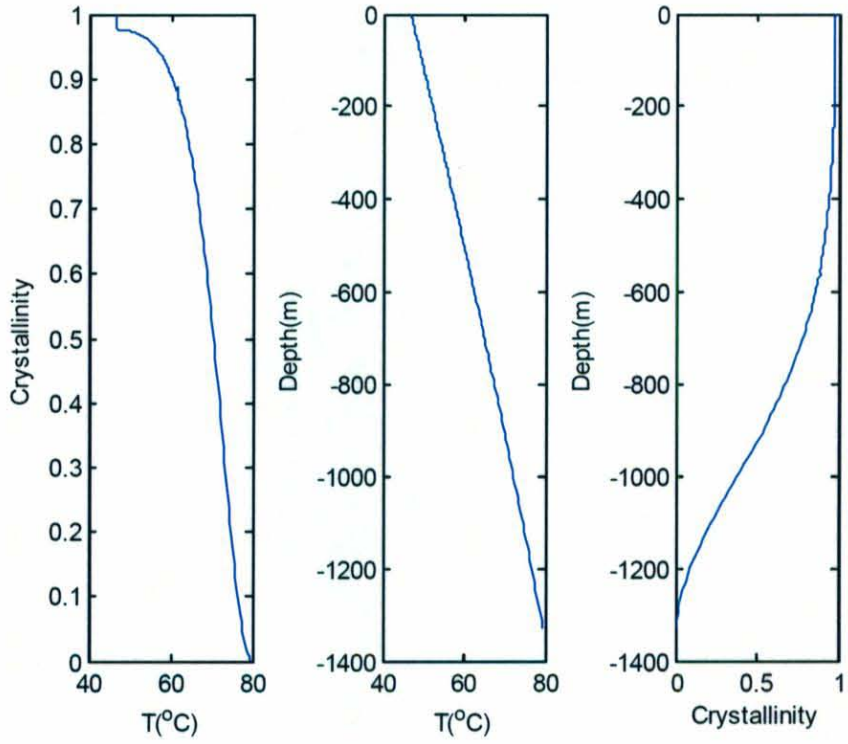


Figure 6-5-2 Begatin et al. Model

Similarly, Figure 6-5-2 was generated using the Begatin et al. (2008) wax crystallization model. The crystallinity model itself produces a profile (Figure 6-5-

2(a)) much different from the Zougari and Sopkow model. The result of this model is also specific to the well compared to the Zougari and Sopkow model. This is because this model utilizes the axial temperature gradient and velocity values calculated in the vertical flow model to produce the crystallinity profile, as opposed to the radial cooling rate in the Zougari and Sopkow model. Figure 6-5-2(b) is the temperature profile generated for the well using the vertical model. The wax crystallinity profile for this well is shown in Figure 6-5-2(c). The shape of this profile is much different from the shape generated using the Zougari and Sopkow model, hence highlighting the differences between the models. However, like the results of the Zougari and Sopkow model, the crystallinity values itself (shown in this figure) are only a product of the fitting constant used. In order to apply the model in real life situation, some experimentation is necessary to evaluate the value of this constant.

The need for such experimentation is a drawback, since it prevents the user from using the model very easily. However, some level of experimentation is necessary to be able to apply such phase-change models. Other models often require a thorough compositional analysis, and the availability of properties that are very difficult to evaluate. Therefore, relatively, the crystallization models described in this thesis are not very difficult to apply.

CHAPTER 7

SUMMARY

This research explored a few opportunities of improving the simulations available to reservoir engineers in the oil and gas industry. Today, the industry is paying a lot more to explore and produce crude petroleum from extreme locations. Therefore, it is of more interest today than ever before to use mathematical models and computer software packages as a cheap and highly effective tool to plan for petroleum production.

Three very specific models were used in this thesis. Firstly, Thanyamanta's (2009) IPR model for horizontal wells was improved. Although the model itself has made a big contribution to provide a comprehensive 2-phase model for horizontal wells, there was room to add to its abilities. It was made easy to input specific fluid properties into the program. In this way, IPR curves could be generated for horizontal wells with not only specific completion designs, and with regional reservoir conditions, but also with very specific fluid properties. This could not be achieved using property correlations (as recommended by Liu (2009)). With this addition, the model is able to produce accurate profiles for flow rate, pressure drop and liquid fractions over the length of the well, and as well as IPR plots. Such information is a must for good well design and operation plan.

The second model used was the vertical flow model by Hagedorn and Brown (1965) in conjunction with the temperature model developed by Ramey (1961). In this research work, the two independent models were intertwined in a way such that their sequential and iterative solution would provide the temperature, pressure, flow rate and liquid fraction profiles. These are important parameters to consider during the design of this part of the well, when the fluid moves from bottom-hole to tubing head. A big change in pressure, temperature and velocities are often encountered during this journey. The model is also able to produce lift curves. Lift curves, together with the IPR plots, provide the optimum operating conditions for the well. This, in turn, dictates the design of this portion of the well.

As can be seen, both the horizontal and vertical models play an important role in the design and the day to day planning, scoping and operation of an oil well unit. Other than working as an everyday crucial tool for the industry, these horizontal and vertical models also play an important basis for further research work and risk assessment. This was illustrated by using the temperature profile generated in the vertical model in this thesis as the basis for two different wax crystallization models. These wax models are still in the process of being fully developed. By providing accurate temperature and velocity profiles from the vertical flow model, it gives these preliminary wax models a platform to be evaluated for their performance in oil field situations, which can provide important clues about how the models needs to be further developed and direct new areas of experimentation. It was shown how a widely used analytical

temperature model provides very different profiles compared to the ones generated in this thesis.

The results generated from the models applied in this thesis were coherent and logical. A few trends could be confirmed from the results, as well as from doing the work of creating the program. These are outlined in the next section, together with recommendations for further study on the research carried out in this thesis. The last section of this thesis discusses the novelty of the research work done in this study.

7-1 Findings and Recommendations

While working with the horizontal model on a separate project, the importance of applying appropriate fluid properties was discovered. The model produced distorted results if an error was made with providing the correct values. This was one of the motivations to make changes to Thanyamanta's model (2009) to allow easy switch over from one fluid to another. It is recommended that this mechanism be applied to the 3-phase model developed by Liu (2009).

The use of different horizontal completions in Thanyamanta's model (2009) was one of the strengths of the model. In this research work, only one complex completion was further investigated. Further investigation of other such completions is encouraged. It would be very useful if the user is able to switch

from one completion to another much easily. The current method is not very user friendly, and hence leaves much room for the user to make errors while setting up such completions.

It would also be useful to tie the horizontal model with a reservoir streamline model, such as the one recommended by Skinner (in progress). This would not only make the IPR more accurate, it would also give the user a visual representation of situations such as gas breakout, water coning, etc. Moreover, by changing the calculation grid such that reservoir nodes were able to communicate between each other, it could be possible to give the program the ability to determine the direction of flow. This would make the simulator more powerful. When using the calculation grid for horizontal well used in this thesis, the flow directions had to be predetermined by the user.

It is also recommended that the horizontal model be compared with field data to evaluate if there is room for improvement. Since the use of temperature sensors are becoming common practice on horizontal wells, the temperature model proposed by Thanyamanta (2009) could also be compared and evaluated.

The current horizontal model allows the user to input different rock and completion properties for each segment described using the model grid. It would be useful to extend this property to allow segments to deviate from the horizontal placement by few degrees, since this is a common situation in real wells. This

would allow the model to be more accurate in terms of producing data for the specific wells. It is known that these slight deviations in well placement have a substantial effect on the temperature profile of the horizontal well. It would be interesting to evaluate if the produces a prominent effect on the flow and pressure profiles as well.

It was noticed that the productivity of the horizontal well would dictate the design of the tubing in the vertical well. This is because the high flow rates of horizontal wells require wider vertical wells for lifting the fluid out of the well. It was also noticed that high flow rate in vertical flow caused fluid to heat up due to friction. It would be great if these data generated by the vertical model could be compared with field data to evaluate its performance. It would be interesting to further study to see if wax deposition is minimal in high production wells, since this study suggests that the high flow rate has the tendency to heat up the fluid as it travels up the pipe, as opposed to cooling it. Moreover, expanding the vertical model to include flow in risers would be very useful, since a big temperature drop can be experienced there.

The wax models applied in this thesis were under development. It is recommended that these models be first applied by fitting them with field or experimental data to evaluate their performance and aid their development. Use of high temperature and high pressure flow loops could play a critical role to provide useful data. It is also recommended to apply the wax models when they

are complete, such that the location and amount of wax deposition can be predicted. Lindeloff and Krejbjerg (2002) also concluded that it is important to perform transient analysis (with respect to fluid temperature profile in the vertical section) for wax deposition. This may also be further investigated using the models developed in this thesis.

7-2 Novelty of Research

The work done in this research work comprised of the application of various existing models. The novelty of the work lies in the improvements that were made and how these models were applied.

The use of fluid property tables to interpolate for the right data point was applied for the first time in the horizontal model. This was a favourable improvement, both in terms of results generated and as well as increasing the versatility and usability of the program.

The complete Hagedorn and Brown method (1965) and the complete Ramey's model (1961) were applied together in the proposed complex, yet relevant and appropriate solution process for the first time. The use of this vertical model and Thanyamanta's horizontal model (2009) to determine operating conditions using IPR and lift curve intersect was also a new addition.

The wax crystallinity models applied in this research were only applied using analytical temperature models so far. In this research, these models were applied using data from the integrated horizontal and vertical model developed earlier. The wax deposition software packages that are available today all use diffusion as the driving force for wax deposition, while the models applied in this study consider a kinetic limited crystallization process as the main mechanism for wax deposition.

Therefore, a substantial amount of new research work has been presented in this thesis. However, there is always room for improvement, some of which are outlined in the previous section. It is hoped that the work in this research would provide some new, useful information, as well as encourage further study in this subject.

REFERENCES

Ali, S. (1981) A Comprehensive Wellbore Stream/Water Flow Model for Steam Injection and Geothermal Applications. **SPE Journal,** 21(5), pp.527-534.

Anklam, E. and Wiggins, M. (2005) Horizontal well productivity and Wellbore Pressure Behaving Incorporating Wellbore Hydraulics. **SPR Production and Operations Symposium, April 17-19, Oklahoma City, Oklahoma.**

Ansari, A., Sylvester, N., Shoham, O. and Brill, J. (1990) A Comprehensive Mechanistic Model for Upward Two-Phase Flow in Wellbores. **SPE Annual Technical Conference and Exhibition, September 23-26, New Orleans, Louisiana** (20630-MS).

Avrami, M. (1940) Kinetics of Phase Change. II Transformation – Time Relations for Random Distribution of Nuclei. **Journal of Chemical Physics**, 8(2), pp.212-224.

Barnea, D. (1987) A Unified Model for Predicting Flow-Pattern Transitions for the Whole Range of Pipe Inclinations. **International Journal of Multiphase Flow**, 13, pp.1-12.

Beggs, H. and Brill, J. (1973) A Study of Two-Phase Flow in Inclined Pipes. **Journal of Petroleum Technology**, 25(5), pp.607-617.

Begatin, R. et al. (2008) Wax Modeling: There is a Need for Alternatives. **SPE Russian Oil and Gas Technical Conference and Exhibition, October 28-30, Moscow, Russia.**

Bendakhlia, H. and Aziz, K. (1989) Inflow Performance Relationships for Solution-Gas Drive Horizontal Wells. **64th Annual Technical Conference, October 8-11, San Antonio, Texas** (SPE 19823).

Bidmus, H. and Mehrotra, A. (2004) Heat-Transfer Analogy for Wax Deposition from Paraffinic Mixtures. **Industrial and Engineering Chemistry Research**, 43(3), pp.791-803.

Brill, P. and Arirachakaran, S. (1992) State of the Art in Multiphase Flow. **Journal of Petroleum Technology**, 44(5), pp. 538-541.

Bryne, M., Jimenez, M., Rojas, E. and Chavez, J. (2010) Modeling Well Inflow Potential in Three Dimensions Using Computational Fluid Dynamics. **SPE International Symposium and Exhibition on Formation Damage Control, February 10-12, Lafayette, Louisiana.**

Butler, R. (1994) Horizontal Wells for the Recovery of Oil, Gas and Bitumen. **Petroleum Society of CIM**, Petroleaum Monograph No.2.

Calange, S., Ruffier-Meray, V. and Behar, E. (1997) Onset Crystallization Temperature and Deposit Amount for Waxy Crudes: Experimental Determination and Thermodynamic Modelling. **SPE International Symposium on Oilfield Chemistry, February 18-21, Houston, Texas.**

Cazarez-Candia, P. and Vasquez-Cruz, M. (2004) Prediction of Pressure, Temperature, and Velocity Distribution of Two-Phase Flow in Oil Wells. **Journal of Petroleum Science and Engineering**, 46, pp.195-208.

CBC News (2005) Supply and Demand: World Oil Market Under Pressure. **CBC News** [Internet], 20 April 2005. Available from: <http://www.cbc.ca/news/background/oil/supply_demand.html> [Accessed 2 November 2010].

Cheng, A. (1990) Inflow Performance Relationship for Solution-Gas-Drive Slanted/Horizontal Wells. **65th Annual Technical Conference, September 23-26, New Orleans, Louisiana** (SPE 20720).

Correra, S., Fasano, A., Fusi, L. and Primicerio, M. (n.d.) **Modelling Wax Diffusion in Crude Oils: The Cold Finger Device** [Internet], Italy, University of Florence.

Available from: <http://web.math.unifi.it/users/primicer/coldagitato.pdf> [Accessed 28 November 2010].

Correra, S., Fasano, A., Fusi, L., Primicerio, M. and Rosso, F. (n.d.) **Wax Diffusivity Under Given Thermal Gradient: A Mathematical Modal** [Internet], Italy, University of Florence. Available from: <http://web.math.unifi.it/users/primicer/coldstatico.pdf> [Accessed 28 November 2010].

Dias-Couto, L. and Golan, M. (1982) General Inflow Performance Relationship for Solution-Gas Reservoir Wells. **Journal of Petroleum Technology**, 34(2), pp.285-288.

Dikken, B. (1989) Pressure Drop in Horizontal Wells and its Effect on their Production Performance. **64th Annual Technical Conference and Exhibition of the Society of Petroleum Engineers, October 8-11, San Antonio, Texas.**

Dukler, E., Baker, O., Cleveland, R., Hubbard, M. and Wicks, M. (1969) Gas-Liquid Flow in Pipe Lines, Part 1, Research Results. Monograph. **NX-28, University of Houston, May 1965, Houston, Texas.**

Duns, H. and Ros, N. (1963) Vertical Flow of Gas and Liquid Mixtures in Wells. **6th World Petroleum Congress, June 19-26, Frankfurt am Main, Germany** (10132).

Dawkrajai, P., Yoshioka, K., Romero, A., Zhu, D., Hill, A., and Lake, L. (2005) A Comprehensive Statistically-Based Method to Interpret Real-Time Flowing Measurements. **Annual Report University of Texas, October, Austin, Texas.**

Erickson, D., Niesen, V. and Brown, T. (1993) Thermodynamic Measurement and Prediction of Paraffin Precipitation in Crude Oil. **SPE Annual Technical Conference and Exhibition, October 3-6, Houston, Texas** (26604-MS).

Essley, P. (1974) **Petroleum Perspective**. SPE Eastern Regional Meeting of the Society of Petroleum Engineers of AIME, November 14-15, Washington, D.C. (SPE 4944).

Ezzati, M., Bailis, R., Kammen, D., Holloway, T., Price, L., Cifuentes, L., Barnes, B., Chaurey, A. and Dhanapala, K. (2004) Energy Management and Global Health. **Annual Reviews of Environment and Resources,** 29, pp. 383-419.

Falcone, G., Trodoriu, C., Reinicke, K. and Bello, O. (2007) Multiphase Flow Modeling Based on Experimental Testing: A Comprehensive Overview of Research Facilities Worldwide and the Need for Future Developments. **SPE**

Annual Technical Conference and Exhibition, November 11-14, Anaheim, California (SPE110116).

Fasano, A. and Primicerio, M. (n.d.) **Wax Deposition In Crude Oils: A New Approach** [Internet], Italy, University of Florence. Available from: <http://web.math.unifi.it/users/primicer/Wax%20crude%20oils.pdf> [28 November 2010].

Fetkovich, J. (1973) The Isochronal Testing of Oil Wells. **Fall Meeting of the Society of Petroleum Engineers of AIME, September 30 – October 30, Las Vegas, Nevada** (4521-MS).

Fourth quarter (2005) The Daily Canada's Population. **Statistics Canada** [Internet], 28 March 2006. Available from: <http://www.statcan.gc.ca/daily-quotidien/060328/dq060328e-eng.htm> [Accessed 2 November 2010].

Gilbert, W. (1954) Flowing and Gas-Lift Performance. **Drilling and Production Practices**, pp. 126-157.

Griffith, P. (1962) Two-Phase Flow in Pipes. **Special Summer Program Massachusetts Institute of Technology, Cambridge, Massachusetts.**

Griffith, P. and Wallis, G. (1961) Two-Phase Slug Flow. **Journal of Heat Transfer**, 83, pp.307-320.

Hagedorn, A. and Brown, K. (1965) Experimental Study of Pressure Gradients Occurring During Continuous Two-Phase Flow in Small-Diameter Vertical Conduits. **Journal of Petroleum Technology**, 17(4), pp.175-484.

Hagoort, J. (2004) Ramey's Wellbore Heat Transmission Revisited. **SPE Journal**, 9(4), pp.465-474.

Hammami, A. (1992) A Non-Isothermal Crystallization Kinetics of n-Paraffins with Chain Lengths between Thirty and Fifty. **Thermochimica Acta**, 211, pp.137.

Hasan, A. and Kabir, C. (1990) A New Model for Two-Phase Oil/Water Flow: Production Log Interpretation and Tubular Calculations. **SPE Production Engineering**, 5(2), pp.193-199.

Hoffman, J. (1985) Growth Rate of Extended-Chain Crystals. **Macromolecules**, 18, pp.772.

Holder, G. and Winkler, J. (1965) Was Crystallization from Distillate Fuels. Part I. Cloud and Pour Point Phenomena Exhibited by Solutions of Binary n-Paraffin Mixtures. **Journal of the Institute of Petroleum**, 51, pp.228-234.

Hsu, J. and Brubaker, J. (1995) Wax Deposition Scale-Up Modeling for Waxy Crude Production Lines. **27th Annual OTC, Houston, USA.**

Hsu, J., Santamaria, M. and Brubaker, J. (1994) Wax Deposition of Waxy Live Crudes Under Turbulent Flow Conditions. **69th Annual Technical Conference and Exhibition, New Orleans, USA.**

International Energy Agency (2010) **IEA Oil Market Report** [Internet], September 2010. France, International Energy Agency. Available from: <http://omrpublic.iea.org/world/wb_wodem.pdf> [Accessed 19 October 2010].

Joshi, S. (1988) Augmentation of Well Productivity with Slant and Horizontal Wells. **Journal of Petroleum Technology**, 40(6), pp.729-739.

Jahanbani, A. and Shadizadeh, S. (2009) Determination of Inflow Performance Relationship by Well Testing. **Canadian International Petroleum Conference, June 16-18, Calgary, Alberta.**

Johansen, A. (2007) **Work Term Report**. Work Term Report: Memorial University of Newfoundland, Canada.

Johansen, T. (2008) **Principles of Reservoir Engineering.** Canada, Memorial University of Newfoundland.

Johansen, T. and Khoryakov, V. (2007) Iterative Techniques in Modeling of Multi-Phase Flow in Advanced Wells and the Near Well Region. **Journal of Petroleum Science and Engineering**, 58, pp.49-67.

Joshi, S. (1991) **Horizontal Well Technology**, PennWell Books, Tulso.

Kabir, C. (1992) Inflow Performance of Slanted and Horizontal Wells in Solution-Gas-Drive Reservoirs. **SPE Western Regional Meeting, March 30-April 1, Bakersfield, California** (SPE 24050).

Kamkom, R. and Zhu, D. (2005) Evaluation of Two-Phase IPR Correlations for Horizontal Wells. **SPE Production Operations Symposium, April 16-19, Oklahoma City, Oklahoma** (93986-MS).

Lekia, S. and Evans, R. (1990) Generalized Inflow Performance Relationship for Stimulated Wells. **The Journal of Canadian Petroleum Technology**, 29(6), pp.71-75.

Lyons, W. and Plisga, G. (2005) **Standard Handbook of Petroleum and Natural Gas Engineering**. 2nd Edition. Elsevier.

Levant, E. (2010) **Ethical Oil: The Case of Canada's Oil Sands.** McClelland & Stewart.

Lindeloff, N. and Krejberg, K. (2002) A Compositional model Simulating Wax Deposition in Pipeline Systems. **Energy and Fuels**, 16, pp.887-891.

Liu, J. (2009) **Three-Phase Network Simulator for Horizontal Wells with Complex Advanced Well Completions**. Master's Thesis: Memorial University of Newfoundland.

McCarty, D. and Peaceman, D. (1957) Applications of Large Computers to Reservoir Engineering Problems. **Journal of Petroleum Technology**. 9(10), pp. 14-18.

Merino-Garcia, D., Margarone, M. and Correra, S. (2007) Kinetics of Waxy Gel Formation from Batch Experiments. **Energy and Fuels**, 21, pp.1287-1295.

Misra, S., Baruah, A. and Singh, K. (1995) Paraffin Problems in Crude Oil Production and Transportation: A Review. **SPE Production and Facilities**, 10(1), pp.50-54.

Moss, J. and White, P. (1959) How to Calculate Temperature Profiles in a Water-Injection Well. **Oil Gas Journal**, 57, pp.174-178.

Nazar, A., Dabir, B., Vaziri, H. and Islam, M. (2001) Experimental and Mathematical Modeling of Wax Deposition and Propagation in Pipes Transporting

Crude Oil. **SPE Production and Operations Symposium, March 24-27, Oklahoma City, Oklahoma.**

Novy, R. (1995) Pressure Drop in Horizontal Wells: When can they be Ignored? **SPE Reservoir Engineering**, 10(1), pp.29-35.

Nzekwu, B. (1989) Critical Review of the Application of Horizontal Wells. **The Third Technical meeting of the South Saskatchewan Section of the Petroleum Society of CIM, September 25-27, Regina, Manitoba.**

Orkizewski, J. (1967) Predicting Two-Phase Pressure Drops in Vertical Pipe. **Journal of Petroleum Technology**, 19(6), pp.829-838.

Oliver, M. and Calvert, P. (1975) Homogeneous Nucleation of n-Alkanes Measured by Differential Scanning Calorimetry. **Journal of Crystal Growth**, 30, pp. 343.

Ostrowski, L., Galimzyanov, A. and Uelker, E. (2010) Advances in Modeling of Passive Inflow Control Devices Help Optimizing Horizontal Well Completions. **Russian Oil and Gas Technical Conference and Exhibition, October 26-28, Moscow, Russia.**

Ozawa, T. (1971) Kinetics of Non-Isothermal Crystallization. **Polymer**, 12(3), pp.150-158.

Poettman, F. and Carpenter, P. (1952) The Multiphase Flow of Gas, Oil and Water Through Vertical Flow Strings. **Drilling and Production Practice**, 257.

Pruess, K. and Zhang, Y. (2005) A Hybrid Semi-Analytical and Numerical Method for Modeling Wellbore Heat Transmission. **30th Workshop on Geothermal Reservoir Engineering Stanford University, January 31-February 2, Stanford, California.**

Rao, B. (1998) **Multiphase Flow Models Range of Applicability** [Internet]. Texas, CETS. Available from: < http://www.ctes.com/Documentation/technotes/Tech%20Note%20Multiphase%20Flow%20Models.pdf> [Accessed 18 November 2010].

Ramey, H. (1961) Wellbore Heat Transmission. **Journal of Petroleum Technology**, 14(4), pp.427-435.

Retnanto, A. and Economides, M. (1998) Inflow Performance Relationships of Horizontal and Multi-branched Wells in a Solution-Gas-Drive Reservoir. **European Petroleum Conference, October 20-22, Hague, Netherlands** (SPE 50659).

Sharma, R., Zimmerman, D. and Mourits, F. (1995) Modeling of Undulating Wellbore Trajectories. **The Journal of Canadian Petroleum Technology**, 34(10), pp.16-24.

Skinner, J. (in progress) **Near Wellbore Modeling for Advanced Well Completions and Complex Well Trajectories.** M.Eng. Thesis: Memorial University of Newfoundland, Canada.

Standing, M. (1971) Concerning the Calculation of Inflow Performance Relationship for Solution-Gas Reservoir Wells. **Journal of Petroleum Technology**, 34(2), pp.285-288.

Tabatabaei, M. and Ghalambor, A. (2009) A New Method to Predict Performance of Horizontal and Multi-lateral Wells. **International Petroleum Technology Conference, December 7-9, Doha, Qatar.**

Taitel, Y. and Dukler, A. (1976) Theoretical Approach to Lockhart Martinelli Correlation for Stratified Flow. **International Journal for Multiphase Flow**, 2, pp.591-595.

Taitel, Y., Barnea, D. and Dukler, A. (1980) Modelling Flow Pattern Transitions for Steady Upward Gas-Liquid Flow in Vertical Tubes. **AIChE Journal**, 26, pp.345-354.

Thanyamanta, W., Johansen, T. and Hawboldt, K. (2009) Prediction of Asphaltene Precipitation using Non-Isothermal Compositional Network Model. **Journal of Petroleum Science and Engineering**, 24, pp.11-19.

Turnbull, D. and Spacpen, F. (1978) Crystal Nucleation and Crustal-Melt Interfacial Tension in Linear Hydrocarbons. **Journal of Polymer Science**, 63, pp. 137.

Vogel, V. (1968) Inflow Performance Relationships for Solution-Gas Drive Wells. **Journal of Petroleum Technology**, 20(1), pp.83-92.

Warren, J. and Mueller, T. (1957) Solution of a Typical Reservoir Problem of a Large Scale Computer. **Southern California Petroleum Section Annual Fall Meeting, October 17-18, Los Angeles, California** (959-G).

Weingarten, J. and Euchner, J. (1988) Methods for Predicting Wax Precipitation and Deposition. SPE **Production Engineering**, 3(1), pp.121-126.

Wooley, G. (1980) Computing Downhole Temperatures in Circulation, Injection, and Production Wells. **Journal of Petroleum Technology**, 32(9), pp.1509-1522.

Wu, Y. and Pruess, K. (1990) An Analytical Solution for Wellbore Heat Transmission in Layered Formations. **SPE Reservoir Engineering**, 5(4), pp.531-538.

Zougari, M. and Sopkow, T. (2007) Introduction to Crude Oil Wax Crystallization Kinetics: Process Modeling. **Industrial and Engineering Chemistry Results**, 46, pp.1360-1368.

APPENDIX A: MATLAB CODE

Horizontal Model Code

File name	Description
checkConvergence.m	Checks for convergence of parameters
checkconvmu.m	Checks for convergence of viscosity
checkConvT.m	Checks for convergence of temperature
conversion.m	Converts pressure to bara and flowrates to m^3/d
displaymu.m	Categorizes viscosity values
displayoutput.m	Display converged results in the appropriate units
displayoutputT.m	Display and categorize converged temperature values
f1generator.m	Generates function matrix for segment 1
f1T.m	Generates function matrix for segment 1 for temperatures
f2generator.m	Generates function matrix for segment 2 to N-1
f2T.m	Generates function matrix for segment 2 to N-1 for temperatures
f3generator.m	Generates function matrix for segment N
f3T.m	Generates function matrix for segment N for temperatures
flowrates.m	Categorizes flowrates
fractions.m	Categorizes fractions
gasmix.m**	Contains tables of fluid properties
generateBg.m*	Calculates Bg from tables
generateBgres.m*	Calculates Bg from tables
generateBindex.m	Input direction of flow of each bridge
generateBo.m*	Calculates Bo from tables
generateBores.m*	Calculates Bo from tables
generateflow.m	Generates flow rates for use in temperature calculations
generateFractions.m	Generates liquid fractions for temperature calculations
generatekappa.m	Generates overall heat transfer coefficient
Generatemu.m*	Calculates viscosity from tables
Generatemu2P.m*	Calculates two phase viscosity
generatemures.m*	Calculates viscosity from tables
generatepdrop.m	Generate pressure drop between nodes
generateResprop.m*	Calculating black oil properties at reservoir conditions using tables
generaterho.m*	Calculatesdensity from tables
generateRhores.m*	Calculates density from tables
generateRs.m*	Calculates Rs from tables
generateRsres.m*	Calculates Rs from tables
generateT.m	Setting wellbore temperatues to be same as reservoir temperatures for isothermal calculations

generateTres.m	Input reservoir temperatures
guess.m	Setting initial guessed values for unknown parameters
guessGenerator.m	Generate "guess" matrix
guessGeneratorT.m	Generate "guess" matrix for temperature calculations
input_alpha.m	Input liquid fractions in the reservoir
input_c.m	Input slot discharge coefficients
input_data.m	Input data for calculations
input_dataT.m	Input data for temperature calculations
input_K.m	Input absolute permeabilities
input_krg.m	Input gas relative permeabilities
input_kro.m	Input oil relative permeabilities
input_L.m	Input segment lengths
input_p.m	Input reservoir pressues
input_s.m	Input skin values
interpolate.m**	Interpolates to calculate properties
Iteration.m	Solving usning Newton-Raphson method
iterationT.m	Solving temperatures using Newton-Raphson method
j1Generator.m	Generates Jacobian matrix for segment 1
j1T.m	Generates Jacobian matrix for segment 1
j2Generator.m	Generates Jacobian matrix for segment 2 to N-1
j2T.m	Generates Jacobian matrix for segment 2 to N-1
j3Generator.m	Generates Jacobian matrix for segment N
j3T.m	Generates Jacobian matrix for segment N
main.m*	Oranizes files to solve the iterative process
Mysubplots.m**	Plots flowrates, pressures and flow rate graphs in the same figure
Networksolver.m	Organizes files to run the iterative method
plotflowrates.m	Makes flow rate plots
plotfractions.m	Makes liquid fraction plots
plotmu.m	Makes viscosity plots
plotpressure.m	Makes pressure plots
plotT.m	Makes temperature plots
plotting.m	Makes user defined plots
precalculations.m	Pre-calculates some coefficient values to run faster
pressures.m	Categorizes pressures
Tconversion.m	Scaling back from being relative to reference variables
Temperatures.m	Categorizes temperatures
TempSolver.m	Organizes files to solve for temperatures
updatemu.m	Recalculates viscosity values

*Only these files were significantly changed (about 90% average) to the work done by Thanyamanta et al. (2009)
**Only these files were newly added to the work done by Thanyamanta et al. (2009)

checkConvergence.m

```
% Checks for convergence

%Input:
%
%X1          :   Unknown parameters at nth iteration
%X2          :   Unknown parameters at n+1th iteration
%N           :   Number of unknowns
%threshold   :   Tolerance value for checking for convergence
%
%Return:
%flag        :   Convergence status (false = convergence)
%epsilon     :   Convergence value

function [func, func1] = checkConvergence(X1,X2,N,threshold)

flag = true;
epsilon = 0;
temp = 0;

for i=1:N
    if X1(i)~=0
        temp = abs(X1(i)-X2(i))/(X1(i)*N);
    else
        if X2==0
            temp=0;
        else
            X1(i) = 1e-20;
        end
    end

    epsilon = epsilon + temp;

end

if(epsilon < threshold)

   flag = false;

end

func = flag;
func1 = epsilon;
```

checkconvmu.m

```
% Check for convergence

%Input:
%
%mu2P        :   Viscosities used in last iteration
%mu2P_temp   :   Recalculated viscosities
%bridges     :   Number of bridges
%Threshold   :   Tolerance number
%
%Return:
%flag        :   Convergence status (false = convergence)
%epsilon     :   Convergence value

function [func1 func2] =
checkconvmu(mu2P,mu2P_temp,bridges,threshold)

flag = true;
epsilon = 0;
temp = 0;

for i=1:bridges

    temp = (1-mu2P_temp(i)/mu2P(i))^2;

    epsilon = epsilon + temp;

end

if(epsilon < threshold)

   flag = false;

end

func1 = flag;
func2 = epsilon;
```

checkConvT.m

```
% Checking for convergence

%Input:
%
%X1          :   Unknown temperatures at nth iteration
%X2          :   Unknown temperatures at n+1th iteration
%N           :   Number of unknown temperatures
%threshold   :   Tolerance value for checking for convergence
%
%Return:
%flag        :   Convergence status (false = convergence)
%epsilon     :   Convergence value

function [func, func1] = checkConvT(X1,X2,N,threshold)

flag = true;
epsilon = 0;
temp = 0;

for i=1:N
    if X1(i)~=0
        temp = abs(X1(i)-X2(i))/(X1(i)*N);
    else
        if X2==0
            temp=0;
        else
            X1(i) = 1e-20;
        end
    end

    epsilon = epsilon + temp;

end

if(epsilon < threshold)

   flag = false;

end

func = flag;
func1 = epsilon;
```

conversion.m

```
% Scaling back the variables so that pressures are in bara and
flowrates in
% m^3/d

%Input:
%
%X1          :   Unknown parameters (converged values)
%pref        :   Reference pressure
%qref        :   Reference flow rate
%num_var     :   Number of unknowns
%
%Return:
%An array containing values of unknown variables: pressures in bara
and
%flowrates in m^3/d

function func = conversion(X1,pref,qref,num_var)

tempCounter = 1;

for i=1:num_var-2

    if(tempCounter <= 2)            % Conversion of pressure
variables

        X1(i) = X1(i)*pref/1e5;
        tempCounter = tempCounter + 1;

    else                            % Conversion of flow rates
variables

        if (tempCounter<=6)
            X1(i) = X1(i)*qref*60*60*24;
            tempCounter = tempCounter + 1;
        elseif (tempCounter>6)
            tempCounter = tempCounter + 1;
        end

    end

    if(tempCounter == 10)

        tempCounter = 1;

    end

end

func = X1;
```

displaymu.m

```
% Categorize viscosity values

mu_tubing = zeros(1,N);              % Viscosity in tubing
mu_annulus = zeros(1,N);             % Viscosity in annulus
mu_reservoir = zeros(1,N-1);         % Viscosity in reservoir

for i=0:N-2      % Segment 2 to N-1
    mu_tubing(i+1) = mu2P(4*i+1);
    mu_annulus(i+1) = mu2P(4*i+3);
    mu_reservoir(i+1) = mu2P(4*i+4);
end              % Segment N
    mu_tubing(N) = mu2P(bridges-1);
    mu_annulus(N) = mu2P(bridges);

% Plot results
%plotmu;
```

displayoutput.m

```
% Conversions of the converged variables back to their appropriate
units and
% result display

% Display number of iterations
disp('Number of iterations before convergence');
disp(num2str(sentinelCount));

% Scaling back the variables so that pressures are in bara and
flowrates in
% m^3/d
X = conversion(X1,pref,qref,num_var);

% Display final results
disp('-----------------------');
disp('X');
disp(num2str(transpose(X)));

% Categorize variables into pressures, flow rates, and liquid holdups
p_tubing = [];              % Pressures in tubing
p_annulus = [];             % Pressures in annulus
tubingFlowrates = [];       % Tubing flow rates
slotFlowRates = [];         % Annular-to-tubing flow rates
annularFlowrates = [];      % Annular flow rates
inflowRates = [];           % Inflow rates
tubingFractions = [];       % Liquid holdups in tubing
slotFractions = [];         % Liquid holdups in Annular-to-tubing
flows
annularFractions = [];      % Liquid holdups in annulus

[p_tubing, p_annulus] = pressures(X,N,num_var);
% Categorize pressures
[tubingFlowrates, slotFlowRates, annularFlowrates, inflowRates] =
flowrates(X,N,num_var,bindex);    % Categorize flow rates
[tubingFractions, slotFractions, annularFractions] =
fractions(X,N,num_var);                    % Categorize fractions

% Calculate cumulative inflows at each segment
integralFlows = zeros(1,N);

for i=1:N-1

    for j=1:i
       integralFlows(i) = integralFlows(i) + inflowRates(j);
    end
    integralFlows(N) = integralFlows(N-1);

end

% Plot results
%plotting
```

displayoutputT.m

```
% Categorize converged temperature variables and
% result display

% Display number of iterations
disp('Number of iterations before convergence');
disp(num2str(sentinelCount));    %.............................??

% Scaling back the variables if the values are relative to reference
value
XT = Tconversion(XT,num_varT,Tref);

% Display final results
disp('-----------------------');
disp('X');
disp(num2str(transpose(XT)));

% Categorize variables
T_tubing = [];                  % Temperatures in tubing
T_annulus = [];                 % Temperatures in annulus

[T_tubing, T_annulus] = Temperatures(XT,N,num_varT,Tres);

% Plot results
%plotT
```

f1generator.m

```
% Generate function matrix for Segment 1

%Input:
%
%X1          :   Unknown parameters at each iteration
%I           :   Pre-calculated coefficient for inflow equations
%pres        :   Reservoir pressures
%beta        :   Pre-calculated coefficient for tubing flow
calculations
%alpha       :   Pre-calculated coefficient for annular
flowcalculations
%B           :   Pre-calculated coefficient for slot/valve flow
calculations
%Bo,Bg,Rs    :   Black-oil properties
%mu2P        :   Two-phase viscosities
%rho2P       :   Two-phase densities
%alpha_res   :   Liquid holdups in reservoir
%f1          :   Generated zero function matrix
%pref        :   Reference pressure
%qref        :   Reference flow rate
%
%Return:
%Function matrix for Segment 1

function func =
f1Generator(X1,I,pres,beta,alpha,B,Bo,Bg,Rs,mu2P,rho2P,alpha_res,f1,p
ref,qref)

% Liquid-phase material balance
    f1(1) = X1(4)*X1(8)/Bo(2) - X1(3)*X1(7)/Bo(1); % At node 1
    f1(2) = X1(6)*alpha_res(1)/Bo(3)...
            - X1(4)*X1(8)/Bo(2)...
            - X1(5)*X1(9)/Bo(2);                    % At node 2
% Inflow equation
    f1(3) = X1(6)...
            - I(1)*(pres(1)/pref - X1(2))*pref/qref;
% Momentum balance for tubing bridge
    f1(4) = X1(1) - X1(10)...
            - beta(1)*(X1(3)^1.75)*rho2P(1)^0.75*mu2P(1)^0.25;
% Flow equation for annular-to-tubing bridge
    f1(5) = X1(2) - X1(1)...
            - B(1)*(X1(4)^2)*rho2P(2);
% Momentum balance for annular bridge
    f1(6) = X1(2) - X1(11)...
            - alpha(1)*(X1(5)^1.75)*rho2P(3)^0.75*mu2P(3)^0.25;
% Gas-phase material balance
    f1(7) = ((1-X1(8))*X1(4)/Bg(2) + X1(8)*Rs(2)*X1(4)/Bo(2))...
            - ((1-X1(7))*X1(3)/Bg(1) + X1(7)*Rs(1)*X1(3)/Bo(1));    %
At node 1
    f1(8) = ((1-alpha_res(1))*X1(6)/Bg(3) +
alpha_res(1)*Rs(3)*X1(6)/Bo(3))...
            - ((1-X1(8))*X1(4)/Bg(2) + X1(8)*Rs(2)*X1(4)/Bo(2))...
            - ((1-X1(9))*X1(5)/Bg(2) + X1(9)*Rs(2)*X1(5)/Bo(2));    %
At node 2
```

```
% Split equation
    f1(9) = X1(9) - X1(8);

func = f1;
```

f1T.m

```
% Generate function matrix for Segment 1

%Input:
%
%Doil, Dgas :   Precalculated coefficients
%Tres       :   Reservoir temperatures
%Tref       :   Reference temperature
%XT         :   Unknown temperatures at each iteration
%q          :   Flow rates
%Lfrac      :   Liquid holdups
%Bo,Bg,Rs   :   Black-oil properties
%Kappa_t    :   Overall heat transfer coefficients for fluid in
tubing
%Kappa_a    :   Overall heat transfer coefficients for fluid in
annulus
%f1         :   Generated zero function matrix
%deltaP_t   :   Pressure drop between nodes for fluid in tubing
%deltaP_a   :   Pressure drop between nodes for fluid in annulus
%KJT        :   Joule-Thompson coefficient
%
%Return:
%Function matrix for Segment 1

function func =
f1T(Doil,Dgas,Tres,Tref,XT,q,Lfrac,Bo,Bg,Rs,Kappa_t,Kappa_a,f1,deltaP
_a,deltaP_t,KJT)

Ttoe = Tres(1);

% Assign temperature in tubing of Segment one equal reservoir
temperature
    f1(1) = XT(1) - Ttoe/Tref;
% Energy balance at tubing node
    f1(2) = -(Doil*q(2)*Lfrac(2)/Bo(2)...
                + Dgas*(q(2)*(1-Lfrac(2))/Bg(2)...
                + q(2)*Lfrac(2)*Rs(2)/Bo(2)))*((XT(1)-Ttoe/Tref))...
            - (Doil*q(1)*Lfrac(1)/Bo(1)...
                + Dgas*(q(1)*(1-Lfrac(1))/Bg(1)...
                + q(1)*Lfrac(1)*Rs(1)/Bo(1)))*((XT(3)-XT(1))-
KJT*deltaP_t(1))...
            - Kappa_t(1)*(XT(1)-Ttoe/Tref);
% Energy balance at annular node
    f1(3) = - (Doil*q(3)*Lfrac(3)/Bo(2)...
                + Dgas*(q(3)*(1-Lfrac(3))/Bg(2)...
                + q(3)*Lfrac(3)*Rs(2)/Bo(2)))*((XT(2)-Ttoe/Tref)-
KJT*deltaP_a(1));

func = f1;
```

f2generator.m

```
% Generate function matrix for Segment 2 to N-1

%Input:
%
%X1          :   Unknown parameters at each iteration
%beta        :   Pre-calculated coefficient for tubing flow
calculations
%alpha       :   Pre-calculated coefficient for annular
flowcalculations
%B           :   Pre-calculated coefficient for slot/valve flow
calculations
%I           :   Pre-calculated coefficient for inflow equations
%pres        :   Reservoir pressures
%Bo,Bg,Rs    :   Black-oil properties
%mu2P        :   Two-phase viscosities
%rho2P       :   Two-phase densities
%alpha_res   :   Liquid holdups in reservoir
%f2          :   Generated zero function matrix
%pref        :   Reference pressure
%qref        :   Reference flow rate
%N           :   Number of segments
%b           :   Bridge indexes
%
%Return:
%Function matrix for Segment 2 to N-1

function func =
f2Generator(X1,beta,alpha,B,I,pres,Bo,Bg,Rs,mu2P,rho2P,alpha_res,f2,p
ref,qref,N,b)

for i=0:N-3

    var = i*9;

% Liquid-phase material balance
    f2(1+var) = X1(13+var)*X1(17+var)/Bo(3*i+5)*b(4*i+6)...
                + X1(3+var)*X1(7+var)/Bo(3*i+1)*b(4*i+1)...
                - X1(12+var)*X1(16+var)/Bo(3*i+4)*b(4*i+5);          %
At tubing node
    f2(2+var) = X1(5+var)*X1(9+var)/Bo(3*i+2)*b(4*i+3)...
                + X1(15+var)*alpha_res(i+2)/Bo(3*i+6)*b(4*i+8)...
                - X1(14+var)*X1(18+var)/Bo(3*i+5)*b(4*i+7)...
                - X1(13+var)*X1(17+var)/Bo(3*i+5)*b(4*i+6);          %
At annular node
% Inflow equation
    if b(4*i+8) ~= 0        % Inlet flow exists
        f2(3+var) = X1(15+var)...
                    - (I(i+2)*(pres(i+2)/pref -
X1(11+var)))*pref/qref;
    else                    % No inlet flow
        f2(3+var) = 0;
    end
% Momentum balance for tubing bridge
    f2(4+var) = X1(10+var) - X1(19+var)...
```

```
                -
beta(i+2)*(X1(12+var)^1.75)*rho2P(4*i+5)^0.75*mu2P(4*i+5)^0.25;
% Flow equation for annular-to-tubing bridge
    if b(4*i+6) ~= 0        % Annular-to-tubing flow exists
        f2(5+var) = X1(11+var) - X1(10+var)...
                    - B(i+2)*(X1(13+var)^2)*rho2P(4*i+6);
    else                    % No annular-to-tubing flow
        f2(5+var) = 0;
        f2(9+var) = 0;      % No split equation
    end
% Momentum balance for annular bridge
    if b(4*i+7) ~= 0        % Annular flow exists
        f2(6+var) = X1(11+var) - X1(20+var)...
                -
alpha(i+2)*(X1(14+var)^1.75)*rho2P(4*i+7)^0.75*mu2P(4*i+7)^0.25*b(4*i
+7);
        if b(4*i+7) == 1
            if b(4*i+6) ~= 0    % If there is both annular and
annular-to-tubing flows
                f2(9+var) = X1(17+var) - X1(18+var);    % Split
equation
            end
        elseif b(4*i+7) == -1   % If flow in annulus is toward toe of
well
            f2(9+var) = 0;                              % No split
equation
        end
    else                    % No annular flow
        f2(6+var) = 0;
        f2(9+var) = 0;
    end
% Gas-phase material balance
    f2(7+var) = ((1-X1(17+var))*X1(13+var)/Bg(3*i+5) +
X1(17+var)*Rs(3*i+5)*X1(13+var)/Bo(3*i+5))*b(4*i+6)...
                + ((1-X1(7+var))*X1(3+var)/Bg(3*i+1) +
X1(7+var)*Rs(3*i+1)*X1(3+var)/Bo(3*i+1))*b(4*i+1)...
                - ((1-X1(16+var))*X1(12+var)/Bg(3*i+4) +
X1(16+var)*Rs(3*i+4)*X1(12+var)/Bo(3*i+4))*b(4*i+5);        % At
tubing node
    f2(8+var) = ((1-X1(9+var))*X1(5+var)/Bg(3*i+2) +
X1(9+var)*Rs(3*i+2)*X1(5+var)/Bo(3*i+2))*b(4*i+3)...
                + ((1-alpha_res(i+2))*X1(15+var)/Bg(3*i+6) +
alpha_res(i+2)*Rs(3*i+6)*X1(15+var)/Bo(3*i+6))*b(4*i+8)...
                - ((1-X1(18+var))*X1(14+var)/Bg(3*i+5) +
X1(18+var)*Rs(3*i+5)*X1(14+var)/Bo(3*i+5))*b(4*i+7)...
                - ((1-X1(17+var))*X1(13+var)/Bg(3*i+5) +
X1(17+var)*Rs(3*i+5)*X1(13+var)/Bo(3*i+5))*b(4*i+6);        % At
annular node

end

func = f2;
```

f2T.m

```
% Generate function matrix for Segment 2 to N-1

%Input:
%
%Doil, Dgas :   Precalculated coefficients
%Tres       :   Reservoir temperatures
%Tref       :   Reference temperature
%XT         :   Unknown temperatures at each iteration
%q          :   Flow rates
%Lfrac      :   Liquid holdups
%Bo,Bg,Rs   :   Black-oil properties
%N          :   Number of segments
%Kappa_t    :   Overall heat transfer coefficients for fluid in
tubing
%Kappa_a    :   Overall heat transfer coefficients for fluid in
annulus
%f2         :   Generated zero function matrix
%b          :   Bridge indexes
%deltaP_t   :   Pressure drop between nodes for fluid in tubing
%deltaP_a   :   Pressure drop between nodes for fluid in annulus
%KJT        :   Joule-Thompson coefficient
%
%Return:
%Function matrix for Segment 2 to N-1

function func =
f2T(Doil,Dgas,Tres,Tref,XT,q,Lfrac,Bo,Bg,Rs,N,Kappa_t,Kappa_a,f2,b,de
ltaP_a,deltaP_t,KJT)

for i=0:N-3

    var = i*4;

% Energy balance at tubing node
    f2(1+2*i) = -(Doil*q(var+6)*Lfrac(var+6)/Bo(3*i+5)...
                    + Dgas*(q(var+6)*(1-Lfrac(var+6))/Bg(3*i+5)...
                    +
q(var+6)*Lfrac(var+6)*Rs(3*i+5)/Bo(3*i+5)))*((XT(2*i+3)-
XT(2*i+2)))*b(var+6)...
                - (Doil*q(var+5)*Lfrac(var+5)/Bo(3*i+4)...
                    + Dgas*(q(var+5)*(1-Lfrac(var+5))/Bg(3*i+4)...
                    +
q(var+5)*Lfrac(var+5)*Rs(3*i+4)/Bo(3*i+4)))*((XT(2*i+5)-XT(2*i+3))-
KJT*deltaP_t(i+2))*b(var+5)...
                - Kappa_t(i+2)*(XT(2*i+3)-XT(2*i+2));
% Energy balance at annular node
    f2(2+2*i) = -(Doil*q(var+8)*Lfrac(var+8)/Bo(3*i+6)...
                    + Dgas*(q(var+8)*(1-Lfrac(var+8))/Bg(3*i+6)...
                    +
q(var+8)*Lfrac(var+8)*Rs(3*i+6)/Bo(3*i+6)))*((XT(2*i+2)-
Tres(i+2)/Tref))*b(var+8)...
                - Kappa_a(i+2)*(XT(2*i+2)-Tres(i+2)/Tref)...
                - (Doil*q(var+7)*Lfrac(var+7)/Bo(3*i+5)...
                    + Dgas*(q(var+7)*(1-Lfrac(var+7))/Bg(3*i+5)...
```

```
                    +
q(var+7)*Lfrac(var+7)*Rs(3*i+5)/Bo(3*i+5)))*((XT(2*i+4)-XT(2*i+2))-
KJT*deltaP_a(i+2))*b(var+7);
% If there is discontinuity in annular flow the temperature at the
first
% annular node is assumed to have the temperature of the inflow from
the
% reservoir
   if b(var+7)==0
        f2(2+2*i) = XT(i*2+4)-Tres(i+2)/Tref;
   end

end

func = f2;

% For case 3 (well with multiple inflow control valves), add
% c = b;                        % c is bridge indices
% Create all-positive bridge indeces so that the temperature change
due to
% pressure drop is dependent on the flow direction but only on the
pressure
% drop (either positive or negative change along the flow direction)
% for i = 1:(4*N-2)
%     if b(i)<0
%         b(i)=-b(i);
%     end
% end
% Flow directions can still be determined using "c".  At the annular
node
% where the reversed flow first starts, the temperature is fixed so
that it
% equals to the temperature of the reservoir inflow.
% if c(var+3)==-1
%   if c(var+7)==1
%       f2(2+2*i) = XT(i*2+2)-Tres(i+2)/Tref;
%       f2(2+2*(i-1)) = XT((i-1)*2+4)-Tres(i+2)/Tref;
%   end
% end
```

f3generator.m

```
% Generate function matrix for Segment N

%Input:
%
%X1          :   Unknown parameters at each iteration
%beta        :   Pre-calculated coefficient for tubing flow
calculations
%B           :   Pre-calculated coefficient for slot/valve flow
calculations
%Bo,Bg,Rs    :   Black-oil properties
%mu2P        :   Two-phase viscosities
%rho2P       :   Two-phase densities
%f3          :   Generated zero function matrix
%pref        :   Reference pressure
%pbh         :   Bottomhole pressure
%N           :   Number of segments
%num_var     :   Number of unknowns
%Nodes       :   Number of nodes
%bridges     :   Number of bridges
%b           :   Bridge indexes
%
%Return:
%Function matrix for Segment N

function func =
f3Generator(X1,beta,B,Bo,Bg,Rs,mu2P,rho2P,f3,pref,pbh,N,num_var,Nodes
,bridges,b)

% Liquid-phase material balance
    f3(1) = X1(num_var-12)*X1(num_var-8)/Bo(Nodes-4)*b(bridges-5)...
            + X1(num_var-2)*X1(num_var)/Bo(Nodes)*b(bridges)...
            - X1(num_var-3)*X1(num_var-1)/Bo(Nodes-1)*b(bridges-1);
% At tubing node
    f3(2) = X1(num_var-10)*X1(num_var-6)/Bo(Nodes-3)*b(bridges-3)...
            - X1(num_var-2)*X1(num_var)/Bo(Nodes)*b(bridges);
% At annular node
% Momentum balance for tubing bridge
    f3(3) = X1(num_var-5) - pbh/pref...
            - beta(N)*(X1(num_var-3)^1.75)*rho2P(bridges-
1)^0.75*mu2P(bridges-1)^0.25;
% Flow equation for annular-to-tubing bridge
    if b(bridges) ~= 0          % Annular-to-tubing flow exists
        f3(4) = X1(num_var-4) - X1(num_var-5) - B(N)*(X1(num_var-
2)^2)*rho2P(bridges);
    else                        % No annular-to-tubing flow
        f3(4) = 0;
    end
% Gas-phase material balance
    f3(5) = ((1-X1(num_var-8))*X1(num_var-12)/Bg(Nodes-4) +
X1(num_var-8)*Rs(Nodes-4)*X1(num_var-12)/Bo(Nodes-4))*b(bridges-5)...
            + ((1-X1(num_var))*X1(num_var-2)/Bg(Nodes) +
X1(num_var)*Rs(Nodes)*X1(num_var-2)/Bo(Nodes))*b(bridges)...
```

```
            - ((1-X1(num_var-1))*X1(num_var-3)/Bg(Nodes-1) +
X1(num_var-1)*Rs(Nodes-1)*X1(num_var-3)/Bo(Nodes-1))*b(bridges-1);
% At tubing node
    f3(6) = ((1-X1(num_var-6))*X1(num_var-10)/Bg(Nodes-3) +
X1(num_var-6)*Rs(Nodes-3)*X1(num_var-10)/Bo(Nodes-3))*b(bridges-3)...
            -  ((1-X1(num_var))*X1(num_var-2)/Bg(Nodes) +
X1(num_var)*Rs(Nodes)*X1(num_var-2)/Bo(Nodes))*b(bridges);
% At annular node

func = f3;
```

f3T.m

```
% Generate function matrix for Segment N

%Input:
%
%Doil, Dgas :   Precalculated coefficients
%XT         :   Unknown temperatures at each iteration
%q          :   Flow rates
%Lfrac      :   Liquid holdups
%Bo,Bg,Rs   :   Black-oil properties
%N          :   Number of segments
%f3         :   Generated zero function matrix
%num_varT   :   Number of unknown temperatures
%Nodes      :   Number of node
%bridges    :   Number of bridges
%Kappa_t    :   Overall heat transfer coefficients for fluid in
tubing
%Kappa_a    :   Overall heat transfer coefficients for fluid in
annulus
%b          :   Bridge indexes
%deltaP_a   :   Pressure drop between nodes for fluid in annulus
%deltaP_t   :   Pressure drop between nodes for fluid in tubing
%KJT        :   Joule-Thompson coefficient
%
%Return:
%Function matrix for Segment N

function func =
f3T(Doil,Dgas,XT,q,Lfrac,Bo,Bg,Rs,N,f3,num_varT,Nodes,bridges,Kappa_t
,Kappa_a,b,deltaP_a,deltaP_t,KJT)

% Energy balance at tubing node
    f3(1) = -(Doil*q(bridges)*Lfrac(bridges)/Bo(Nodes)...
                + Dgas*(q(bridges)*(1-Lfrac(bridges))/Bg(Nodes)...
                +
q(bridges)*Lfrac(bridges)*Rs(Nodes)/Bo(Nodes)))*((XT(num_varT-1)-
XT(num_varT-2)))*b(bridges)...
            - (Doil*q(bridges-1)*Lfrac(bridges-1)/Bo(Nodes-1)...
                + Dgas*(q(bridges-1)*(1-Lfrac(bridges-1))/Bg(Nodes-
1)...
                + q(bridges-1)*Lfrac(bridges-1)*Rs(Nodes-1)/Bo(Nodes-
1)))*((XT(num_varT)-XT(num_varT-1))-KJT*deltaP_t(N))*b(bridges-1)...
            - Kappa_t(N)*(XT(num_varT-1)-XT(num_varT-2));

func = f3;
```

flowrates.m

```
% Categorize flow rates

%Input:
%
%X1          :   Unknown parameters (converged values)
%N           :   Number of segments
%num_var     :   Number of unknowns
%
%Return:
%tubingFlowrates     :   Tubing flow rates
%slotFlowRates       :   Annular-to-tubing flow rates
%annularFlowrates    :   Annular flow rates
%inflowRates         :   Inflow rates at each inlet bridge

function [func1, func2, func3, func4] =
flowrates(X1,N,num_var,bindex)

% Segment 1
tubingFlowrates(1) = X1(3);
slotFlowRates(1) = X1(4);
annularFlowrates(1) = X1(5);
inflowRates(1) = X1(6);

% Segment 2 to N-1
for i=1:N-2

    tubingFlowrates(i+1) = X1(3+9*i);
    slotFlowRates(i+1) = X1(4+9*i);
    annularFlowrates(i+1) = X1(5+9*i);
    inflowRates(i+1) = X1(6+9*i);

end

% Segment N
tubingFlowrates(N) = X1(num_var-3);
slotFlowRates(N) = X1(num_var-2);

for i=1:N-1
    annularFlowrates(i) = annularFlowrates(i)*bindex(4*(i-1)+3);
    slotFlowRates(i) = slotFlowRates(i)*bindex(4*(i-1)+2);
end

func1 = tubingFlowrates;
func2 = slotFlowRates;
func3 = annularFlowrates;
func4 = inflowRates;
```

fractions.m

```
% Categorize fractions

%Input:
%
%X1          :   Unknown parameters (converged values)
%N           :   Number of segments
%num_var     :   Number of unknowns
%
%Return:
%tubingFractions     :   Liquid holdups in tubing
%slotFractions       :   Liquid holdups in annular-to-tubing flows
%annularFractions    :   Liquid holdups in annulus

function [func1, func2, func3] = fractions(X1,N,num_var)

% Segment 1
tubingFractions(1) = X1(7);
slotFractions(1) = X1(8);
annularFractions(1) = X1(9);

% Segment 2 to N-1
for i=1:N-2

    tubingFractions(i+1) = X1(7+9*i);
    slotFractions(i+1) = X1(8+9*i);
    annularFractions(i+1) = X1(9+9*i);

end

% Segment N
tubingFractions(N) = X1(num_var-1);
slotFractions(N) = X1(num_var);

func1 = tubingFractions;
func2 = slotFractions;
func3 = annularFractions;
```

gasmix.m

```
% all properties are at 100C
% this fluid is a liquid at all temperature and pressure range shown
below
% fluid composition is a random one

%bara
Pressure=[
1.00
43.86
86.71
129.57
172.43
215.29
258.14
301.00
343.86
386.71
429.57
472.43
515.29
558.14
601.00
643.86
686.71
729.57
772.43
815.29
858.14
901.00];


%g/cm3
Density_Vap = [
0.0010
0.0324
0.0638
0.0961
0.1288
0.1613
0.1935
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
```

```
0.0000
0.0000
];

Density_Liq = [
0.8500
0.8083
0.7885
0.7710
0.7544
0.7383
0.7225
0.7252
0.7313
0.7369
0.7420
0.7467
0.7511
0.7552
0.7590
0.7626
0.7660
0.7691
0.7721
0.7750
0.7776
0.7802];


Viscosity_Vap = [
0.0130
0.0146
0.0159
0.0175
0.0195
0.0219
0.0248
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000];

Viscosity_Liq=[
4.3112
1.3087
```

```
1.0176
0.8767
0.7761
0.6623
0.5568
0.5651
0.5925
0.6196
0.6465
0.6731
0.6994
0.7256
0.7515
0.7771
0.8026
0.8278
0.8528
0.8777
0.9024
0.9270];

xBo = [
1.043
1.171
1.221
1.271
1.324
1.381
1.444
1.456
1.445
1.433
1.422
1.412
1.403
1.395
1.387
1.381
1.374
1.368
1.362
1.357
1.352
1.347
1.343];

xBg = [
1.325
0.028
0.014
0.009
0.007
0.006
0.005
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0];

xRs = [
0
33.1
52.3
72.3
93.9
117.4
143.4
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2
148.2];
```

generateBg.m

```
% Calculate pressure-dependent gas formation volume factors

%Input:
%
%X1          :   Unknown parameters at each iteration
%pres        :   Reservoir pressures at reservoir nodes
%Bg_res      :   Gas formaiton volume factor in reservoir
%pref        :   Reference pressure
%pb          :   Bubblepoint pressure
%N           :   Number of segments
%num_var     :   Number of unknowns
%Nodes       :   Number of nodes
%
%Return:
%Array containing gas formation volume factor at every node in the
network

function func = generateBg(X1,pres,Bg_res,pref,pb,N,num_var,Nodes)

Bg = zeros(1,Nodes);
gasmix;

for i=0:N-2

    var = i*9;

    X1(1+var) = X1(1+var)*pref/10^5;                    % Change unit
of pressure from Pa to bara
    X1(2+var) = X1(2+var)*pref/10^5;                    % Change unit
of pressure from Pa to bara

    % Calculate gas formaiton volume factor at tubing node of Segment
i+1
    p = X1(1+var);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
        % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    yBg = (p-p1)/dp * (xBg(i1)-xBg(i2)) + xBg(i1);
    Bg(3*i+1) = yBg;


    % Calculate gas formaiton volume factor at annular node of
Segment i+1
    p = X1(2+var);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
```

```
        % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    yBg = (p-p1)/dp * (xBg(i1)-xBg(i2)) + xBg(i1);
    Bg(3*i+2) = yBg;


    % Gas formaiton volume factor at inlet node = oil formaiton
volume factor in the reservoir
     Bg(3*i+3) = Bg_res(i+1);

end

X1(num_var-5) = X1(num_var-5)*pref/10^5;                    % Change unit
of pressure from Pa to bara
X1(num_var-4) = X1(num_var-4)*pref/10^5;                    % Change unit
of pressure from Pa to bara

% Calculate gas formaiton volume factor at tubing node of Segment N
    p = X1(num_var-5);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    yBg = (p-p1)/dp * (xBg(i1)-xBg(i2)) + xBg(i1);
    Bg(Nodes-1) = yBg;

% Calculate gas formaiton volume factor at annular node of Segment N
    p = X1(num_var-4);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    yBg = (p-p1)/dp * (xBg(i1)-xBg(i2)) + xBg(i1);
    Bg(Nodes) = yBg;

func = Bg;
```

generateBgres.m

```
% Calculate pressure-dependent gas formation volume factors

%Input:
%
%N          :   Number of segments
%pres       :   Reservoir pressures at reservoir nodes
%pb         :   Bubblepoint pressure
%
%Return:
%Array containing gas formation volume factors at inlet nodes

function func = generateBgres(N,pres,pb)

pres_temp = pres/10^5;              % Change unit of pressure from Pa
to bara
Bg_res = zeros(1,N-1);
gasmix;

for i=1:N-1

    p = pres_temp(i);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
        % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    yBg = (p-p1)/dp * (xBg(i1)-xBg(i2)) + xBg(i1);
    Bg_res(i) = yBg;


%     if pres_temp(i)<pb              % For pressures below the
bubblepoint pressures
%         Bg_res(i) = -4e-09*pres_temp(i)^3 + 2e-06*pres_temp(i)^2 -
0.0004*pres_temp(i) + 0.0378;       % Expression obtained from
curve-fitting of pre-generated values using an EOS
%     else                            % For pressures above the
bubblepoint pressures
%         Bg_res(i) = -4e-09*pb^3 + 2e-06*pb^2 - 0.0004*pb + 0.0378;
%     end

end

func = Bg_res;
```

generateBindex.m

```
% Generate indexes for directions of flow through each bridge

% bindex value of +1 means flowing toward well's heel
% bindex value of -1 means flowing toward well's toe
% bindex value of 0 means the bridge is removed

bindex = ones(1,bridges);        % Assign "+1" for all bridges

% for blah=1:50
%     bindex(4*(blah-1)+3) = -1;       % To create reversed flow in
annulus of Segment blah
%     bindex(4*(blah-1)+2) = 0;        % To remove the annular-to-
tubing bridges
% end
%
%
% bindex(4*(N-1)+1) = 1;       % To create reversed flow in annulus of
Segment blah
% bindex(4*(N-1)+2) = 1;        % To remove the annular-to-tubing
bridges


% Assign "-1" or "0" as required

% For Example case 1 where there is a 500-meter packed off at the end
of the well, type:

% bindex(4*(149-1)+3) = 0;       % To remove the annular bridge
representing pack-off material
% for i = 150:199                % To remove the annular-to-tubing
bridges representing disconnections between annulus and tubing
%     bindex(4*(i-1)+2) = 0;
% end

% For case 3 (well with multiple inflow control valves),
% bindex(4*(69-1)+3) = 0;        % To remove the annular bridge
representing pack-off material
% for i = 70:99                  % To remove the annular-to-tubing
bridges representing disconnections between annulus and tubing
%     bindex(4*(i-1)+2) = 0;
% end
% bindex(4*(95-1)+2) = 1;        % To create annular-to-tubing flow
through ICV 2 at Segment 95
% bindex(4*(95-1)+3) = -1;       % To create reversed flow in annulus
of Segment 95
% bindex(4*(96-1)+3) = -1;       % To create reversed flow in annulus
of Segment 96
% bindex(4*(97-1)+3) = -1;       % To create reversed flow in annulus
of Segment 97
% bindex(4*(98-1)+3) = -1;       % To create reversed flow in annulus
of Segment 98
```

```
% For Case 2-C (well with two temperature zones), different-pressure
case,
% bindex(4*(100-1)+3) = 0;       % To create discontinuity between
Zone-1 and Zone-2 wells
% for i = 101:199
%     bindex(4*(i-1)+2) = 0;     % To remove the annular-to-tubing
bridges representing disconnections between the two wells
% end
%
%
% bindex(4*(69-1)+3) = 0;        % To remove the annular bridge
representing pack-off material
% for i = 70:99                  % To remove the annular-to-tubing
bridges representing disconnections between annulus and tubing
%     bindex(4*(i-1)+2) = 0;
% end
% bindex(4*(95-1)+2) = 1;        % To create annular-to-tubing flow
through ICV 2 at Segment 95
% bindex(4*(95-1)+3) = -1;       % To create reversed flow in annulus
of Segment 95
% bindex(4*(96-1)+3) = -1;       % To create reversed flow in annulus
of Segment 96
% bindex(4*(97-1)+3) = -1;       % To create reversed flow in annulus
of Segment 97
% bindex(4*(98-1)+3) = -1;       % To create reversed flow in annulus
of Segment 98
```

generateBo.m

```
% Calculate pressure-dependent oil formation volume factors

%Input:
%
%X1          :   Unknown parameters at each iteration
%pres        :   Reservoir pressures at reservoir nodes
%Bo_res      :   Oil formaiton volume factor in reservoir
%pref        :   Reference pressure
%pb          :   Bubblepoint pressure
%N           :   Number of segments
%num_var     :   Number of unknowns
%Nodes       :   Number of nodes
%
%Return:
%Array containing oil formaiton volume factor at every node in the
network

function func = generateBo(X1,pres,Bo_res,pref,pb,N,num_var,Nodes)

Bo = zeros(1,Nodes);
gasmix;


for i=0:N-2

    var = i*9;

    X1(1+var) = X1(1+var)*pref/10^5;            % Change unit of
pressure from Pa to bara
    X1(2+var) = X1(2+var)*pref/10^5;            % Change unit of
pressure from Pa to bara

    % Calculate gas formaiton volume factor at tubing node of Segment
i+1
    p = X1(1+var);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
        % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    yBo = (p-p1)/dp * (xBo(i1)-xBo(i2)) + xBo(i1);
    Bo(3*i+1) = yBo;


    % Calculate gas formaiton volume factor at annular node of
Segment i+1
    p = X1(2+var);
    [i1, p1, i2, p2] = interpolate(p);
```

```
    dp = p1-p2;
        % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    yBo = (p-p1)/dp * (xBo(i1)-xBo(i2)) + xBo(i1);
    Bo(3*i+2) = yBo;


    % Gas formaiton volume factor at inlet node = oil formaiton
volume factor in the reservoir
     Bo(3*i+3) = Bo_res(i+1);

end

X1(num_var-5) = X1(num_var-5)*pref/10^5;              % Change unit of
pressure from Pa to bara
X1(num_var-4) = X1(num_var-4)*pref/10^5;              % Change unit of
pressure from Pa to bara

% Calculate gas formaiton volume factor at tubing node of Segment N
    p = X1(num_var-5);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    yBo = (p-p1)/dp * (xBo(i1)-xBo(i2)) + xBo(i1);
    Bo(Nodes-1) = yBo;

% Calculate gas formaiton volume factor at annular node of Segment N
    p = X1(num_var-4);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    yBo = (p-p1)/dp * (xBo(i1)-xBo(i2)) + xBo(i1);
    Bo(Nodes) = yBo;

func = Bo;
```

generateBores.m

```
% Calculate pressure-dependent oil formation volume factors

%Input:
%
%N          :   Number of segments
%pres       :   Reservoir pressures at reservoir nodes
%pb         :   Bubblepoint pressure
%
%Return:
%Array containing oil formation volume factors at inlet nodes

function func = generateBores(N,pres,pb)

pres_temp = pres/10^5;          % Change unit of pressure from Pa to
bara
Bo_res = zeros(1,N-1);

for i=1:N-1

    p = pres_temp(i);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
        % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    yBo = (p-p1)/dp * (xBo(i1)-xBo(i2)) + xBo(i1);
    Bo_res(i) = yBo;

end

func = Bo_res;
```

generateflow.m

```
% Generate flow rates for use in temperature calculations and convert
from m^3/d to m^3/s

q = zeros(1,bridges);

for i=0:N-1
    if i~=N-1        % Segment 1 to N-1
        q(i*4+1) = tubingFlowrates(i+1)/(60*60*24);          % In
tubing
        q(i*4+2) = slotFlowRates(i+1)/(60*60*24);            % In
annular-to-tubing flow
        q(i*4+3) = annularFlowrates(i+1)/(60*60*24);         % In
annulus
        q(i*4+4) = inflowRates(i+1)/(60*60*24);              % In
inflow
    elseif i==N-1    % Segment N
        q(i*4+1) = tubingFlowrates(i+1)/(60*60*24);          % In
tubing
        q(i*4+2) = slotFlowRates(i+1)/(60*60*24);            % In
annular-to-tubing flow
    end
end
```

generateFractions.m

```
% Generate liquid holdups for use in temperature calculations

Lfrac = zeros(1,bridges);

for i=0:N-1
    if i~=N-1            % Segment 1 to N-1
        Lfrac(i*4+1) = tubingFractions(i+1);        % In tubing
        Lfrac(i*4+2) = slotFractions(i+1);          % In annular-to-
tubing flow
        Lfrac(i*4+3) = annularFractions(i+1);       % In annulus
        Lfrac(i*4+4) = alpha_res(i+1);              % In inflow
    elseif i==N-1        % Segment N
        Lfrac(i*4+1) = tubingFractions(i+1);        % In tubing
        Lfrac(i*4+2) = slotFractions(i+1);          % In annular-to-
tubing flow
    end
end
```

generatekappa.m

```
% Generate overall heat transfer coefficients

% Calculate dimensionless variables
for i=1:N-1      % Segment 1 to N-1

% Reynolds numbers of fluid in tubing
Re_No(i) =
tubingFlowrates(i)/60/60/24/(pi*ri(i)^2)*(2*ri(i))*rho2P((i-
1)*4+2)/mu2P((i-1)*4+2);
% Reynolds numbers of fluid in annulus
Re_ann(i) = annularFlowrates(i)/60/60/24/(pi*ro(i)^2-
pi*ri(i)^2)*(4*(pi*ro(i)^2-
pi*ri(i)^2)/(2*pi*ro(i)+2*pi*ri(i)))*rho2P((i-1)*4+2)/mu2P((i-
1)*4+2);
% Prandtl numbers
Pr_No(i) = Co*mu2P((i-1)*4+2)/h_fl;

end
% Segment N
Re_No(N) =
tubingFlowrates(N)/60/60/24/(pi*ri(N)^2)*(2*ri(N))*rho2P((N-
1)*4+2)/mu2P((N-1)*4+2);
Pr_No(N) = Co*mu2P((N-1)*4+2)/h_fl;

% Calculate overall heat transfer coefficients

for i=1:N-1      % For Segment 1 to N-1

if Re_ann(i) < 3000          % For laminar flows
    h_ann(i) = 3.656*h_fl/(ro(i)-ri(i));
% Heat transfer coefficient of fluid in annulus
else                         % For turbulence flows
    h_ann(i) = 0.023*Re_ann(i)^0.8*Pr_No(i)^0.33*h_fl/(ro(i)-ri(i));
end
    U_ann(i) = (x_cem/k_cem+x_case/k_case+1/h_ann(i))^(-1);
% Overall heat transfer coefficient of fluid in annulus

if Re_No(i) < 3000           % For laminar flows
    h_fluid(i) = 3.656*h_fl/2/ri(i);
% Heat transfer coefficient of fluid in tubing
else                         % For turbulence flows
    h_fluid(i) = 0.023*Re_No(i)^0.8*Pr_No(i)^0.33*h_fl/2/ri(i);
end
    U_tube(i) = (x_tube/k_tube+1/h_fluid(i))^(-1);
% Overall heat transfer coefficient of fluid in tubing

% Calculate coefficients over the area for radial heat transfer
if bindex(4*(i-1)+2)~=0              % Flow throught slots
    Kappa_t(i) = gam_tube*U_tube(i)*2*pi*ri(i)*L(i)*Tref;
% Heat transfer coefficient of fluid in tubing
else                                 % No slots -- no annular-to-
tubing flow
    Kappa_t(i) = 1*U_tube(i)*2*pi*ri(i)*L(i)*Tref;
```

```
end
Kappa_a(i) = gam_ann*U_ann(i)*2*pi*ro(i)*L(i)*Tref;
% Heat transfer coefficient of fluid in annulus

end

% For Segment N
h_fluid(N) = 0.023*Re_No(N)^0.8*Pr_No(i)^0.33*h_fl/2/ri(N);
% Heat transfer coefficient of fluid in tubing
U_tube(N) = (x_tube/k_tube+1/h_fluid(N))^(-1);
Kappa_t(N) = U_tube(N)*2*pi*ri(N)*L(i)*Tref;

% To exclude heat transfer between fluid and surroundings, type:
% Kappa_a = Kappa_a*0;
% Kappa_t = Kappa_t*0;
```

generatemu.m

```
% Calculate pressure/temperature dependent viscosities

%Input:
%
%T_temp      :   Temperatures used in isothermal calculations
%X1          :   Unknown parameters at each iteration
%Tres        :   Reservoir temperatures at reservoir nodes
%pref        :   Reference pressure
%pb          :   Bubblepoint pressure
%N           :   Number of segments
%num_var     :   Number of unknowns in isothermal calculations
%num_varT    :   Number of unknown temperatures
%Nodes       :   Number of nodes
%bridges     :   Number of bridges
%Rs          :   Gas solubilities
%mu2P_res    :   Two-phase viscosities at reservoir conditions
%mu_res      :   Viscosities of both phases at reservoir conditions
%
%Return:
%mu2P        :   Two-phase viscosities in every bridge in the network
%mu          :   Viscosities of both phases at every node in the
network

function [func1 func2] =
generatemu(T_temp,X1,Tres,pref,pb,N,num_var,num_varT,Nodes,bridges,Rs
,mu2P_res,mu_res)

pb_temp=pb;                                    % Bubblepoint pressure in
bara
pb = pb*1e2;                                   % Bubblepoint pressure in
kPa

mu = zeros(2,Nodes);                           % Viscosities of both
phases (1=liquid, 2=gas)
mu_od = zeros(1,Nodes);                        % Dead-oil viscosities
mu_sat = zeros(1,Nodes);                       % Saturated oil
viscosities
gasmix;

for i=0:N-2

        var = i*9;

        X1(1+var) = X1(1+var)*pref/10^3;       % Change unit of
pressure from Pa to kPa
        X1(2+var) = X1(2+var)*pref/10^3;       % Change unit of
pressure from Pa to kPa

% Calculate viscosity for the gas phase (from curve-fit to values
from EOS)
        X_temp(1) = X1(1+var)/10^2;            % Change unit of
pressure from kPa to bara
```

```
        X_temp(2) = X1(2+var)/10^2;                    % Change unit of
pressure from kPa to bara

    % Tubing node - calculating viscosity (gas phase)
    p = X_temp(1);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,3*i+1) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2)) +
Viscosity_Vap(i1);
    mu(1,3*i+1) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2)) +
Viscosity_Liq(i1);;


    % Annular node - calculating viscosity (gas phase)
    p = X_temp(2);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,3*i+2) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2)) +
Viscosity_Vap(i1);
    mu(1,3*i+2) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2)) +
Viscosity_Liq(i1);


    % Reservoir node - calculating viscosity (gas phase)
    mu(2,3*i+3) = mu_res(2,i+1);
    mu(1,3*i+3) = mu_res(1,i+1);

end

    X1(num_var-5) = X1(num_var-5)*pref/10^3;          % Change unit of
pressure from Pa to kPa
    X1(num_var-4) = X1(num_var-4)*pref/10^3;          % Change unit of
pressure from Pa to kPa

    % Calculate viscosity for the gas phase (from curve-fit to values
from EOS)
        X_temp(1) = X1(num_var-5)/10^2;               % Change unit of
pressure from kPa to bara
        X_temp(2) = X1(num_var-4)/10^2;               % Change unit of
pressure from kPa to bara

    % Tubing node - calculating viscosity (gas phase)
    p = X_temp(1);
    [i1, p1, i2, p2] = interpolate(p);
```

```
    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,Nodes-1) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2))
+ Viscosity_Vap(i1);
    mu(1,Nodes-1) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2))
+ Viscosity_Liq(i1);


    % Annular node - calculating viscosity (gas phase)
    p = X_temp(2);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,Nodes) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2)) +
Viscosity_Vap(i1);
    mu(1,Nodes) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2)) +
Viscosity_Liq(i1);

mu2P = zeros(1,bridges);            % Generate two-phase viscosities
in every bridge

for i=0:N-2

    var = i*9;

    mu2P(i*4+1) = mu(1,3*i+1)*X1(var+7) + mu(2,3*i+1)*(1-X1(var+7));
% Tubing bridge of Segment i+1
    mu2P(i*4+2) = mu(1,3*i+2)*X1(var+8) + mu(2,3*i+2)*(1-X1(var+8));
% Annulus-to-tubing bridge of Segment i+1
    mu2P(i*4+3) = mu(1,3*i+2)*X1(var+9) + mu(2,3*i+2)*(1-X1(var+9));
% Annular bridge of Segment i+1
    mu2P(i*4+4) = mu2P_res(i+1);
% Inlet bridge of Segment i+1

end

mu2P(bridges-1) = mu(1,Nodes-1)*X1(num_var-1) + mu(2,Nodes-1)*(1-
X1(num_var-1));        % Tubing bridge of Segment N
mu2P(bridges) = mu(1,Nodes)*X1(num_var) + mu(2,Nodes)*(1-
X1(num_var));                  % Annulus-to-tubing bridge of Segment
N

func1 = mu2P*10^(-3);
func2 = mu*10^(-3);
```

generatemu2P.m

```
% Calculate two-phase fluid viscosities

%Input:
%
%mu_res      :   Viscosities of both phases at reservoir conditions
%N           :   Number of segments
%alpha_res   :   Liquid holdups at reservoir conditions
%pres        :   Reservoir pressures at reservoir nodes
%pb          :   Bubblepoint pressure
%
%Return:
%Array containing two-phase viscosities at inlet nodes

function func = generatemu2P(mu_res,N,alpha_res,pres,pb)

mu2P_res = zeros(1,N-1);

for i=1:N-1

    if pres(i)/1e5 < pb
        mu2P_res(i) = mu_res(1,i)*alpha_res(i) + mu_res(2,i)*(1-
alpha_res(i));
    else
        mu2P_res(i) = mu_res(1,i);
    end

end

func = mu2P_res;
```

generatemures.m

```
% Calculate pressure/temperature dependent oil viscosities

%Input:
%
%Tres        :   Reservoir temperatures
%pres        :   Reservoir pressures at reservoir nodes
%pb          :   Bubblepoint pressure
%N           :   Number of segments
%Rs_res      :   Gas solubilities in reservoir
%
%Return:
%Array containing oil viscosities at inlet nodes

function func = generatemures(Tres,pres,pb,N,Rs_res)

pres_temp = pres/10^5;              % Change unit of pressure from Pa to
bara
pb_temp=pb;                         % Bubblepoint pressure in bara
pb = pb*1e2;                        % Bubblepoint pressure in kPa
pres = pres/1e3;                    % Change unit of pressure from Pa to
kPa

mu_res = zeros(2,N-1);              % Viscosities of both phases in
reservoir (1=liquid, 2=gas)
mu_od = zeros(1,N-1);               % Dead-oil viscosities
mu_sat = zeros(1,N-1);              % Saturated oil viscosities

gasmix;

for i=0:N-2

    p = pres_temp(i+1);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;

    % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    % calculating viscosity (gas phase)
    mu_res(2,i+1) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2))
+ Viscosity_Vap(i1);
    mu_res(1,i+1) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2))
+ Viscosity_Liq(i1);

end

func = mu_res*10^(-3);
```

generatepdrop.m

```
% Generate pressure drop between nodes

deltaP_t = zeros(1,N);          % Pressure drop between nodes in
tubing
deltaP_a = zeros(1,N-1);        % Pressure drop between nodes in
annulus

for i=1:N-1
        deltaP_t(i) = (p_tubing(i+1)-p_tubing(i))/Tref;
        deltaP_a(i) = (p_annulus(i+1)-p_annulus(i))/Tref;
end

deltaP_t(N) = (pbh*1e-5-p_tubing(N))/Tref;

% To exclude Joule-Thompson effect, type:
% deltaP_t = deltaP_t*0;
% deltaP_a = deltaP_a*0;
```

generateResprop.m

```
% Calculate pressure-dependent black-oil properties at reservoir
conditions

%Input:
%
%N          :   Number of segments
%pres       :   Reservoir pressures at reservoir nodes
%pb         :   Bubblepoint pressure
%
%Return:
%Bo_res     :   Oil formation volume factors at inlet nodes
%Bg_res     :   Gas formation volume factors at inlet nodes
%Rs_res     :   Gas solubilities at inlet nodes

function [func1 func2 func3] = generateResprop(N,pres,pb)

pres_temp = pres/10^5;          % Change unit of pressure from Pa to
bara
Bo_res = zeros(1,N-1);
Bg_res = zeros(1,N-1);
Rs_res = zeros(1,N-1);
gasmix;

for i=1:N-1

    p = pres_temp(i);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;

    % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    % calculating Bo
    yBo = (p-p1)/dp * (xBo(i1)-xBo(i2)) + xBo(i1);
    Bo_res(i) = yBo;

    % calculating Bg
    yBg = (p-p1)/dp * (xBg(i1)-xBg(i2)) + xBg(i1);
    Bg_res(i) = yBg;

    % calculating Rs
    Rs_res(i) = (p-p1)/dp * (xRs(i1)-xRs(i2)) + xRs(i1);

end

func1 = Bo_res;
func2 = Bg_res;
func3 = Rs_res;
```

generaterho.m

```
% Calculate pressure-dependent densities

%Input:
%
%X1          :   Unknown parameters at each iteration
%pres        :   Reservoir pressures at reservoir nodes
%rho2P_res   :   Two-phase densities at reservoir conditions
%rho_res     :   Densities of both phases at reservoir conditions
%pref        :   Reference pressure
%pb          :   Bubblepoint pressure
%N           :   Number of segments
%num_var     :   Number of unknowns in isothermal calculations
%Nodes       :   Number of nodes
%bridges     :   Number of bridges
%
%Return:
%rho2P       :   Two-phase densities in every bridge in the network
%rho         :   Densities of both phases at every node in the network

function [func1,func2] =
generaterho(X1,pres,rho2P_res,rho_res,pref,pb,N,num_var,Nodes,bridges
)

rho_res = rho_res/10^(3);                          % Change unit of
pressure from kg/m^3 to g/cm^3
rho2P_res = rho2P_res/10^(3);                      % Change unit of
pressure from kg/m^3 to g/cm^3

% Calculate densities of each phase at every node

rho = zeros(2,Nodes);
gasmix;

for i=0:N-2

    var = i*9;

    X1(1+var) = X1(1+var)*pref/10^5;               % Change unit of
pressure from Pa to bara
    X1(2+var) = X1(2+var)*pref/10^5;               % Change unit of
pressure from Pa to bara
% Tubing node
    p = X1(1+var);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    rho(2,3*i+1) = (p-p1)/dp * (Density_Vap(i1)-Density_Vap(i2)) +
Density_Vap(i1);
```

```
    rho(1,3*i+1) = (p-p1)/dp * (Density_Liq(i1)-Density_Liq(i2)) +
Density_Liq(i1);
% Annular node
    p = X1(2+var);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    rho(2,3*i+2) = (p-p1)/dp * (Density_Vap(i1)-Density_Vap(i2)) +
Density_Vap(i1);
    rho(1,3*i+2) = (p-p1)/dp * (Density_Liq(i1)-Density_Liq(i2)) +
Density_Liq(i1);
% Inlet node
    rho(1,3*i+3) = rho_res(1,i+1);
    rho(2,3*i+3) = rho_res(2,i+1);

end

X1(num_var-5) = X1(num_var-5)*pref/10^5;        % Change unit of
pressure from Pa to bara
X1(num_var-4) = X1(num_var-4)*pref/10^5;        % Change unit of
pressure from Pa to bara

% Tubing node
    p = X1(num_var-5);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    rho(2,Nodes-1) = (p-p1)/dp * (Density_Vap(i1)-Density_Vap(i2)) +
Density_Vap(i1);
    rho(1,Nodes-1) = (p-p1)/dp * (Density_Liq(i1)-Density_Liq(i2)) +
Density_Liq(i1);
% Annular node
    p = X1(num_var-4);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    rho(2,Nodes) = (p-p1)/dp * (Density_Vap(i1)-Density_Vap(i2)) +
Density_Vap(i1);
    rho(1,Nodes) = (p-p1)/dp * (Density_Liq(i1)-Density_Liq(i2)) +
Density_Liq(i1);


% Generate two-phase densities in every bridge
```

```
rho2P = zeros(1,bridges);

for i=0:N-2

    var = i*9;

    rho2P(i*4+1) = rho(1,3*i+1)*X1(var+7) + rho(2,3*i+1)*(1-
X1(var+7));             % Tubing bridge of Segment i+1
    rho2P(i*4+2) = rho(1,3*i+2)*X1(var+8) + rho(2,3*i+2)*(1-
X1(var+8));             % Annulus-to-tubing bridge of Segment i+1
    rho2P(i*4+3) = rho(1,3*i+2)*X1(var+9) + rho(2,3*i+2)*(1-
X1(var+9));             % Annular bridge of Segment i+1
    rho2P(i*4+4) = rho2P_res(i+1);
% Inlet bridge of Segment i+1

end

rho2P(bridges-1) = rho(1,Nodes-1)*X1(num_var-1) + rho(2,Nodes-1)*(1-
X1(num_var-1));             % Tubing bridge of Segment N
rho2P(bridges) = rho(1,Nodes)*X1(num_var) + rho(2,Nodes)*(1-
X1(num_var));                   % Annulus-to-tubing bridge of
Segment N

rho = rho*10^(3);               % Change unit of pressure from g/cm^3
to kg/m^3
rho2P = rho2P*10^(3);           % Change unit of pressure from g/cm^3
to kg/m^3

func1 = rho2P;
func2 = rho;
```

generateRhores.m

```
% Calculate pressure-dependent densities

%Input:
%
%pres        :   Reservoir pressures at reservoir nodes
%pb          :   Bubblepoint pressure
%N           :   Number of segments
%alpha_res   :   Liquid holdups in reservoir
%
%Return:
%rho_res     :   Densities of both phases at reservoir conditions
%rho_res2P   :   Two-phase fluid densities at reservoir conditions

function [func1 func2] = generateRhores(pres,pb,N,alpha_res)

pres_temp = pres/10^5;          % Change unit of pressure from Pa to
bara

% Calculate densities of each phase at inlet nodes
% Expression obtained from curve-fitting of pre-generated values
using an EOS

rho_res = zeros(2,N-1);
gasmix;

for i=1:N-1

    p = pres_temp(i);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;

    % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    % calculating density (gas phase)
    rho_res(2,i) = (p-p1)/dp * (Density_Vap(i1)-Density_Vap(i2)) +
Density_Vap(i1);
    rho_res(1,i) = (p-p1)/dp * (Density_Liq(i1)-Density_Liq(i2)) +
Density_Liq(i1);


end

% Calculate two-phase densities at inlet nodes

rho2P_res = zeros(1,N-1);

for i=1:N-1
```

```
    rho2P_res(i) = rho_res(1,i)*alpha_res(i) + rho_res(2,i)*(1-
alpha_res(i));

end

rho_res = rho_res*10^(3);                    % Change unit of pressure
from g/cm^3 to kg/m^3
rho2P_res = rho2P_res*10^(3);                % Change unit of pressure
from g/cm^3 to kg/m^3

func1 = rho2P_res;
func2 = rho_res;
```

generateRs.m

```
% Calculate pressure-dependent gas solubilities

%Input:
%
%X1          :   Unknown parameters at each iteration
%pres        :   Reservoir pressures at reservoir nodes
%Rs_res      :   Gas solubilities in reservoir
%pref        :   Reference pressure
%pb          :   Bubblepoint pressure
%N           :   Number of segments
%num_var     :   Number of unknowns
%Nodes       :   Number of nodes
%
%Return:
%Array containing gas solubility at every node in the network

function func = generateRs(X1,pres,Rs_res,pref,pb,N,num_var,Nodes)

Rs = zeros(1,Nodes);
gasmix;

for i=0:N-2

    var = i*9;

    X1(1+var) = X1(1+var)*pref/10^5;            % Change unit of
pressure from Pa to bara
    X1(2+var) = X1(2+var)*pref/10^5;            % Change unit of
pressure from Pa to bara
% Tubing node
    p = X1(1+var);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    Rs(3*i+1) = (p-p1)/dp * (xRs(i1)-xRs(i2)) + xRs(i1);

    % Annular node
    p = X1(2+var);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    Rs(3*i+2) = (p-p1)/dp * (xRs(i1)-xRs(i2)) + xRs(i1);
```

```
    Rs(3*i+3) = Rs_res(i+1);
end

X1(num_var-5) = X1(num_var-5)*pref/10^5;          % Change unit of
pressure from Pa to bara
X1(num_var-4) = X1(num_var-4)*pref/10^5;          % Change unit of
pressure from Pa to bara

% % Calculate gas solubility at tubing node of Segment N
    p = X1(num_var-5);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    Rs(Nodes-1) = (p-p1)/dp * (xRs(i1)-xRs(i2)) + xRs(i1);


 % % Calculate gas solubility at tubing node of Segment N+1
    p = X1(num_var-4);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    Rs(Nodes) = (p-p1)/dp * (xRs(i1)-xRs(i2)) + xRs(i1);


func = Rs;
```

generateRsres.m

```
% Calculate pressure-dependent gas solubilities

%Input:
%
%N          :   Number of segments
%pres       :   Reservoir pressures at reservoir nodes
%pb         :   Bubblepoint pressure
%
%Return:
%Array containing gas solubilities at inlet nodes

function func = generateRsres(N,pres,pb)

pres_temp = pres/10^5;                  % Change unit of pressure from Pa
to bara
Rs_res = zeros(1,N-1);

gasmix;

for i=1:N-1

    p = pres_temp(i);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;

    % eradicates the possibility of division by zero
    if dp==0
        dp = 2;
    end

    % calculating density (gas phase)
    Rs_res(i) = (p-p1)/dp * (xRs(i1)-xRs(i2)) + xRs(i1);

end

func = Rs_res;
```

generateT.m

```
% Generate wellbore temperature to be used in isothermal calculations

% Temperatures in annulus and tubing are equal to the reservoir
temperature
% of the corresponding segment
T_temp = zeros(1,num_varT);
for i = 0:N-1
    T_temp(i*2+1) = Tres(i+1);        %Annulus
    T_temp(i*2+2) = Tres(i+1);        %Tubing
end
```

generateTres.m

```
% Generate reservoir temperatures

Tres = 100*ones(1,N);

% For Case 2-A (well with linearly decreasing reservoir temperature),
% Tres(1) = 100;                % Temperature at the toe
% sumL = 0;
% for i = 2:N                   % Temperature decreases at the rate
0.01 C/m
%     sumL = sumL+L(i);
%     Tres(i) = 100-sumL*0.01;
% end

% For Case 2-B (well with linearly increasing reservoir temperature),
% Tres(1) = 80;                 % Temperature at the toe
% sumL = 0;
% for i = 2:N                   % Temperature increases at the rate
0.01 C/m
%     sumL = sumL+L(i);
%     Tres(i) = 80+sumL*0.01;
% end

% For Case 2-C (well with two temperature zones),
% Tres = 80*ones(1,N);          % Temperatures in Zone 1
%
% for i=101:200                 % Temperatures in Zone 2
%     Tres(i) = 100;
% end
```

guess.m

```
% Initial guessed values of unknown parameters

% Generate distances from toe of well (e.g. 10 m, 20 m,..., 10xN m
for 10-m segments)
lengths = [];
temp = 0;
for i=1:N

    lengths(i) = L(i) + temp;
    temp = lengths(i);

end

% Generate initial guessed unknown parameters
X1_temp =
guessGenerator(pres,I,L,alpha_res,pref,qref,pbh,N,num_var,ri,ro);

% Let unknowns (X1) used in the functions equal initial guessed
values
X1 = X1_temp;
```

guessGenerator.m

```
%Input:
%
%pres        :   Reservoir pressures at reservoir nodes
%I           :   Pre-calculated coefficient for inflow equations
%L           :   Segment lengths
%alpha_res   :   Liquid holdups in reservoir inflows
%pref        :   Reference pressure
%qref        :   Reference flow rate
%pbh         :   Bottomhole pressure
%N           :   Number of segments
%num_var     :   Number of unknowns
%ri          :   Tubing diameter
%ro          :   Annular outer diameter
%
%Return:
%Array containing initial guessed values of unknown parameters

function func =
guessGenerator(pres,I,L,alpha_res,pref,qref,pbh,N,num_var,ri,ro)

a = 0;
b = 0;
% a is the sum of all segment lenghts up to the considered segment
for i=1:N

    a = a + L(i);

end
sum = a;        % The sum of all segment lenghts = the total length
of the well

% Calculate guessed unknown parameters
% Pressures assumed linearly decreasing from reservoir to heel of
well
% Flow rates proportional to cross-sectional areas of the flow paths
% Liquid holdups assumed equal to the values in the reservoir of the
corresponding segments

guess = [];

p3k_1 = 0;              % Pressure in annulus
A1 = zeros(1,N);        % Tubing area
A2 = zeros(1,N-1);      % Annular area

for i=0:N-2

% Calculate cross-sectional areas
A1(i+1) = pi*(ri(i+1)^2);
A2(i+1) = pi*(ro(i+1)^2-ri(i+1)^2);

% Calculate guessed pressures
```

```
    if(i == 0)                      % Segment 1

        a = a - L(i+1);             % a = well lenght - length of Segment
1
        b = b + L(i+1);             % b = length of Segment 1

        p3k_1 = (a*pres(i+1) + b*pbh)/(sum*pref);       % Pressure in
annulus of Segment 1

        guess(1 + 9*i) = p3k_1;             % Guessed pressure in
annulus of Segment 1
        guess(2 + 9*i) = guess(1+9*i);      % Guessed pressure in
tubing of Segment 1 = pressure in annulus

    else                            % Segment 2 to N-1

        p3k_1 = (a*pres(i+1) + b*pbh)/(sum*pref);       % Pressure in
annulus of Segment i+1

        guess(1 + 9*i) = p3k_1;                         % Guessed
pressure in annulus
        guess(2 + 9*i) = guess(1 + 9*i);                % Guessed
pressure in tubing = pressure in annulus

    end

% Calculate guessed flow rates

    q3k_3k_1 = I(i+1)*(pref/qref)*(pres(i+1)/pref - p3k_1);     %
Inflow of Segment i+1

    if(i == 0)                      % Segment 1

        guess(6 + 9*i) = q3k_3k_1;
% Guessed inflow rate
        guess(3 + 9*i) = ((A1(i+1)/(A1(i+1)+A2(i+1)))*q3k_3k_1);
% Guessed tubing flow rate
        guess(4 + 9*i) = guess(3 + 9*i);
% Guessed slot/valve flow rate = tubing flow rate
        guess(5 + 9*i) = ((A2(i+1)/(A1(i+1)+A2(i+1)))*q3k_3k_1);
% Guessed annular flow rate

    else                            % Segment 2 to N-1

    guess(6 + 9*i) = q3k_3k_1;
% Guessed inflow rate
    guess(3 + 9*i) = (A1(i+1)/(A1(i+1)+A2(i+1)))*(q3k_3k_1 +
guess(9*(i-1)+5)) + guess(9*(i-1)+3);       % Guessed tubing flow rate
= tubing rate of Segment i+1 + tubing rate of Segment i
    guess(4 + 9*i) = (A1(i+1)/(A1(i+1)+A2(i+1)))*(q3k_3k_1 +
guess(9*(i-1)+5));                          % Guessed slot/valve flow
rate
```

```
    guess(5 + 9*i) = (A2(i+1)/(A1(i+1)+A2(i+1)))*(q3k_3k_1 +
guess(9*(i-1)+5));                          % Guessed annular flow
rate

    end

% Calculate guessed liquid holdups

    guess(7 + 9*i) = alpha_res(i+1);     % Guessed liquid holdup in
tubing
    guess(8 + 9*i) = alpha_res(i+1);     % Guessed liquid holdup in
slot/valve
    guess(9 + 9*i) = alpha_res(i+1);     % Guessed liquid holdup in
annulus

    if(i==N-2)                           % Segment N-1

        guess(num_var - 5) = pbh/pref;                              %
Guessed pressure in tubing of Segment N = bottomhole pressure
        guess(num_var - 4) = pbh/pref;                              %
Guessed pressure in annulus of Segment N = bottomhole pressure
        guess(num_var - 3) = guess(3 + 9*i) + guess(5 + 9*i);       %
Guessed flow rate in tubing of Segment N
        guess(num_var - 2) = guess(5 + 9*i);                        %
Guessed flow rate in slot/valve of Segment N = flow rate in annulus
of Segment N-1
        guess(num_var - 1) = alpha_res(i+1);                        %
Guessed liquid holdup in tubing of Segment N
        guess(num_var)     = alpha_res(i+1);                        %
Guessed liquid holdup in annulus of Segment N

        break;
    end

% Update a and b values for calculating the parameters of the next
segment
     a = a - L(i+2);
     b = b + L(i+2);

end

func = guess;
```

guessGeneratorT.m

```
% Generate initial guessed unknown temperatures

%Input:
%
%Tres        :   Reservoir temperatures
%L           :   Segment lengths
%Tbh         :   Reference temperature
%Tbh         :   Guessed bottomhole temperature
%N           :   Number of segments
%
%Return:
%Initial guessed unknown temperatures

function func = guessGeneratorT(Tres,L,Tref,Tbh,N)

a = 0;
b = 0;

% Sum of all the segment legths -- the length of the well
for i=1:N
    a = a + L(i);
end
sum = a;

% Generate initial temperature values assuming linearly increasing
from toe
% to heel of well
guess = [];
T_1 = 0;
for i=0:N-2
        a = a - L(i+1);
        b = b + L(i+1);
    if(i == 0)                      % Segment 1
        guess(1 + 2*i) = Tres(i+1)/Tref; % Temperature at tubing node
    else                            % Segment 2 to N-1

        T_1 = (a*Tres(i+1) + b*Tbh)/(sum*Tref);
        guess(2 + 2*(i-1)) = T_1; % Temperature at annular node
        guess(1 + 2*i) = guess(2 + 2*(i-1));% Temperaturetubing node
    end

end

% Segment N
guess(2*N-2) = Tbh/Tref;                % Temperature at annular node
guess(2*N-1) = Tbh/Tref;                % Temperature at tubing node
guess(2*N) = Tbh/Tref;                  % Temperature at the bottomhole

func = guess;
```

input_alpha.m

```
% Calculate liquid holdups in the reservoir

alpha_res =    zeros(1,N-1);  %all gas

for i=1:N-1          % Generate alpha_res for N-1 segments

    if pres(i)/1e5 < pb          % For pressures below the bubblepoint
pressures
        alpha_res(i) = kro(i)/mu_res(1,i) /
(kro(i)/mu_res(1,i)+krg(i)/mu_res(2,i));
    else                         % For pressures above the bubblepoint
pressures
        alpha_res(i) = 1;
    end

end
```

input_c.m

```
% Input slot/valve discharge coefficients

c       =       10*ones(1,N);                   % Generate discharge
coefficients of annular-to-tubing flows in all N segments

% To change discharge coefficients of annular-to-tubing flows in some
segments e.g. Segment 200
% c(200)  =       0.6e6;


% For case 3 (well with multiple inflow control valves),
% c(95) =    0.5e7;                           % Discharge coefficient of
ICV 2
% c(100) =    0.5e7;                          % Discharge coefficient of
ICV 1 before being plugged with asphaltene precipitate
% After the plugging resulting in reversed flow in 3 segments after
the 95th-setment valve
% c(95) =    0.5e7;                           % Discharge coefficient of
ICV 2
% c(100) =    100e7;                          % Discharge coefficient of
ICV 1
```

input_data.m

```
% Input data

N           = 100;                  % Number of segments
L           = 5;                    % Segment length (when all
segments are of the same length or modify input_L.m)
threshold   = 1e-12;                % Tolerance value to check for
convergence
stop        = 100;                  % Number of iterations before
stop iteratiing
re          = 20;                   % Drainage radius (m)
ri          = 5*0.0254/2;           % Inner radius (m)
ro          = ri+0.02;              % Outer radius (m)
ri          = ri*ones(1,N);         % Generate ri for all N segments
ro          = ro*ones(1,N-1);       % Generate ro for all N-1
segments
r_temp      = ri;                   % Inner radius of annulus (m)
pres        = 267;                  % Reservoir pressure (bara) (when
all segments have the same reservoir pressures or modify input_p.m)
pbh         = 267;                  % Bottomhole pressure (bara)
pb          = 265;                  % Bubblepoint pressure (bara)
slot_den    = 30000;                % Slot density (slots/m)
slot_L      = 0.01;                 % Slot length (m)
slot_W      = 0.001;                % Slot width (m)
So          = 1*ones(1,N-1);        % Oil saturation (when So's are
equal for all segments)
n_k         = 2*ones(1,N-1);        % Exponent for calculating
relative permeabilities (when n_k's are equal for all segments)
generateTres;                       % Generate reservoir temperatures
delta_pbh   = 3e5;

num_var = 9*N-3;                    % Calculate number of unknowns in
isothermal calculations
Nodes = 3*N-1;                      % Calculate number of nodes
bridges = 4*N-2;                    % Calculate number of bridges
num_varT = 2*N;                     % Calculate number of temperature
unknowns in temperature calculations

pbh = pbh*10^5;                     % Change unit of pressure from
bara to Pa

generateT;                          % Generate wellbore temperature
to be used in isothermal calculations

% %For Case 2-C (well with two temperature zones), different-pressure
case,
% ri          = 4*0.0254/2;           % Inner radius (m)
% ro          = ri+0.02;              % Outer radius (m)
% ri          = ri*ones(1,N);         % Generate ri for all N
segments
% ro          = ro*ones(1,N-1);       % Generate ro for all N-1
segments
% r_temp      = ri;                   % Inner radius of annulus (m)
% for i = 101:N-1
```

```
%     r_temp(i)=0;                          % Annulus used as tubing in the
well in Zone 2
% end
% for i = 101:N-1
%     ro(i)    = 5*0.0254/2;                % Outer radius of well in Zone
2
% end
% pres1         = 370;                      % Pressure in Zone 1
% pres2         = 360;                      % Pressure in Zone 2

% For case 3 (well with multiple inflow control valves),
% N = 100;                                  % Well divided into 100
segments

% For Case 4 (well with restricted flow in annulus), assign
% r_temp(170)   = ri(169)+0.005;
```

input_dataT.m

```
% Recalculate pressure/temperature dependent viscosities

%Input:
%
%XT          :   Unknown temperatures (converged)
%X1          :   Array containing pressure parameters
%Tres        :   Reservoir temperatures at reservoir nodes
%pref        :   Reference pressure
%pb          :   Bubblepoint pressure
%N           :   Number of segments
%num_var     :   Number of unknowns in isothermal calculations
%num_varT    :   Number of unknown temperatures
%Nodes       :   Number of nodes
%bridges     :   Number of bridges
%Rs          :   Gas solubilities
%mu2P_res    :   Two-phase viscosities at reservoir conditions
%mu_res      :   Viscosities of both phases at reservoir conditions
%
%Return:
%mu2P        :   Two-phase viscosities in every bridge in the network
%mu          :   Viscosities of both phases at every node in the
network

function func =
updatemu(XT,X1,Tres,pres,pref,pb,N,num_var,num_varT,Nodes,bridges,Rs,
mu2P_res,mu_res)

pb_temp=pb;                 % Bubblepoint pressure in bara
pb = pb*1e2;                % Bubblepoint pressure in kPa
XT_toe = Tres(1);           % Temperature at toe of well

mu = zeros(2,Nodes);            % Viscosities of both phases (1=liquid,
2=gas)
mu_od = zeros(1,Nodes);         % Dead-oil viscosities
mu_sat = zeros(1,Nodes);        % Saturated oil viscosities
gasmix;

for i=0:N-2

        var = i*9;

        X1(1+var) = X1(1+var)*pref/10^3;            % Change unit of
pressure from Pa to kPa
        X1(2+var) = X1(2+var)*pref/10^3;            % Change unit of
pressure from Pa to kPa

% Calculate viscosity for the gas phase (from curve-fit to values
from EOS)
        X_temp(1) = X1(1+var)/10^2;                 % Change unit of
pressure from kPa to bara
        X_temp(2) = X1(2+var)/10^2;                 % Change unit of
pressure from kPa to bara
```

```
    % Tubing node - calculating viscosity (gas phase)
    p = X_temp(1);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,3*i+1) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2)) +
Viscosity_Vap(i1);
    mu(1,3*i+1) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2)) +
Viscosity_Liq(i1);


    % Annular node - calculating viscosity (gas phase)
    p = X_temp(2);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,3*i+2) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2)) +
Viscosity_Vap(i1);
    mu(1,3*i+2) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2)) +
Viscosity_Liq(i1);


    % Reservoir node - calculating viscosity (gas phase)
    mu(2,3*i+3) = mu_res(2,i+1);
    mu(1,3*i+3) = mu_res(1,i+1);


end

    X1(num_var-5) = X1(num_var-5)*pref/10^3;          % Change unit of
pressure from Pa to kPa
    X1(num_var-4) = X1(num_var-4)*pref/10^3;          % Change unit of
pressure from Pa to kPa

% Calculate viscosity for the gas phase (from curve-fit to values
from EOS)
        X_temp(1) = X1(num_var-5)/10^2;               % Change unit of
pressure from kPa to bara
        X_temp(2) = X1(num_var-4)/10^2;               % Change unit of
pressure from kPa to bara


        % Tubing node - calculating viscosity (gas phase)
    p = X_temp(1);
    [i1, p1, i2, p2] = interpolate(p);
```

```
    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,Nodes-1) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2))
+ Viscosity_Vap(i1);
    mu(1,Nodes-1) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2))
+ Viscosity_Liq(i1);


    % Annular node - calculating viscosity (gas phase)
    p = X_temp(2);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,Nodes) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2)) +
Viscosity_Vap(i1);
    mu(1,Nodes) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2)) +
Viscosity_Liq(i1);


mu2P = zeros(1,bridges);            % Generate two-phase viscosities
in every bridge

for i=0:N-2

    var = i*9;

    mu2P(i*4+1) = mu(1,3*i+1)*X1(var+7) + mu(2,3*i+1)*(1-X1(var+7));
% Tubing bridge of Segment i+1
    mu2P(i*4+2) = mu(1,3*i+2)*X1(var+8) + mu(2,3*i+2)*(1-X1(var+8));
% Annulus-to-tubing bridge of Segment i+1
    mu2P(i*4+3) = mu(1,3*i+2)*X1(var+9) + mu(2,3*i+2)*(1-X1(var+9));
% Annular bridge of Segment i+1
    mu2P(i*4+4) = mu2P_res(i+1);
% Inlet bridge of Segment i+1

end

mu2P(bridges-1) = mu(1,Nodes-1)*X1(num_var-1) + mu(2,Nodes-1)*(1-
X1(num_var-1));            % Tubing bridge of Segment N
mu2P(bridges) = mu(1,Nodes)*X1(num_var) + mu(2,Nodes)*(1-
X1(num_var));                    % Annulus-to-tubing bridge of
Segment N

func = mu2P*10^(-3);
```

input_K.m

```
% Input absolute permeabilities

K = 0.8*ones(1,N-1);          % Generate reservoir permeabilities for
N-1 segments (K = 0.8 Darcy in this case)

% To change discharge coefficients of annular-to-tubing flows in some
segments e.g. Segment 200
% K(200)    =    0.5;

K = K*10^(-12);               % Change unit of permeability from Darcy
to m^2
```

input_krg.m

```
% Calculate gas relative permeabilities

krg = zeros(1,N-1);

for i=1:N-1          % Generate krg for N-1 segments

    krg(i)      =  (1-So(i))^n_k(i);

end
```

input_kro.m

```
% Calculate oil relative permeabilities

kro = zeros(1,N-1);

for i=1:N-1          % Generate kro for N-1 segments

    kro(i)      =  So(i)^n_k(i);

end
```

input_L.m

```
% Input segment lengths

L =     L*ones(1,N);                % Generate segment lengths of all N
segments

% To change discharge coefficients of annular-to-tubing flows in some
segments e.g. Segment 200
% L(200)    =    20;
```

input_p.m

```
% Input reservoir pressures

pres    =  pres*ones(1,N-1);          % Generate pres for N-1 segments

pres = pres*10^5;                     % Change unit of pressure from
bara to Pa

pref = pres(1);                       % Assign reservoir pressure at
Segment 1 as reference pressure


% For Case 2-C (well with two temperature zones), different-pressure
case,
% pres    =  pres1*ones(1,N-1);       % Reservoir pressure in Zone 1
% for i = 101:N-1                     % Reservoir pressure in Zone 2
%     pres(i) = pres2;
% end
% pres = pres*10^5;                   % Change unit of pressure from
bara to Pa
% pref = pres(1);                     % Assign reservoir pressure at
Segment 1 as reference pressure
```

input_s.m

```
% Input skin factors

s =     1*ones(1,N-1);                  % Generate s for N-1 segments

% To change discharge coefficients of annular-to-tubing flows in some
segments e.g. Segment 200
% s(200)    =    2;
```

interpolate.m

```
function [u, v, w, x] = interpolate(p)
%   calculate_p - calculates the various properties from PVT data
%   SGo, SGg, Rs, Bo, Bg, miu_o, miu_g, sigma_o, z, SGw, Bw, miu_w,
sigma_w

p=200;
% data from PVTsim
gasmix;

for i = 1:22
    if Pressure(i,1) == p
        p1 = Pressure(i,1)
        p2 = Pressure(i,1);
        i1 = i;
        i2 = i;
        break
    else if Pressure(i,1)<p
        p1 = Pressure (i,1);
    else
        p2 = Pressure(i,1);
        i1 = i-1;
        i2 = i;
        break
        end
    end
end
    if i1 == 0
        i1 = 1;
        p1 = Pressure (i1,1);
    end

u=i1;
v=p1;
w=i2;
x=p2;
```

iteration.m

```
% Iterate function calculations to solve for unknowns using Newton-
Raphson
% method

f1 = zeros(1,9);                        % Function matrix for Segment 1
f2 = zeros(1,num_var-(9+6));            % Function matrix for Segment 2 to N-
1
f3 = zeros(1,6);                        % Function matrix for Segment N

j1 = zeros(9,num_var);                  % Jacobian matrix for Segment
1
j2 = zeros(num_var-(9+6),num_var);      % Jacobian matrix for Segment
2 to N-1
j3 = zeros(6,num_var);                  % Jacobian matrix for Segment
N

converge = 0;                           % Convergence value to be compared
with the tolerance value
while(flag)

    % Generating function matrices
    f1 =
f1Generator(X1,I,pres,beta,alpha,B,Bo,Bg,Rs,mu2P,rho2P,alpha_res,f1,p
ref,qref);
    f2 =
f2Generator(X1,beta,alpha,B,I,pres,Bo,Bg,Rs,mu2P,rho2P,alpha_res,f2,p
ref,qref,N,bindex);
    f3 =
f3Generator(X1,beta,B,Bo,Bg,Rs,mu2P,rho2P,f3,pref,pbh,N,num_var,Nodes
,bridges,bindex);

    f = [f1 f2 f3];             % Combine the matrices

    % Generating jacobian matrices
    j1 =
j1Generator(I,X1,beta,alpha,B,j1,Bo,Bg,Rs,rho2P,mu2P,alpha_res,pref,q
ref);
    j2 =
j2Generator(X1,beta,alpha,I,B,j2,Bo,Bg,Rs,rho2P,mu2P,alpha_res,pref,q
ref,N,bindex);
    j3 =
j3Generator(X1,beta,B,j3,Bo,Bg,Rs,rho2P,mu2P,pref,N,num_var,Nodes,bri
dges,bindex);

    jac = [j1;j2;j3];           % Combine the matrices

    % LU factorization -- Inversion of the jacobian matrix

    [L1 U1] = lu(jac);

    L1_INV = inv(L1);
    U1_INV = inv(U1);
```

```
    temp1 = L1_INV*transpose(f);
    temp2 = U1_INV*temp1;

    temp3 = transpose(temp2);

    X2 = X1 - temp3;

    % Checking for convergence.  If checkConvergence finds that the
method
    % converges it will set flag to false and the program will stop.

    [flag, test] = checkConvergence(X1,X2,num_var,threshold);
    converge(sentinelCount+1) = test            % Convergence value of
each iteration

    %Setting Xn = Xn+1 for the next iteration

    X1 = X2;

    %----------RECALCULATE WELLBORE FLUID PROPERTIES BEGIN--------
    Rs = generateRs(X1,pres,Rs_res,pref,pb,N,num_var,Nodes);
% Calculate pressure-dependent gas solubilities
    Bo = generateBo(X1,pres,Bo_res,pref,pb,N,num_var,Nodes);
% Calculate pressure-dependent oil formation volume factors
    Bg = generateBg(X1,pres,Bg_res,pref,pb,N,num_var,Nodes);
% Calculate pressure-dependent gas formation volume factors
    [mu2P mu] =
generatemu(T_temp,X1,Tres,pref,pb,N,num_var,num_varT,Nodes,bridges,Rs
,mu2P_res,mu_res);  % Calculate pressure/temperature dependent
viscosities for each phase and two-phase (TP) fluid
    [rho2P rho] =
generaterho(X1,pres,rho2P_res,rho_res,pref,pb,N,num_var,Nodes,bridges
);                  % Calculate pressure-dependent densities for each
phase and two-phase (TP) fluid

    %----------RECALCULATE WELLBORE FLUID PROPERTIES END--------

    % This if statement makes sure that the iteration stops if
    % the method does not converge whithin a number of iteration
input by the
    % user in input_data.m

    if(sentinelCount == stop)

        disp(' ');
        disp('Did not converge whithin the limitations given!');
        break;

    end

    % Update iteration index (sentinelCount)
    sentinelCount = sentinelCount + 1;
```

```
end
    % Check for imaginary numbers
for i=1:num_var
    if imag(X1)<1e-12    % If the imaginary part of the solution is
less than a value it is negligible
        X1=real(X1);
    else                 % If the imaginary part is large display
"imag"
        disp('imag');
    end
end

% Conversions of the converged variables back to their appropriate
units and
% result display
displayoutput;
```

iterationT.m

```
% Iterate function calculations to solve for temperature unknowns
using
% Newton-Raphson method

f1 = zeros(1,3);                % Function matrix for Segment 1
f2 = zeros(1,num_varT-(4));     % Function matrix for Segment 2 to N-
1
f3 = zeros(1,1);                % Function matrix for Segment N

j1 = zeros(3,num_varT);                 % Jacobian matrix for Segment
1
j2 = zeros(num_varT-(4),num_varT);      % Jacobian matrix for Segment
2 to N-1
j3 = zeros(1,num_varT);                 % Jacobian matrix for Segment
N

generateFlow;                   % Generate flow rates for use in
temperature calculations and convert from m^3/d to m^3/s
generateFractions;              % Generate liquid holdups for use in
temperature calculations

converge = 0;                   % Convergence value to be compared
with the tolerance value
while(flagT)

    % Generating function matrices
    f1 =
f1T(Doil,Dgas,Tres,Tref,XT,q,Lfrac,Bo,Bg,Rs,Kappa_t,Kappa_a,f1,deltaP
_a,deltaP_t,KJT);
    f2 =
f2T(Doil,Dgas,Tres,Tref,XT,q,Lfrac,Bo,Bg,Rs,N,Kappa_t,Kappa_a,f2,bind
ex,deltaP_a,deltaP_t,KJT);
    f3 =
f3T(Doil,Dgas,XT,q,Lfrac,Bo,Bg,Rs,N,f3,num_varT,Nodes,bridges,Kappa_t
,Kappa_a,bindex,deltaP_a,deltaP_t,KJT);

    f = [f1 f2 f3];         % Combine the matrices

    % Generating jacobian matrices
    j1 = j1T(Doil,Dgas,XT,q,Lfrac,Bo,Bg,Rs,Kappa_t,Kappa_a,j1);
    j2 =
j2T(Doil,Dgas,XT,q,Lfrac,Bo,Bg,Rs,N,Kappa_t,Kappa_a,j2,bindex);
    j3 =
j3T(Doil,Dgas,XT,q,Lfrac,Bo,Bg,Rs,N,j3,num_varT,Nodes,bridges,Kappa_t
,Kappa_a,bindex);

    jac = [j1;j2;j3];       % Combine the matrices

    % LU factorization -- Inversion of the jacobian matrix

    [L1 U1] = lu(jac);
```

```
    L1_INV = inv(L1);
    U1_INV = inv(U1);

    temp1 = L1_INV*transpose(f);
    temp2 = U1_INV*temp1;

    temp3 = transpose(temp2);

    XT1_temp = XT;
    XT2 = XT - temp3;

    % Checking for convergence.  If checkConvergence finds that the
method
    % converges it will set flag to false and the program will stop.

    [flagT, test] = checkConvT(XT,XT2,num_varT,threshold);
    converge(sentinelCount+1) = test                    % Convergence
value of each iteration

    %Setting Xn = Xn+1 for the next iteration

    XT = XT2;

    % This if statement makes sure that the iteration stops if
    % the method does not converge whithin a number of iteration
input by the
    % user in input_data.m

    if(sentinelCount == stop)

        disp(' ');
        disp('Did not converge whithin the limitations given!');
        break;

    end

    % Update iteration index (sentinelCount)
    sentinelCount = sentinelCount + 1;

end

% Categorize converged temperature variables and
% result display
displayoutputT;
```

j1Generator.m

```
% Generate jacobian matrix for Segment 1

%Input:
%
%I          :   Pre-calculated coefficient for inflow equations
%X1         :   Unknown parameters at each iteration
%beta       :   Pre-calculated coefficient for tubing flow
calculations
%alpha      :   Pre-calculated coefficient for annular
flowcalculations
%B          :   Pre-calculated coefficient for slot/valve flow
calculations
%j1         :   Generated zero jacobian matrix
%Bo,Bg,Rs   :   Black-oil properties
%mu2P       :   Two-phase viscosities
%rho2P      :   Two-phase densities
%alpha_res  :   Liquid holdups in reservoir
%pref       :   Reference pressure
%qref       :   Reference flow rate
%
%Return:
%Jacobian matrix for Segment 1

function func =
j1Generator(I,X1,beta,alpha,B,j1,Bo,Bg,Rs,rho2P,mu2P,alpha_res,pref,q
ref)

j1(1,3) = -X1(7)/Bo(1);
j1(1,4) = X1(8)/Bo(2);
j1(1,7) = -X1(3)/Bo(1);
j1(1,8) = X1(4)/Bo(2);

j1(2,4) = -X1(8)/Bo(2);
j1(2,5) = -X1(9)/Bo(2);
j1(2,6) = alpha_res(1)/Bo(3);
j1(2,8) = -X1(4)/Bo(2);
j1(2,9) = -X1(5)/Bo(2);

j1(3,2) = I(1)*(pref/qref);
j1(3,6) = 1;

j1(4,1) = 1;
j1(4,3) = -1.75*beta(1)*(X1(3)^0.75)*rho2P(1)^0.75*mu2P(1)^0.25;
j1(4,10) = -1;

j1(5,1) = -1;
j1(5,2) = 1;
j1(5,4) = -2*B(1)*X1(4)*rho2P(2);

j1(6,2) = 1;
j1(6,5) = -1.75*alpha(1)*(X1(5)^0.75)*rho2P(3)^0.75*mu2P(3)^0.25;
j1(6,11) = -1;
```

```
j1(7,3) = -((1-X1(7))/Bg(1) + X1(7)*Rs(1)/Bo(1));
j1(7,4) = ((1-X1(8))/Bg(2) + X1(8)*Rs(2)/Bo(2));
j1(7,7) = -(-X1(3)/Bg(1) + Rs(1)*X1(3)/Bo(1));
j1(7,8) = (-X1(4)/Bg(2) + Rs(2)*X1(4)/Bo(2));

j1(8,4) = -((1-X1(8))/Bg(2) + X1(8)*Rs(2)/Bo(2));
j1(8,5) = -((1-X1(9))/Bg(2) + X1(9)*Rs(2)/Bo(2));
j1(8,6) = ((1-alpha_res(1))/Bg(3) + alpha_res(1)*Rs(3)/Bo(3));
j1(8,8) = -(-X1(4)/Bg(2) + Rs(2)*X1(4)/Bo(2));
j1(8,9) = -(-X1(5)/Bg(2) + Rs(2)*X1(5)/Bo(2));

j1(9,8) = -1;
j1(9,9) = 1;

func = j1;
```

j1T.m

```
% Generate jacobian matrix for Segment 1

%Input:
%
%Doil, Dgas :   Precalculated coefficients
%XT         :   Unknown temperatures at each iteration
%q          :   Flow rates
%Lfrac      :   Liquid holdups
%Bo,Bg,Rs   :   Black-oil properties
%Kappa_t    :   Overall heat transfer coefficients for fluid in
tubing
%Kappa_a    :   Overall heat transfer coefficients for fluid in
annulus
%j1         :   Generated zero jacobian matrix
%
%Return:
%Jacobian matrix for Segment 1

function func = j1T(Doil,Dgas,XT,q,Lfrac,Bo,Bg,Rs,Kappa_t,Kappa_a,j1)

j1(1,1) = 1;
j1(2,1) = -(Doil*q(2)*Lfrac(2)/Bo(2)...
                + Dgas*(q(2)*(1-Lfrac(2))/Bg(2)...
                + q(2)*Lfrac(2)*Rs(2)/Bo(2)))*(+1)...
            - (Doil*q(1)*Lfrac(1)/Bo(1)...
                + Dgas*(q(1)*(1-Lfrac(1))/Bg(1)...
                + q(1)*Lfrac(1)*Rs(1)/Bo(1)))*(-1)...
            - Kappa_t(1)*(+1);

j1(2,3) = - (Doil*q(1)*Lfrac(1)/Bo(1)...
                + Dgas*(q(1)*(1-Lfrac(1))/Bg(1)...
                + q(1)*Lfrac(1)*Rs(1)/Bo(1)))*(+1);

j1(3,1) = - 0*(Doil*q(2)*Lfrac(2)/Bo(2)...
                + Dgas*(q(2)*(1-Lfrac(2))/Bg(2)...
                + q(2)*Lfrac(2)*Rs(2)/Bo(2)))*(+1)...
            + 0*Kappa_t(1)*(+1);
j1(3,2) = - (Doil*q(3)*Lfrac(3)/Bo(2)...
                + Dgas*(q(3)*(1-Lfrac(3))/Bg(2)...
                + q(3)*Lfrac(3)*Rs(2)/Bo(2)))*(+1);

func = j1;
```

j2Generator.m

```
% Generate jacobian matrix for Segment 2 to N-1

%Input:
%
%X1          :   Unknown parameters at each iteration
%beta        :   Pre-calculated coefficient for tubing flow
calculations
%alpha       :   Pre-calculated coefficient for annular
flowcalculations
%I           :   Pre-calculated coefficient for inflow equations
%B           :   Pre-calculated coefficient for slot/valve flow
calculations
%j2          :   Generated zero jacobian matrix
%Bo,Bg,Rs    :   Black-oil properties
%mu2P        :   Two-phase viscosities
%rho2P       :   Two-phase densities
%alpha_res   :   Liquid holdups in reservoir
%pref        :   Reference pressure
%qref        :   Reference flow rate
%N           :   Number of segments
%b           :   Bridge indexes
%
%Return:
%Jacobian matrix for Segment 2 to N-1

function func =
j2Generator(X1,beta,alpha,I,B,j2,Bo,Bg,Rs,rho2P,mu2P,alpha_res,pref,q
ref,N,b)

for i=0:N-3

    var = 9*i;

    j2(1+var,3+var) = X1(7+var)/Bo(3*i+1)*b(4*i+1);
    j2(1+var,7+var) = X1(3+var)/Bo(3*i+1)*b(4*i+1);
    j2(1+var,12+var) = -X1(16+var)/Bo(3*i+4)*b(4*i+5);
    j2(1+var,13+var) = X1(17+var)/Bo(3*i+5)*b(4*i+6);
    j2(1+var,16+var) = -X1(12+var)/Bo(3*i+4)*b(4*i+5);
    j2(1+var,17+var) = X1(13+var)/Bo(3*i+5)*b(4*i+6);

    j2(2+var,5+var) = X1(9+var)/Bo(3*i+2)*b(4*i+3);
    j2(2+var,9+var) = X1(5+var)/Bo(3*i+2)*b(4*i+3);
    j2(2+var,13+var) = -X1(17+var)/Bo(3*i+5)*b(4*i+6);
    j2(2+var,14+var) = -X1(18+var)/Bo(3*i+5)*b(4*i+7);
    j2(2+var,15+var) = alpha_res(i+2)/Bo(3*i+6)*b(4*i+8);
    j2(2+var,17+var) = -X1(13+var)/Bo(3*i+5)*b(4*i+6);
    j2(2+var,18+var) = -X1(14+var)/Bo(3*i+5)*b(4*i+7);

    j2(7+var,3+var) = ((1-X1(7+var))/Bg(3*i+1) +
X1(7+var)*Rs(3*i+1)/Bo(3*i+1))*b(4*i+1);
    j2(7+var,7+var) = (-X1(3+var)/Bg(3*i+1) +
Rs(3*i+1)*X1(3+var)/Bo(3*i+1))*b(4*i+1);
```

```
    j2(7+var,12+var) = -((1-X1(16+var))/Bg(3*i+4) +
X1(16+var)*Rs(3*i+4)/Bo(3*i+4))*b(4*i+5);
    j2(7+var,13+var) = ((1-X1(17+var))/Bg(3*i+5) +
X1(17+var)*Rs(3*i+5)/Bo(3*i+5))*b(4*i+6);
    j2(7+var,16+var) = -(-X1(12+var)/Bg(3*i+4) +
Rs(3*i+4)*X1(12+var)/Bo(3*i+4))*b(4*i+5);
    j2(7+var,17+var) = (-X1(13+var)/Bg(3*i+5) +
Rs(3*i+5)*X1(13+var)/Bo(3*i+5))*b(4*i+6);

    j2(8+var,5+var) = ((1-X1(9+var))/Bg(3*i+2) +
X1(9+var)*Rs(3*i+2)/Bo(3*i+2))*b(4*i+3);
    j2(8+var,9+var) = (-X1(5+var)/Bg(3*i+2) +
Rs(3*i+2)*X1(5+var)/Bo(3*i+2))*b(4*i+3);
    j2(8+var,13+var) = -((1-X1(17+var))/Bg(3*i+5) +
X1(17+var)*Rs(3*i+5)/Bo(3*i+5))*b(4*i+6);
    j2(8+var,14+var) = -((1-X1(18+var))/Bg(3*i+5) +
X1(18+var)*Rs(3*i+5)/Bo(3*i+5))*b(4*i+7);
    j2(8+var,15+var) = ((1-alpha_res(i+2))/Bg(3*i+6) +
alpha_res(i+2)*Rs(3*i+6)/Bo(3*i+6))*b(4*i+8);
    j2(8+var,17+var) = -(-X1(13+var)/Bg(3*i+5) +
Rs(3*i+5)*X1(13+var)/Bo(3*i+5))*b(4*i+6);
    j2(8+var,18+var) = -(-X1(14+var)/Bg(3*i+5) +
Rs(3*i+5)*X1(14+var)/Bo(3*i+5))*b(4*i+7);

    if b(4*i+8) ~= 0     % There is inflow equation
        j2(3+var,11+var) = I(i+2)*pref/qref;
        j2(3+var,15+var) = 1;
    else                 % There is no inflow equation
        j2(3+var,15+var) = 1;
    end

    j2(4+var,10+var) = 1;
    j2(4+var,12+var) = -
1.75*beta(i+2)*(X1(12+var)^(0.75))*rho2P(4*i+5)^0.75*mu2P(4*i+5)^0.25
;
    j2(4+var,19+var) = -1;

    if b(4*i+6) ~= 0     % There is annular-to-tubing flow equation
        j2(5+var,11+var) = 1;
        j2(5+var,10+var) = -1;
        j2(5+var,13+var) = -2*B(i+2)*X1(13+var)*rho2P(4*i+6);
    else                 % There is no annular-to-tubing flow equation
        j2(5+var,13+var) = 1;
        j2(9+var,17+var) = 1;
        j2(9+var,18+var) = 1;
    end

    if b(4*i+7) ~= 0         % There is annular flow equation
        j2(6+var,11+var) = 1;
        j2(6+var,14+var) = -
alpha(i+2)*1.75*(X1(14+var)^0.75)*rho2P(4*i+7)^0.75*mu2P(4*i+7)^0.25;
        j2(6+var,20+var) = -1;
        if b(4*i+6) ~= 0     % There is tubing flow equation -- there
is split equation
            j2(9+var,17+var) = 1;
            j2(9+var,18+var) = -1;
```

```
        end
        if b(4*i+7) == -1    % If flow in annulus is toward toe of
well
            j2(6+var,11+var) = -1;
            j2(6+var,20+var) = 1;
        end
    elseif b(4*i+7) == 0     % There is no annular flow equation
        j2(6+var,14+var) = 1;            % The value "1" does not
affect the results
        j2(9+var,17+var) = 1;
        j2(9+var,18+var) = 1;
        if b(4*i+6) == 0
            j2(6+var,17+var) = 1e-20;    % To avoid singularity
            j2(2+var,14+var) = 1e-20;
            j2(2+var,17+var) = 1e-20;
            j2(8+var,14+var) = 1e-20;
            j2(8+var,17+var) = 1e-20;
        end
    end

end

func = j2;
```

j2T.m

```
% Generate jacobian matrix for Segment 2 to N-1

%Input:
%
%Doil, Dgas :   Precalculated coefficients
%XT         :   Unknown temperatures at each iteration
%q          :   Flow rates
%Lfrac      :   Liquid holdups
%Bo,Bg,Rs   :   Black-oil properties
%N          :   Number of segments
%Kappa_t    :   Overall heat transfer coefficients for fluid in
tubing
%Kappa_a    :   Overall heat transfer coefficients for fluid in
annulus
%j2         :   Generated zero jacobian matrix
%b          :   Bridge indexes
%
%Return:
%Jacobian matrix for Segment 2 to N-1

function func =
j2T(Doil,Dgas,XT,q,Lfrac,Bo,Bg,Rs,N,Kappa_t,Kappa_a,j2,b)

for i=0:N-3

    var = 4*i;

    j2(1+2*i,2*i+3) = -(Doil*q(var+6)*Lfrac(var+6)/Bo(3*i+5)...
                        + Dgas*(q(var+6)*(1-
Lfrac(var+6))/Bg(3*i+5)...
                        +
q(var+6)*Lfrac(var+6)*Rs(3*i+5)/Bo(3*i+5)))*(+1)*b(var+6)...
                    - (Doil*q(var+5)*Lfrac(var+5)/Bo(3*i+4)...
                        + Dgas*(q(var+5)*(1-
Lfrac(var+5))/Bg(3*i+4)...
                        +
q(var+5)*Lfrac(var+5)*Rs(3*i+4)/Bo(3*i+4)))*(-1)*b(var+5)...
                    - Kappa_t(i+2)*(+1);
    j2(1+2*i,2*i+2) = - (Doil*q(var+6)*Lfrac(var+6)/Bo(3*i+5)...
                        + Dgas*(q(var+6)*(1-
Lfrac(var+6))/Bg(3*i+5)...
                        +
q(var+6)*Lfrac(var+6)*Rs(3*i+5)/Bo(3*i+5)))*(-1)*b(var+6)...
                    - Kappa_t(i+2)*(-1);
    j2(1+2*i,2*i+5) = - (Doil*q(var+5)*Lfrac(var+5)/Bo(3*i+4)...
                        + Dgas*(q(var+5)*(1-
Lfrac(var+5))/Bg(3*i+4)...
                        +
q(var+5)*Lfrac(var+5)*Rs(3*i+4)/Bo(3*i+4)))*(+1)*b(var+5);

    j2(2+2*i,2*i+2) = -(Doil*q(var+8)*Lfrac(var+8)/Bo(3*i+6)...
                        + Dgas*(q(var+8)*(1-
Lfrac(var+8))/Bg(3*i+6)...
```

```
                                +
q(var+8)*Lfrac(var+8)*Rs(3*i+6)/Bo(3*i+6)))*(+1)*b(var+8) ...
                          - Kappa_a(i+2)*(+1)...
                          - (Doil*q(var+7)*Lfrac(var+7)/Bo(3*i+5)...
                              + Dgas*(q(var+7)*(1-
Lfrac(var+7))/Bg(3*i+5)...
                                +
q(var+7)*Lfrac(var+7)*Rs(3*i+5)/Bo(3*i+5)))*(-1)*b(var+7);
    j2(2+2*i,2*i+4) = - (Doil*q(var+7)*Lfrac(var+7)/Bo(3*i+5)...
                              + Dgas*(q(var+7)*(1-
Lfrac(var+7))/Bg(3*i+5)...
                                +
q(var+7)*Lfrac(var+7)*Rs(3*i+5)/Bo(3*i+5)))*(+1)*b(var+7);

    if b(4*i+7) == 0
            j2(2+2*i,2*i+4) = 1;
    end

end

func = j2;

% For case 3 (well with multiple inflow control valves), add
% c = b;                        % c is bridge indices
% Create all-positive bridge indeces so that the temperature change
due to
% pressure drop is dependent on the flow direction but only on the
pressure
% drop (either positive or negative change along the flow direction)
% for i = 1:(4*N-2)
%     if b(i)<0
%         b(i)=-b(i);
%     end
% end
% Flow directions can still be determined using "c".  At the annular
node
% where the reversed flow first starts, the temperature is fixed so
that it
% equals to the temperature of the reservoir inflow.
% if c(var+3)==-1
%   if c(var+7)==1
%       j2(2+2*i,2*i+2) = 1;
%       j2(2+2*(i-1),2*(i-1)+4) = 1;
%   end
% end
```

j3Generator.m

```
% Generate jacobian matrix for Segment N

%Input:
%
%X1          :   Unknown parameters at each iteration
%beta        :   Pre-calculated coefficient for tubing flow
calculations
%B           :   Pre-calculated coefficient for slot/valve flow
calculations
%j3          :   Generated zero jacobian matrix
%Bo,Bg,Rs    :   Black-oil properties
%mu2P        :   Two-phase viscosities
%rho2P       :   Two-phase densities
%pref        :   Reference pressure
%N           :   Number of segments
%num_var     :   Number of unknowns
%Nodes       :   Number of nodes
%bridges     :   Number of bridges
%b           :   Bridge indexes
%
%Return:
%Jacobian matrix for Segment N

function func =
j3Generator(X1,beta,B,j3,Bo,Bg,Rs,rho2P,mu2P,pref,N,num_var,Nodes,bri
dges,b)

j3(1,num_var-12) = X1(num_var-8)/Bo(Nodes-4)*b(bridges-5);
j3(1,num_var-8) = X1(num_var-12)/Bo(Nodes-4)*b(bridges-5);
j3(1,num_var-3) = -X1(num_var-1)/Bo(Nodes-1)*b(bridges-1);
j3(1,num_var-2) = X1(num_var)/Bo(Nodes)*b(bridges);
j3(1,num_var-1) = -X1(num_var-3)/Bo(Nodes-1)*b(bridges-1);
j3(1,num_var) = X1(num_var-2)/Bo(Nodes)*b(bridges);

j3(2,num_var-10) = X1(num_var-6)/Bo(Nodes-3)*b(bridges-3);
j3(2,num_var-6) = X1(num_var-10)/Bo(Nodes-3)*b(bridges-3);
j3(2,num_var-2) = -X1(num_var)/Bo(Nodes)*b(bridges);
j3(2,num_var) = -X1(num_var-2)/Bo(Nodes)*b(bridges);

j3(5,num_var-12) = ((1-X1(num_var-8))/Bg(Nodes-4) + X1(num_var-
8)*Rs(Nodes-4)/Bo(Nodes-4))*b(bridges-5);
j3(5,num_var-8) = (-X1(num_var-12)/Bg(Nodes-4) + Rs(Nodes-
4)*X1(num_var-12)/Bo(Nodes-4))*b(bridges-5);
j3(5,num_var-3) = -((1-X1(num_var-1))/Bg(Nodes-1) + X1(num_var-
1)*Rs(Nodes-1)/Bo(Nodes-1))*b(bridges-1);
j3(5,num_var-2) = ((1-X1(num_var))/Bg(Nodes) +
X1(num_var)*Rs(Nodes)/Bo(Nodes))*b(bridges);
j3(5,num_var-1) = -(-X1(num_var-3)/Bg(Nodes-1) + Rs(Nodes-
1)*X1(num_var-3)/Bo(Nodes-1))*b(bridges-1);
j3(5,num_var) = (-X1(num_var-2)/Bg(Nodes) + Rs(Nodes)*X1(num_var-
2)/Bo(Nodes))*b(bridges);
```

```
j3(6,num_var-10) = ((1-X1(num_var-6))/Bg(Nodes-3) + X1(num_var-
6)*Rs(Nodes-3)/Bo(Nodes-3))*b(bridges-3);
j3(6,num_var-6) = (-X1(num_var-10)/Bg(Nodes-3) + Rs(Nodes-
3)*X1(num_var-10)/Bo(Nodes-3))*b(bridges-3);
j3(6,num_var-2) = -((1-X1(num_var))/Bg(Nodes) +
X1(num_var)*Rs(Nodes)/Bo(Nodes))*b(bridges);
j3(6,num_var) = -(-X1(num_var-2)/Bg(Nodes) + Rs(Nodes)*X1(num_var-
2)/Bo(Nodes))*b(bridges);

j3(3,num_var-5) = 1;
j3(3,num_var-3) = -1.75*beta(N)*(X1(num_var-3)^0.75)*rho2P(bridges-
1)^0.75*mu2P(bridges-1)^0.25;

if b(bridges) ~= 0          % There is annular-to-tubing flow
equation
    j3(4,num_var-5) = -1;
    j3(4,num_var-4) = 1;
    j3(4,num_var-2) = -2*B(N)*X1(num_var-2)*rho2P(bridges);
else                        % There is no annular-to-tubing flow
equation
    j3(4,num_var-5) = 1;    % The value "1" does not affect the
results
    j3(4,num_var-4) = 1;
    j3(4,num_var-2) = 1;
end

func = j3;
```

j3T.m

```
% Generate jacobian matrix for Segment N

%Input:
%
%Doil, Dgas :   Precalculated coefficients
%XT         :   Unknown temperatures at each iteration
%q          :   Flow rates
%Lfrac      :   Liquid holdups
%Bo,Bg,Rs   :   Black-oil properties
%N          :   Number of segments
%j3         :   Generated zero jacobian matrix
%num_varT   :   Number of unknown temperatures
%Nodes      :   Number of nodes
%bridges    :   Number of bridges
%Kappa_t    :   Overall heat transfer coefficients for fluid in
tubing
%Kappa_a    :   Overall heat transfer coefficients for fluid in
annulus
%b          :   Bridge indexes
%
%Return:
%Jacobian matrix for Segment N

function func =
j3T(Doil,Dgas,XT,q,Lfrac,Bo,Bg,Rs,N,j3,num_varT,Nodes,bridges,Kappa_t
,Kappa_a,b)

j3(1,num_varT-1) =  -(Doil*q(bridges)*Lfrac(bridges)/Bo(Nodes)...
                        + Dgas*(q(bridges)*(1-
Lfrac(bridges))/Bg(Nodes)...
                        +
q(bridges)*Lfrac(bridges)*Rs(Nodes)/Bo(Nodes)))*(+1)*b(bridges)...
                    - (Doil*q(bridges-1)*Lfrac(bridges-1)/Bo(Nodes-
1)...
                        + Dgas*(q(bridges-1)*(1-Lfrac(bridges-
1))/Bg(Nodes-1)...
                        + q(bridges-1)*Lfrac(bridges-1)*Rs(Nodes-
1)/Bo(Nodes-1)))*(-1)*b(bridges-1)...
                    - Kappa_t(N)*(+1);
j3(1,num_varT-2) = -(Doil*q(bridges)*Lfrac(bridges)/Bo(Nodes)...
                    + Dgas*(q(bridges)*(1-
Lfrac(bridges))/Bg(Nodes)...
                    +
q(bridges)*Lfrac(bridges)*Rs(Nodes)/Bo(Nodes)))*(-1)*b(bridges)...
                - Kappa_t(N)*(-1);
j3(1,num_varT)   = - (Doil*q(bridges-1)*Lfrac(bridges-1)/Bo(Nodes-
1)...
               + Dgas*(q(bridges-1)*(1-Lfrac(bridges-1))/Bg(Nodes-
1)...
               + q(bridges-1)*Lfrac(bridges-1)*Rs(Nodes-1)/Bo(Nodes-
1)))*(+1)*b(bridges-1);

func = j3;
```

main.m

```
for zz = 1:5

networkSolver;
TempSolver;

flagmu = true;
muindex = 0;
convergemu = 0;
while (flagmu)

mu2P_temp =
updatemu(XT,X1,Tres,pres,pref,pb,N,num_var,num_varT,Nodes,bridges,Rs,
mu2P_res,mu_res);      % Recalculate pressure/temperature dependent
viscosities

[flagmu epsilonmu] = checkconvmu(mu2P,mu2P_temp,bridges,threshold);
% Check for convergence
convergemu(muindex+1) = epsilonmu

% Assign calculated temperatures for use in recalculating flow
parameters
% in isothermal calculations
mu2P=mu2P_temp;      % Update mu2P
T_temp(1)=Tres(1);   % Generate new T_temp for pressure calculations
    for i=2:num_varT
        T_temp(i)=XT(i-1);
    end
muindex = muindex+1

flag = true;
sentinelCount = 0;
iteration
TempSolver

end

% Categorize viscosity values
%displaymu;

% Get flow parameters at the location where asphaltene precipitation
is
% suspected to be used in the detailed analysis
location;

clc;
disp(num2str(p_tubing(100)));
disp(num2str(tubingFlowrates(100)));

zz
zzp(zz)= p_tubing(100)
zzQ(zz)= tubingFlowrates(100)
```

```
zzpbh(zz)= pbh
end

plot(zzp, zzQ)
ylabel('Flowrate (m^3/d)')
xlabel('Pressure (bara)')
```

mysubplots.m

```
subplot(2,2,[1 3])
plotflowrates;

subplot(2,2,2)
plotpressure;

subplot(2,2,4)
plotfractions;
```

networkSolver.m

```
%----------DATA LOADING BEGIN----------
input_data;                             % Input data file

pbh = pbh - zz * delta_pbh ;

input_p;                                % Input reservoir pressures
input_L;                                % Input segment lengths
input_c;                                % Input slot/valve discharge
coefficients
input_K;                                % Input absolute permeabilities
input_kro;                              % Calculate oil relative
permeabilities
input_krg;                              % Calculate gas relative
permeabilities
input_s;                                % Input skin factors

%----------GENERATING RESERVOIR FLUID PROPERTIES BEGIN--------
% Calculate fluid properties at inlet (reservoir) nodes.
[Bo_res Bg_res Rs_res] = generateResprop(N,pres,pb);            %
Calculate pressure-dependent black-oil properties at reservoir
conditions
mu_res = generatemures(Tres,pres,pb,N,Rs_res);                  %
Calculate pressure/temperature dependent oil viscosities
input_alpha;                                                    %
Calculate liquid holdups in the reservoir
[rho2P_res rho_res] = generateRhores(pres,pb,N,alpha_res);      %
Calculate pressure-dependent densities for each phase and two-phase
(TP) fluid
mu2P_res = generatemu2P(mu_res,N,alpha_res,pres,pb);            %
Calculate two-phase (TP) fluid viscosities

%----------GENERATING RESERVOIR FLUID PROPERTIES END--------

%----------DATA LOADING END----------

clc;

%----------PRE-CALCULATIONS BEGIN----------

precalculations;                        % Precalculate some coefficient
values to help incrase the calculation rate
guess;                                  % Initial guessed values of
unknown parameters

%----------GENERATING WELLBORE FLUID PROPERTIES BEGIN--------
% Calculate fluid properties at all nodes in the well network based
on the guessed unknown parameters.
Rs = generateRs(X1,pres,Rs_res,pref,pb,N,num_var,Nodes);        %
Calculate pressure-dependent gas solubilities
Bo = generateBo(X1,pres,Bo_res,pref,pb,N,num_var,Nodes);        %
Calculate pressure-dependent oil formation volume factors
Bg = generateBg(X1,pres,Bg_res,pref,pb,N,num_var,Nodes);        %
Calculate pressure-dependent gas formation volume factors
```

```
[mu2P mu] =
generatemu(T_temp,X1,Tres,pref,pb,N,num_var,num_varT,Nodes,bridges,Rs
,mu2P_res,mu_res);          % Calculate pressure/temperature dependent
viscosities for each phase and two-phase (TP) fluid
[rho2P rho] =
generaterho(X1,pres,rho2P_res,rho_res,pref,pb,N,num_var,Nodes,bridges
);                          % Calculate pressure-dependent densities for
each phase and two-phase (TP) fluid

%----------GENERATING WELLBORE FLUID PROPERTIES END--------

%----------PRE-CALCULATIONS END----------

%----------ITERATIVE PROCESS BEGIN----------

% SentinelCount counts how many iterations that has been done, so
that the
% program stops when the desired number of iterations are reached.
sentinelCount = 0;
% Flag determines whether the iteration should stop (solutions
converged)
flag = true;
generateBindex;             % Generate indexes for directions of flow
through each bridge
iteration;                  % Iterate function calculations to solve
for unknowns using Newton-Raphson method

%----------ITERATIVE PROCESS END----------

% Isothermal network model end
```

plotflowrates.m

```
plot(lengths,integralFlows,'-c', 'LineWidth',2);

hold on;

plot(lengths,tubingFlowrates,'-r');
plot(lengths(1:end-1),annularFlowrates(1:end),'-g');
plot(lengths(1:end),slotFlowRates(1:end),':b');

title('Flow rate profiles');
legend('Inflow rate','Tubing flow rate', 'Annular flow rate',
'Annular-to-tubing flow rate', 'Location', 'Best');
legend('boxoff');
xlabel('Distance from well toe (m)')
ylabel('Flow rate (m^3/d)')

hold off;
```

plotfractions.m

```
plot(lengths(1:end-1),alpha_res,'-k');

hold on;
plot(lengths,tubingFractions,'-m');
plot(lengths(1:end-1),annularFractions(1:end),':b','LineWidth',2);

title('Liquid volume fraction profiles');
legend('Fraction in reservoir','Fraction in tubing','Fraction in
annulus',  'Location', 'Best');
legend('boxoff');
xlabel('Distance from well toe (m)')
ylabel('Oil volume fraction')

hold off;
```

plotmu.m

```
plot(lengths(1:end-1), mu_reservoir(1:end)*1e3, '-g');

hold on;

plot(lengths, mu_annulus*1e3, ':m');
plot(lengths, mu_tubing*1e3, '-b', 'LineWidth',2);

title('Viscosity profiles');
legend('Viscosity in reservoir','Viscosity in annulus', 'Viscosity in
tubing','Location', 'Best');
legend('boxoff');
xlabel('Distance from well toe (m)')
ylabel('Viscosity (cp)')
%axis([0 xmax ymin ymax])

hold off;
```

plotpressure.m

```
xmax = lengths(N);
ymin = pbh/1e5;
ymax = pres(1)/1e5+1;

plot(lengths(1:N-1), pres/1e5, '-b', 'LineWidth',2);

hold on;

plot(lengths(1:end), p_annulus(1:end), ':m');
plot(lengths, p_tubing, '-g');

title('Pressure profiles');
legend('Reservoir pressure','Pressure in annulus', 'Pressure in
tubing','Location', 'Best');
legend('boxoff');
xlabel('Distance from well toe (m)')
ylabel('Presure (bara)')
axis([0 xmax ymin ymax])

hold off;
```

plotT.m

```
Tresplot = [Tres,Tres(end)];
T_tubing_plot = [T_tubing];
lengths_plot = [0,lengths];

plot(lengths_plot, Tresplot, '-b');

hold on;
plot(lengths_plot(1:end-1),T_annulus(1:end),'-gx');
plot(lengths_plot, T_tubing_plot, '-rx');

title('Temperature profiles');
legend('Reservoir temperature', 'Temperature in annulus',
'Temperature in tubing','Location', 'Best');
legend('boxoff');
xlabel('Distance from well toe (m)')
ylabel('Temperature (C)')

hold off;
```

plotting.m

```
% Plot results

disp('(1) Pressure profiles');
disp('(2) Flow rate profiles');
disp('(3) Liquid volume fraction profiles');
disp('(4) Temperature profiles');

plotFlag = 0;
moreplot = true;
figureFlag = 1;
plotFlag = input('Select a plot to be displayed ');

while (moreplot)
figure(figureFlag)

if(plotFlag == 1)              % Select 1 to plot pressure profile

    plotpressure;

elseif (plotFlag == 2)         % Select 2 to plot flow rate profile

    plotflowrates;

elseif (plotFlag == 3)         % Select 3 to plot liquid holdup profile

    plotfractions;

elseif (plotFlag == 4)         % Select 4 to plot liquid holdup profile

    plotT;

end

moreplot = input('you want to select other plots? if yes = type "1"
if no = type "0" ');
if moreplot == 0
    break;
end
figureFlag = figureFlag + 1;
disp('(1) Pressure profiles');
disp('(2) Flow rate profiles');
disp('(3) Volume fraction profiles');
disp('(4) Temperature profiles');
plotFlag = input('Select a plot to be displayed ');

end
```

precalculations.m

```
% Precalculate some coefficient values to help incrase the
calculation rate

% Calculate cross-sectional areas and hydraulic diameters

spacing_area = zeros(1,N-1);              % Annulus cross-sectional
area
hydraulic_diameter = zeros(1,N-1);        % Annulus hydraulic diameter
tubing_diameter = zeros(1,N);             % Tubing diameter
tubing_area = zeros(1,N);                 % Tubing cross-sectional area

for i=1:N-1
if r_temp(i)== ri(i)
    spacing_area(i) = pi*ro(i)^2 - pi*ri(i)^2;
% Annulus cross-sectional area
    hydraulic_diameter(i) = 4*((spacing_area(i))/(2*pi*ro(i) +
2*pi*ri(i)));  % Annulus hydraulic diameter
else
    spacing_area(i) = pi*ro(i)^2 - pi*r_temp(i)^2;
    hydraulic_diameter(i) = 4*((spacing_area(i))/(2*pi*ro(i) +
2*pi*r_temp(i)));
end
tubing_diameter(i) = 2*ri(i);                                        %
Tubing diameter
tubing_area(i) = pi*(ri(i)^2);                                       %
Tubing cross-sectional area
end
tubing_diameter(N) = 2*ri(N);
tubing_area(N) = pi*(ri(N)^2);

% Calculate coefficients for momentum balance equations

I = [];                                   % Coefficients for inflow
calculations
alpha = [];                               % Coefficients for annular
flow calculations
beta = [];                                % Coefficients for tubing
flow calculations
B = [];                                   % Coefficients for slot flow
calculations
for i=1:N

    if(i ~= N)
        if pres(i)/1e5 < pb
            I(i) = (2*pi*K(i)*L(i))/((log(re/ro(i)) +
s(i)))*(kro(i)/mu_res(1,i)+krg(i)/mu_res(2,i));
        else
            I(i) = (2*pi*K(i)*L(i))/((log(re/ro(i)) +
s(i)))*(kro(i)/mu_res(1,i));
        end
        if (i==1)
            qref = I(i)*(pref(i)-pbh)*(N-1);        % Generate
reference flow rate (inflow at Segment 1 x number of inlet bridges)
```

```
        end
        alpha(i) =
(0.3164*L(i)*(qref^1.75))/(2*(hydraulic_diameter(i)^1.25)*(spacing_ar
ea(i)^1.75)*pref);
    end

    beta(i) =
(0.3164*L(i)*(qref^1.75))/(2*(tubing_diameter(i)^1.25)*(tubing_area(i
)^1.75)*pref);
    B(i) = (c(i)*(qref^2))/(((slot_den*slot_L*slot_W*L(i))^2)*pref);

end
```

pressures.m

```
% Categorize pressures

%Input:
%
%X1          :   Unknown parameters (converged values)
%N           :   Number of segments
%num_var     :   Number of unknowns
%
%Return:
%p_tubing    :   Pressures in tubing
%p_annulus   :   Pressures in annulus

function [func1, func2] = pressures(X1,N,num_var)

% Segment 1
p_tubing(1) = X1(1);
p_annulus(1) = X1(2);

% Segment 2 to N-1
for i=1:N-2

    p_tubing(i+1) = X1(1+9*i);
    p_annulus(i+1) = X1(2+9*i);

end

% Segment N
p_tubing(N) = X1(num_var - 5);
p_annulus(N) = X1(num_var - 4);

func1 = p_tubing;
func2 = p_annulus;
```

Tconversion.m

```
% Scaling back the variables from relative to reference value

%Input:
%
%XT          :   Unknown temperatures (converged values)
%num_varT    :   Number of unknown temperatures
%Tref        :   Reference temperature
%
%Return:
%Converted temperature values

function func = Tconversion(XT,num_varT,Tref)

for i=1:num_varT

     XT(i) = XT(i)*Tref;

end

func = XT;
```

Temperatures.m

```
% Categorize temperatures

%Input:
%
%XT          :   Unknown temperatures (converged values)
%N           :   Number of segments
%num_varT    :   Number of unknown temperatures
%Tres        :   Reservoir temperatures
%
%Return:
%T_tubing    :   Temperatures in tubing
%T_annulus   :   Temperatures in annulus

function [func1, func2] = Temperatures(XT,N,num_varT,Tres)

% Temperatures in tubing
for i=0:N-1                             % Segment 1 to N

    T_tubing(i+1) = XT(1+2*i);

end
T_tubing(N+1) = XT(num_varT);           % Bottomhole temperature

% Temperatures in annulus
T_annulus(1) = Tres(1);                 % Segment 1
for i=0:N-2

    T_annulus(i+2) = XT(2+2*i);         % Segment 2 to N

end

func1 = T_tubing;
func2 = T_annulus;
```

TempSolver.m

```
%----------DATA LOADING BEGIN----------

input_dataT;                    % Input data file for temperature
calculations

generatepdrop;                  % Generate pressure drop between nodes

%----------DATA LOADING END----------

clc;

%----------PRE-CALCULATIONS BEGIN----------

XT_temp = zeros(1,num_varT);                    % Initial guessed
unknown temperatures
XT_temp = guessGeneratorT(Tres,L,Tref,Tbh,N);
XT = XT_temp;

%----------PRE-CALCULATIONS END----------

%----------ITERATIVE PROCESS BEGIN----------

% SentinelCount counts how many iterations that has been done, so
that the
% program stops when the desired number of iterations are reached.
sentinelCount = 0;
% Flag determines whether the iteration should stop (solutions
converged)
flagT = true;
iterationT                      % Iterate function calculations to solve
for temperature unknowns using Newton-Raphson method

%----------ITERATIVE PROCESS END----------

% Network model for temperature calculations end
```

updatemu.m

```
% Recalculate pressure/temperature dependent viscosities

%Input:
%
%XT          :   Unknown temperatures (converged)
%X1          :   Array containing pressure parameters
%Tres        :   Reservoir temperatures at reservoir nodes
%pref        :   Reference pressure
%pb          :   Bubblepoint pressure
%N           :   Number of segments
%num_var     :   Number of unknowns in isothermal calculations
%num_varT    :   Number of unknown temperatures
%Nodes       :   Number of nodes
%bridges     :   Number of bridges
%Rs          :   Gas solubilities
%mu2P_res    :   Two-phase viscosities at reservoir conditions
%mu_res      :   Viscosities of both phases at reservoir conditions
%
%Return:
%mu2P        :   Two-phase viscosities in every bridge in the network
%mu          :   Viscosities of both phases at every node in the
network

function func =
updatemu(XT,X1,Tres,pres,pref,pb,N,num_var,num_varT,Nodes,bridges,Rs,
mu2P_res,mu_res)

pb_temp=pb;                 % Bubblepoint pressure in bara
pb = pb*1e2;                % Bubblepoint pressure in kPa
XT_toe = Tres(1);           % Temperature at toe of well

mu = zeros(2,Nodes);            % Viscosities of both phases (1=liquid,
2=gas)
mu_od = zeros(1,Nodes);         % Dead-oil viscosities
mu_sat = zeros(1,Nodes);        % Saturated oil viscosities
gasmix;

for i=0:N-2


        var = i*9;

        X1(1+var) = X1(1+var)*pref/10^3;            % Change unit of
pressure from Pa to kPa
        X1(2+var) = X1(2+var)*pref/10^3;            % Change unit of
pressure from Pa to kPa

% Calculate viscosity for the gas phase (from curve-fit to values
from EOS)
        X_temp(1) = X1(1+var)/10^2;                 % Change unit of
pressure from kPa to bara
```

```
        X_temp(2) = X1(2+var)/10^2;                 % Change unit of
pressure from kPa to bara

    % Tubing node - calculating viscosity (gas phase)
    p = X_temp(1);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,3*i+1) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2)) +
Viscosity_Vap(i1);
    mu(1,3*i+1) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2)) +
Viscosity_Liq(i1);


    % Annular node - calculating viscosity (gas phase)
    p = X_temp(2);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,3*i+2) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2)) +
Viscosity_Vap(i1);
    mu(1,3*i+2) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2)) +
Viscosity_Liq(i1);


    % Reservoir node - calculating viscosity (gas phase)
    mu(2,3*i+3) = mu_res(2,i+1);
    mu(1,3*i+3) = mu_res(1,i+1);


end

    X1(num_var-5) = X1(num_var-5)*pref/10^3;        % Change unit of
pressure from Pa to kPa
    X1(num_var-4) = X1(num_var-4)*pref/10^3;        % Change unit of
pressure from Pa to kPa

% Calculate viscosity for the gas phase (from curve-fit to values
from EOS)
        X_temp(1) = X1(num_var-5)/10^2;             % Change unit of
pressure from kPa to bara
        X_temp(2) = X1(num_var-4)/10^2;             % Change unit of
pressure from kPa to bara


        % Tubing node - calculating viscosity (gas phase)
```

```
    p = X_temp(1);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,Nodes-1) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2))
+ Viscosity_Vap(i1);
    mu(1,Nodes-1) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2))
+ Viscosity_Liq(i1);

    % Annular node - calculating viscosity (gas phase)
    p = X_temp(2);
    [i1, p1, i2, p2] = interpolate(p);

    dp = p1-p2;
    if dp==0
        dp = 2;
    end

    mu(2,Nodes) = (p-p1)/dp * (Viscosity_Vap(i1)-Viscosity_Vap(i2)) +
Viscosity_Vap(i1);
    mu(1,Nodes) = (p-p1)/dp * (Viscosity_Liq(i1)-Viscosity_Liq(i2)) +
Viscosity_Liq(i1);


mu2P = zeros(1,bridges);                % Generate two-phase viscosities
in every bridge

for i=0:N-2

    var = i*9;

    mu2P(i*4+1) = mu(1,3*i+1)*X1(var+7) + mu(2,3*i+1)*(1-X1(var+7));
% Tubing bridge of Segment i+1
    mu2P(i*4+2) = mu(1,3*i+2)*X1(var+8) + mu(2,3*i+2)*(1-X1(var+8));
% Annulus-to-tubing bridge of Segment i+1
    mu2P(i*4+3) = mu(1,3*i+2)*X1(var+9) + mu(2,3*i+2)*(1-X1(var+9));
% Annular bridge of Segment i+1
    mu2P(i*4+4) = mu2P_res(i+1);
% Inlet bridge of Segment i+1

end

mu2P(bridges-1) = mu(1,Nodes-1)*X1(num_var-1) + mu(2,Nodes-1)*(1-
X1(num_var-1));              % Tubing bridge of Segment N
mu2P(bridges) = mu(1,Nodes)*X1(num_var) + mu(2,Nodes)*(1-
X1(num_var));                % Annulus-to-tubing bridge of
Segment N

func = mu2P*10^(-3);
```

Vertical Model Code

File name	Description
analytical_Tplot.m	Plots analytical temperature profile and plots it on the existing figure
calculate_heatProp.m	Interpolates and converts fluid properties that are used in Ramey's temperature balance only
calculate_prop.m	Interpolates and converts fluid properties that are used in the Hagedorn and Brown correlation only
calculateT.m	Calculates temperature profile using Ramey's model
checkT2match.m	Checks if temperature profile matches to decide for iteration
fluid.m	Contains tables of fluid properties
guessT2p2.m	Guesses the values of T1 and p2 for Hagedorn and Brown correlation
HB.m	Hagedorn and Brown correlation calculations
inputs.m	Inputs into the Hagedown and Brown correlation
interpolateT.m	Interpolates the value of temperature from the previous Ramey's calculation in order to do the next Hagedorn and Brown calculations
main.m	Runs all the file name in sequence and stores the calculated values for future use and plotting
mainQ.m	Executes main.m file for different values of flowrate to give a lift curve
mainSINGLE.m	Runs main.m file for a single run
mainTd.m	Executes main.m file for different values of Td to give temperature profiles as a function of time
plot_final_run_only.m	Plots the final results of the calculations
plot_porgression.m	Plots the progression of calculation
plotHB.m	Plots all the iterations of the vertical flow model

analytical_Tplot.m

```
mdot = q_m/24;                          % m in lbm/hr
cpoo = 1.42*10^(-5)*230*100;            % cp in Btu/lbm/F
DD = dt;                                % D in ft
Uoo = 5;                                % Uto in Btu/hr/ft2/F
Tin = 135;                              % Tin in F
Tamb = 212;                             % Tamb in F

for counter = 1:length(xh)
    Tana(counter,1) = Tamb + (Tin-Tamb) * exp(-
1*3.142*DD*Uoo*xh(counter,Titeration)/cpoo/mdot)
end

hold on
plot((Tana-32)*5/9,-0.3048*xh(1:length(xh),Titeration),'--k')
ylabel('height(m)')
xlabel('T(^oC)')
```

calculate_heatProp.m

```
function [Cp, Cj] = calculate_prop(T,p,HL)
%   calculate_heatProp - calculates heat properties from PVT data
%   Cp, Cj


% data from PVTsim
fluid;

for i = 1:11
    if Pressure(i,1) == p
        p1 = Pressure(i,1);
        p2 = Pressure(i,1);
        i1 = i;
        i2 = i;
        break
    else if Pressure(i,1)<p
        p1 = Pressure (i,1);
    else
        p2 = Pressure(i,1);
        i1 = i-1;
        i2 = i;
        break
        end
    end
end
    if i1 == 0
        i1 = 1;
        p1 = Pressure (i1,1);
    end


for j = 1:10
    if Temperature(1,j) == T
        T1 = Temperature(1,j);
        T2 = Temperature(1,j);
        j1 = j;
        j2 = j;
        break
    else if Temperature(1,j)<T
        T1 = Temperature(1,j);
    else
        T2 = Temperature(1,j);
        j1 = j-1;
        j2 = j;
        break
        end
    end
end

dT = T1-T2;
dp = p1-p2;

if dT == 0
```

```
    dT = 2;
end
if dp ==0
    dp = 2;
end


% calculating Cp (J/mol C)
x = Cp_Tot(i1, j1) + (Cp_Tot(i1, j1)-Cp_Tot(i1, j2))*(T-T1)/(dT);
y = Cp_Tot(i2, j1) + (Cp_Tot(i2, j1)-Cp_Tot(i2, j2))*(T-T1)/(dT);
Cp = x + (x-y)*(p-p1)/(dp);

%Cp = 0.0145/(Cp-32); %(J/mol C to Btu/lbm F)
Cp = 1.42*10^(-5)*Cp*100;


% calculating Cj (C/bara)
x = JT_V(i1, j1) + (JT_V(i1, j1)-JT_V(i1, j2))*(T-T1)/(dT);
y = JT_V(i2, j1) + (JT_V(i2, j1)-JT_V(i2, j2))*(T-T1)/(dT);
JT_V = x + (x-y)*(p-p1)/(dp);

x = JT_L(i1, j1) + (JT_L(i1, j1)-JT_L(i1, j2))*(T-T1)/(dT);
y = JT_L(i2, j1) + (JT_L(i2, j1)-JT_L(i2, j2))*(T-T1)/(dT);
JT_L = x + (x-y)*(p-p1)/(dp);

Cj = JT_V*(1-HL) + JT_L*(HL);

Cj = 0.124 * Cj;

end
```

calculate_prop.m

```
function [SGo, SGg, Rs, Bo, Bg, miu_o, miu_g, sigma_o, z, SGw, Bw,
miu_w, sigma_w] = calculate_prop(T,p)
%   calculate_p - calculates the various properties from PVT data
%   SGo, SGg, Rs, Bo, Bg, miu_o, miu_g, sigma_o, z, SGw, Bw, miu_w,
sigma_w


% data from PVTsim
fluid;

for i = 1:11
    if Pressure(i,1) == p
        p1 = Pressure(i,1);
        p2 = Pressure(i,1);
        i1 = i;
        i2 = i;
        break
    else if Pressure(i,1)<p
        p1 = Pressure (i,1);
    else
        p2 = Pressure(i,1);
        i1 = i-1;
        i2 = i;
        break
        end
    end
end
    if i1 == 0
        i1 = 1;
        p1 = Pressure (i1,1);
    end


for j = 1:10
    if Temperature(1,j) == T
        T1 = Temperature(1,j);
        T2 = Temperature(1,j);
        j1 = j;
        j2 = j;
        break
    else if Temperature(1,j)<T
        T1 = Temperature(1,j);
    else
        T2 = Temperature(1,j);
        j1 = j-1;
        j2 = j;
        break
        end
    end
end

dT = T1-T2;
dp = p1-p2;
```

```
if dT == 0
    dT = 2;
end
if dp ==0
    dp = 2;
end


% calculating SGo (unitless)
x = Density_Liq(i1, j1) + (Density_Liq(i1, j1)-Density_Liq(i1,
j2))*(T-T1)/(dT);
y = Density_Liq(i2, j1) + (Density_Liq(i2, j1)-Density_Liq(i2,
j2))*(T-T1)/(dT);
oil_density = x + (x-y)*(p-p1)/(dp);

SGo = oil_density/1;


% calculating SGg (unitless)
x = Density_Vap(i1, j1) + (Density_Vap(i1, j1)-Density_Vap(i1,
j2))*(T-T1)/(dT);
y = Density_Vap(i2, j1) + (Density_Vap(i2, j1)-Density_Vap(i2,
j2))*(T-T1)/(dT);
gas_density = x + (x-y)*(p-p1)/(dp);

SGg = gas_density/1;


% calculating Rs, Bo, Bg (unitless)
x = xBo(i1, j1) + (xBo(i1, j1)-xBo(i1, j2))*(T-T1)/(dT);
y = xBo(i2, j1) + (xBo(i2, j1)-xBo(i2, j2))*(T-T1)/(dT);
Bo = x + (x-y)*(p-p1)/(dp);

x = xBg(i1, j1) + (xBg(i1, j1)-xBg(i1, j2))*(T-T1)/(dT);
y = xBg(i2, j1) + (xBg(i2, j1)-xBg(i2, j2))*(T-T1)/(dT);
Bg = x + (x-y)*(p-p1)/(dp);

x = xRs(i1, j1) + (xRs(i1, j1)-xRs(i1, j2))*(T-T1)/(dT);
y = xRs(i2, j1) + (xRs(i2, j1)-xRs(i2, j2))*(T-T1)/(dT);
Rs = x + (x-y)*(p-p1)/(dp);



% calculating miu_o (cp)
x = Viscosity_Liq(i1, j1) + (Viscosity_Liq(i1, j1)-Viscosity_Liq(i1,
j2))*(T-T1)/(dT);
y = Viscosity_Liq(i2, j1) + (Viscosity_Liq(i2, j1)-Viscosity_Liq(i2,
j2))*(T-T1)/(dT);
oil_density = x + (x-y)*(p-p1)/(dp);

miu_o = oil_density;
```

```
% calculating miu_g (cp)
x = Viscosity_Vap(i1, j1) + (Viscosity_Vap(i1, j1)-Viscosity_Vap(i1,
j2))*(T-T1)/(dT);
y = Viscosity_Vap(i2, j1) + (Viscosity_Vap(i2, j1)-Viscosity_Vap(i2,
j2))*(T-T1)/(dT);
gas_density = x + (x-y)*(p-p1)/(dp);

miu_g = gas_density;


% calculating sigma_o (1 mN/m = 1 dyne/cm)
x = Surf_Ten(i1, j1) + (Surf_Ten(i1, j1)-Surf_Ten(i1, j2))*(T-
T1)/(dT);
y = Surf_Ten(i2, j1) + (Surf_Ten(i2, j1)-Surf_Ten(i2, j2))*(T-
T1)/(dT);
oil_st = x + (x-y)*(p-p1)/(dp);

sigma_o = oil_st;


% calculating z
x = Z_Factor_Vap(i1, j1) + (Z_Factor_Vap(i1, j1)-Z_Factor_Vap(i1,
j2))*(T-T1)/(dT);
y = Z_Factor_Vap(i2, j1) + (Z_Factor_Vap(i2, j1)-Z_Factor_Vap(i2,
j2))*(T-T1)/(dT);
gas_z = x + (x-y)*(p-p1)/(dp);

z = gas_z;


SGw = 0;
Bw = 0;                          % bbl/stb
miu_w = 1;                       % cp
sigma_w = 0;                     % dyn/cm

end
```

calculateT.m

```
for counter=length(xT2):-1:1

    if counter==length(xT2)

        T_hi(counter, Titeration-1) = T_BH;

    else
        T_ei = xh(counter, Titeration-1)*(T_BH-T_TH)/h_well + T_TH;
        g = 32.17;                                                    %
(ft/s2)
        gc = 32.17;                                                   %
(lbm.ft/lbf/s2)
        Jc = 778.16;                                                  %
(lbf.ft/Btu)
        U = 17.61;                                                    %
(Btu/hr/ft2/degreeF)
        ke = 1.4;                                                     %
(Btu/hr/ft/degreeF)
        T = (T_hi(counter+1,Titeration-1)-32) * 5/9;                  %
convert to degree C to read off the grid
        p = 0.0689475729 * xp2(counter,Titeration-1);                 %
convert to bara to read off the grid
        [Cp, Cj] = calculate_heatProp(T,p,HL);
        Lr = 2*3.142*r_to*U*ke/((Cp*q_m/24)*(ke+r_to*U*Td));          %
(1/ft)

        T_hi(counter, Titeration-1) =((Cp*T_hi(counter+1,Titeration-
1))+(Lr*Cp*xdelta_h(counter,Titeration-1)*T_ei)-
(Cj*Cp*xdelta_p(counter,Titeration-1))+(xdelta_h(counter,Titeration-
1)*g/Jc/gc)+(xdelta_v_m_sq(counter,Titeration-
1)/2/Jc/gc))/((Lr*Cp*xdelta_h(counter,Titeration-1))+Cp);


    end

end
```

checkT2match.m

```
if Titeration == 1
    T_h = h*(T_BH-T_TH)/h_well + T_TH;
else
    if k == kmax
        T_h = T_BH;
    else
        interpolateT;
    end
end


if abs(T2-T_h) < 0.1
    T2check=0;
else
    T2 = T_h;
    T2check=1;
    h = h - delta_h;
end
```

fluid.m

```
% fluid
% PVTsim TEST4 BHS OIL C10+


Pressure= [
1.01    1.01    1.01    1.01    1.01    1.01    1.01    1.01    1.01
1.01
50.91   50.91   50.91   50.91   50.91   50.91   50.91   50.91   50.91
50.91
100.81  100.81  100.81  100.81  100.81  100.81  100.81  100.81
100.81  100.81
150.71  150.71  150.71  150.71  150.71  150.71  150.71  150.71
150.71  150.71
200.61  200.61  200.61  200.61  200.61  200.61  200.61  200.61
200.61  200.61
250.51  250.51  250.51  250.51  250.51  250.51  250.51  250.51
250.51  250.51
300.41  300.41  300.41  300.41  300.41  300.41  300.41  300.41
300.41  300.41
350.31  350.31  350.31  350.31  350.31  350.31  350.31  350.31
350.31  350.31
400.21  400.21  400.21  400.21  400.21  400.21  400.21  400.21
400.21  400.21
450.11  450.11  450.11  450.11  450.11  450.11  450.11  450.11
450.11  450.11
500.01  500.01  500.01  500.01  500.01  500.01  500.01  500.01
500.01  500.01
];


Temperature=[
0   30  60  90  120 150 180 210 240 270
0   30  60  90  120 150 180 210 240 270
0   30  60  90  120 150 180 210 240 270
0   30  60  90  120 150 180 210 240 270
0   30  60  90  120 150 180 210 240 270
0   30  60  90  120 150 180 210 240 270
0   30  60  90  120 150 180 210 240 270
0   30  60  90  120 150 180 210 240 270
0   30  60  90  120 150 180 210 240 270
0   30  60  90  120 150 180 210 240 270
];


Density_Vap= [
0.001   0.001   0.001   0.001   0.0011  0.0012  0.0013  0.0014
0.0015  0.0016
0.0485  0.0436  0.0403  0.0381  0.0367  0.0363  0.0367  0.0379
0.0401  0.0431
0.1084  0.0919  0.0822  0.0759  0.072   0.0699  0.0696  0.0708
0.0737  0.0784
```

```
0.1745  0.1438  0.126   0.1149  0.108   0.1042  0.103   0.1043  0.108
0.1144
0   0.1932  0.1693  0.154   0.1444  0.1391  0.1373  0.1389  0.1439
0.1528
0   0   0   0.1922  0.1808  0.1746  0.1728  0.1753  0.1825  0.1952
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
];

Density_Liq= [
0.8742  0.8667  0.8591  0.8521  0.846   0.8412  0.8375  0.8352
0.8353  0.8381
0.8333  0.8276  0.8192  0.8087  0.7964  0.7826  0.7677  0.752
0.7358  0.7196
0.8072  0.8041  0.7968  0.7866  0.7739  0.7592  0.7426  0.7244
0.7049  0.6842
0.7865  0.7837  0.7769  0.7668  0.7538  0.7383  0.7206  0.7007
0.6788  0.6549
0.7816  0.7656  0.7584  0.748   0.7345  0.7182  0.6993  0.6778
0.6536  0.6267
0.7855  0.767   0.7468  0.7297  0.7155  0.6982  0.6779  0.6545
0.6277  0.5971
0.7892  0.7716  0.7526  0.7323  0.7105  0.6873  0.663   0.6375
0.6113  0.5845
0.7925  0.7759  0.758   0.7388  0.7185  0.6971  0.6746  0.6513
0.6274  0.6031
0.7956  0.7798  0.7628  0.7448  0.7257  0.7057  0.6849  0.6633
0.6413  0.6189
0.7985  0.7834  0.7673  0.7502  0.7322  0.7135  0.694   0.6739
0.6534  0.6326
0.8012  0.7868  0.7714  0.7552  0.7382  0.7205  0.7021  0.6833
0.6641  0.6447
];


Z_Factor_Vap = [
0.6029  0.6233  0.6483  0.6802  0.7162  0.7553  0.7942  0.8318
0.8685  0.9026
0.5879  0.6219  0.6457  0.6637  0.6786  0.6924  0.7066  0.7222
0.7396  0.7591
0.7278  0.7474  0.7616  0.7714  0.7786  0.7846  0.7905  0.7971
0.8052  0.815
0.9434  0.925   0.9162  0.9102  0.905   0.9007  0.8974  0.8955
0.8955  0.8975
1.2201  1.1361  1.0975  1.0702  1.0491  1.032   1.0182  1.0072
0.9989  0.9934
1.5159  1.399   1.3074  1.2451  1.206   1.1747  1.1492  1.1283
1.1115  1.0985
1.8095  1.6676  1.5556  1.4669  1.3965  1.3412  1.2984  1.2664
1.2435  1.2286
2.1012  1.9339  1.8013  1.6953  1.6102  1.5421  1.4879  1.4454
1.4128  1.3886
```

```
2.3911  2.1983  2.0448  1.9213  1.8213  1.7401  1.6744  1.6214
1.5791  1.5458
2.6795  2.461   2.2864  2.1452  2.0301  1.9359  1.8585  1.795
1.7431  1.7009
2.9665  2.7221  2.5263  2.3674  2.2371  2.1296  2.0405  1.9665
1.9051  1.854
];


Viscosity_Vap= [
0.0103  0.0112  0.012   0.0128  0.0135  0.0141  0.0147  0.0154
0.0161  0.0169
0.0121  0.0129  0.0137  0.0145  0.0153  0.0161  0.0169  0.0178
0.0186  0.0194
0.0149  0.015   0.0155  0.0161  0.0168  0.0176  0.0184  0.0192
0.0202  0.0212
0.0202  0.0184  0.0181  0.0183  0.0188  0.0194  0.0202  0.0211
0.0221  0.0233
0   0.0229  0.0215  0.0211  0.0212  0.0217  0.0224  0.0234  0.0246
0.0262
0   0   0   0.0244  0.0242  0.0245  0.0252  0.0263  0.0279  0.0302
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
];


Viscosity_Liq= [
44.719  18.6556 9.1586  5.1278  3.0867  2.0074  1.5025  1.3105
1.2559  1.2594
10.1093 5.036   2.4153  1.3842  1.0434  0.8616  0.7361  0.646
0.5715  0.5062
3.9702  2.0006  1.2934  1.024   0.8605  0.7377  0.6314  0.5304
0.4559  0.4
1.6215  1.1827  1.0089  0.8664  0.7467  0.6232  0.5175  0.4405
0.3797  0.3303
1.4088  0.9895  0.8657  0.7484  0.6161  0.513   0.4351  0.3715
0.3186  0.2742
1.514   1.0215  0.7903  0.6094  0.5106  0.4333  0.3684  0.3136
0.2669  0.2263
1.6218  1.0893  0.8452  0.6167  0.484   0.3938  0.327   0.2761
0.2365  0.2049
1.7321  1.157   0.9004  0.6525  0.5116  0.4173  0.3475  0.2945
0.2532  0.2204
1.8447  1.2244  0.9556  0.6884  0.5386  0.4402  0.3675  0.3122
0.2692  0.2352
1.9597  1.2915  1.0107  0.7243  0.565   0.4626  0.3869  0.3294
0.2847  0.2493
2.077   1.3579  1.0655  0.7603  0.591   0.4846  0.406   0.3462
0.2998  0.2631
];

Surf_Ten= [
```

```
26.554  25.327  24.059  22.821  21.673  20.672  19.831  19.176
18.777  18.665
16.476  16.422  15.982  15.272  14.362  13.303  12.147  10.946  9.748
8.596
8.781   9.908   10.282  10.187  9.766   9.106   8.276   7.336   6.339
5.336
4.044   5.461   6.228   6.5 6.407   6.043   5.481   4.785   4.011
3.214
0   2.891   3.61    3.98    4.041   3.854   3.479   2.973   2.392
1.789
0   0   0   2.347   2.442   2.341   2.085   1.72    1.297   0.866
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
];

Cp_Tot = [
199.44  210.99  222 232.23  241.73  250.44  258.55  266.07  272.92
279.3
208.95  219.67  230.74  241.93  253.09  264.07  274.75  285.03  294.8
303.97
213.78  224.15  234.84  245.75  256.76  267.73  278.55  289.13
299.37  309.18
213.45  225.7   236.97  248.07  259.14  270.15  281.04  291.73
302.15  312.21
212.12  225.06  237.4   249.08  260.46  271.65  282.66  293.46
303.98  314.16
211.45  224.15  236.82  249.19  260.97  272.42  283.6   294.51
305.12  315.37
210.86  223.4   235.88  248.21  260.32  272.13  283.58  294.57
305.02  314.85
210.34  222.76  235.08  247.24  259.15  270.74  281.96  292.74  303
312.7
209.87  222.19  234.39  246.41  258.17  269.61  280.67  291.3
301.44  311.06
209.46  221.68  233.79  245.7   257.34  268.66  279.61  290.14  300.2
309.76
209.08  221.23  233.26  245.08  256.63  267.86  278.72  289.17
299.18  308.71
];

JT_V = [
0.7843  0.6745  0.6086  0.5741  0.5553  0.5447  0.5322  0.517
0.5003  0.4794
0.5683  0.4717  0.398   0.3415  0.2989  0.268   0.2466  0.2326
0.2242  0.2195
0.4039  0.3572  0.3097  0.2692  0.2364  0.2111  0.1921  0.1786
0.1693  0.1633
0.2294  0.2395  0.2245  0.2033  0.1828  0.1652  0.151   0.14
0.1319  0.126
0   0.148   0.1528  0.1466  0.1365  0.1259  0.1164  0.1083  0.1018
0.0964
0   0   0   0.1013  0.0982  0.0929  0.087   0.0814  0.0763  0.0714
```

```
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0
];

JT_L = [
-0.0545 -0.0504 -0.0467 -0.0432 -0.0401 -0.0373 -0.0347 -0.0324 -
0.0304 -0.0288
-0.0525 -0.0485 -0.0445 -0.0406 -0.0367 -0.0327 -0.0285 -0.0241 -
0.0195 -0.0145
-0.0514 -0.0474 -0.0434 -0.0395 -0.0355 -0.0314 -0.027  -0.0223 -
0.0172 -0.0115
-0.0506 -0.0466 -0.0426 -0.0387 -0.0347 -0.0305 -0.026  -0.0212 -
0.0159 -0.0099
-0.0508 -0.046  -0.042  -0.038  -0.034  -0.0298 -0.0253 -0.0204 -
0.0149 -0.0089
-0.0516 -0.0468 -0.042  -0.0375 -0.0334 -0.0291 -0.0246 -0.0197 -
0.0141 -0.0079
-0.0523 -0.0477 -0.0432 -0.0387 -0.0342 -0.0296 -0.0248 -0.0199 -
0.0149 -0.0098
-0.0528 -0.0484 -0.0442 -0.04   -0.0359 -0.0318 -0.0277 -0.0235 -
0.0194 -0.0153
-0.0533 -0.0491 -0.045  -0.0411 -0.0373 -0.0336 -0.0299 -0.0263 -
0.0227 -0.0193
-0.0538 -0.0496 -0.0457 -0.042  -0.0384 -0.035  -0.0317 -0.0284 -
0.0253 -0.0223
-0.0542 -0.0501 -0.0464 -0.0428 -0.0394 -0.0362 -0.0331 -0.0301 -
0.0273 -0.0246
];

xBo = [
0.994   1.007   1.022   1.038   1.054   1.07    1.086   1.1 1.112
2.981
1.103   1.112   1.132   1.166   1.216   1.29    1.393   1.532   1.719
5.245
1.172   1.173   1.19    1.223   1.277   1.357   1.471   1.629   1.843
5.695
1.234   1.233   1.249   1.283   1.34    1.427   1.553   1.728   1.972
6.165
1.254   1.293   1.309   1.346   1.408   1.504   1.644   1.842   2.122
6.732
1.248   1.297   1.356   1.416   1.485   1.592   1.75    1.979   2.311
7.486
1.242   1.289   1.345   1.419   1.519   1.656   1.846   2.11    2.481
8.041
1.237   1.282   1.336   1.406   1.502   1.632   1.814   2.065   2.417
7.792
1.232   1.275   1.327   1.395   1.487   1.612   1.787   2.028   2.365
7.593
1.228   1.269   1.32    1.385   1.473   1.595   1.763   1.996   2.321
7.427
1.223   1.264   1.313   1.376   1.461   1.579   1.743   1.968   2.283
7.288
];
```

```
xBg = [
0.016   1.057   1.164   1.274   1.388   1.509   1.642   1.790   1.961
2.171
0.007   0.019   0.022   0.024   0.026   0.028   0.031   0.033   0.035
0.037
0.005   0.009   0.010   0.012   0.013   0.014   0.015   0.016   0.017
0.018
0.000   0.006   0.007   0.008   0.009   0.009   0.010   0.011   0.012
0.012
0.000   0.004   0.005   0.006   0.006   0.007   0.008   0.008   0.009
0.010
0.000   0.000   0.000   0.005   0.005   0.006   0.006   0.007   0.007
0.008
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
0.000
];

xRs = [
0.2 0   0   0   0   0   0   0   1.7
47.1    39.6    36.3    35.7    37.9    42.4    48.9    56.9    65.7
202.4
81.3    68  61.2    58.8    60.2    64.9    72.5    82.6    94.9
294.1
112.9   96.4    87.1    83.2    84.1    89.2    98.3    111.1   127.7
398.9
125.5   125.4   114.6   109.8   110.6   116.6   127.9   144.4   166.9
528.6
125.5   130.8   137 139.3   140.6   148.3   162.7   184.6   216 699.5
125.5   130.8   137 145.4   157.4   173.8   196 225.2   263.8   845.5
125.5   130.8   137 145.4   157.4   173.8   196 225.2   263.8   845.5
125.5   130.8   137 145.4   157.4   173.8   196 225.2   263.8   845.5
125.5   130.8   137 145.4   157.4   173.8   196 225.2   263.8   845.5
125.5   130.8   137 145.4   157.4   173.8   196 225.2   263.8   845.5
];
```

guessT2p2.m

```
% Guesses P2 and T2

p2=1.01*p1;                          % psig
T2=1.01*T1;                          % degree F
```

HB.m

```
%-------------------------------------%
%   Hagedorn and Brown method (H&B)   %
%-------------------------------------%


% p_bar (psia) and T_bar (degree F)
p_bar = (p1+p2)/2 + 14.7;          % psia
T_bar = (T1+T2)/2;                 % degree F

T = (T_bar-32) * 5/9;              % convert to degree C to read off the
grid
p = 0.0689475729 * p_bar;          % convert to bara to read off the
grid
[SGo, SGg, Rs, Bo, Bg, miu_o, miu_g, sigma_o, z, SGw, Bw, miu_w,
sigma_w] = calculate_prop(T,p);

% mass associated with 1barrel of ST liquid - m_t (lbm)
m_t = 350*SGo*(1/(1+WOR)) + 350*SGw*(WOR/(1+WOR)) + (0.0764*GLR*SGg);

% flowrate at average segment point - q_m (lbm/day)
q_m = m_t * q_l;

% density of liquid phase - roh_l (lb/ft3)
roh_l =
((62.4*SGo+Rs*SGg*0.0764/5.615)/((1+WOR)*Bo))+(62.4*SGw*WOR/(1+WOR));

% density of gas phase - roh_g (lb/ft3)
roh_g = 0.07645*SGg*(p_bar*520/(14.7*z*(T_bar+460)));

% viscosity of liquid mixure - miu_l (cp)
miu_l = (miu_o*1/(1+WOR)) + (miu_w*WOR/(1+WOR)) ;

% surface tension of liquid mixure - sigma_l (dyn/cm)
sigma_l = (sigma_o*1/(1+WOR)) + (sigma_w*WOR/(1+WOR)) ;

% liquid viscosity number - N_l
N_l = 0.15726 * miu_l * (1/(roh_l*sigma_l^3))^0.25 ;

% calculate from graph - CN_l
CN_l = 0.0611*N_l^3 - 0.0929*N_l^2 + 0.0505*N_l + 0.0019;

% superficial liquid velocity - v_sl (ft/s)
v_sl = 5.615*q_l/(86400*At)*((Bo*1/(1+WOR)) + (Bw*WOR/(1+WOR))) ;

% liquid viscosity number - N_LV
N_LV = 1.938*v_sl*(roh_l/sigma_l)^0.25 ;

% superficial gas velocity - v_sg (ft/s)
v_sg = q_l*(GLR-
Rs*1/(1+WOR))/(86400*At)*(14.7/p_bar)*((T_bar+460)/520)*z ;

% gas velocity number - N_GV
```

```
N_GV = 1.938*v_sg*(roh_l/sigma_l)^0.25 ;

% flow regime check - L_B
L_B = 1.071 - 0.2218*(v_sl+v_sg)^2/dt;

if L_B<0.13
    L_B = 0.13;
end

% calculating BB
BB = v_sg/(v_sl+v_sg);

% check difference BB-L_B
diff = BB - L_B;

if diff<0
    %orkiszewski;
    type = 'ork';
else
    type = 'HB';
end

% continuing H&B ... pipe diameter number - N_D
N_D = 120.872*dt*(roh_l/sigma_l)^0.5 ;

% holdup correlation function - phi_HL
phi_HL = (N_LV/N_GV^0.575)*(p_bar/14.7)^0.1*(CN_l/N_D) ;

% calculate from graph - HL_si
if phi_HL<=3e-4
    HL_si = 16*phi_HL^0.4185;
end
if phi_HL>3e-4
    HL_si = 0.2628*log(phi_HL) + 2.6553;
end
if phi_HL>1e-3
    HL_si = 0.1099*log(phi_HL) + 1.5945;
end


% secondary correlation factor - phi_s
phi_s = N_GV*(N_l)^0.38/(N_D)^2.14 ;

% calculate from graph - si
if phi_s<=0.025
    si = 27170*phi_s^3 - 317.52*phi_s^2 + 0.5472*phi_s + 0.9999;
end
if phi_s>0.025
    si = -533.33*phi_s^2 + 58.524*phi_s + 0.1171;
end
if phi_s>0.055
    si = 2.5714*phi_s + 1.5962 ;
end
```

```
% liquid hold up - HL
HL = HL_si*si;

if HL>1
    HL=1;
end

% two phase reynolds number - Re
Re = 2.2e-2*q_m/(dt*miu_l^HL*miu_g^(1-HL));

% calculate friction factor
f = (1.8*log10(6.9/Re + (e/3.7/dt)^(10/9)))^(-2);

% two phase density - roh_m (lb/ft3) methods -(1) HB, (2) no slippage
roh_m1 = roh_l*HL + roh_g*(1-HL) ;
GOR= GLR;
roh_m2 = (350*SGo+0.0764*SGg*GOR+350*SGw*WOR)/(5.61*Bo+5.61*WOR+(GOR-
Rs)*Bg) ;

if roh_m1>roh_m2
    roh_m = roh_m1;
else
    roh_m = roh_m2;
end

% calculations at p1 and T1, two phase velocity - v_m1 (ft/s)
T = (T1-32) * 5/9 ;              % convert to degree C to read off the
grid
p = 0.0689475729 * p1  ;         % convert to bara to read off the
grid
[SGo, SGg, Rs, Bo, Bg, miu_o, miu_g, sigma_o, z, SGw, Bw, miu_w,
sigma_w] = calculate_prop(T,p);

v_sl1 = 5.615*q_l/(86400*At)*((Bo*1/(1+WOR)) + (Bw*WOR/(1+WOR))) ;
v_sg1 = q_l*(GLR-Rs*1/(1+WOR))/(86400*At)*(14.7/p1)*((T1+460)/520)*z
;
v_m1 = v_sl1 + v_sg1 ;


% calculations at p2 and T2, two phase velocity - v_m2 (ft/s)
T = (T2-32) * 5/9;               % convert to degree C to read off the
grid
p = 0.0689475729 * p2;           % convert to bara to read off the
grid
[SGo, SGg, Rs, Bo, Bg, miu_o, miu_g, sigma_o, z, SGw, Bw, miu_w,
sigma_w] = calculate_prop(T,p);

v_sl2 = 5.615*q_l/(86400*At)*((Bo*1/(1+WOR)) + (Bw*WOR/(1+WOR))) ;
v_sg2 = q_l*(GLR-Rs*1/(1+WOR))/(86400*At)*(14.7/p2)*((T2+460)/520)*z
;
v_m2 = v_sl2 + v_sg2 ;


% delta (v_m)^2 (ft^2/s^2)
```

```
delta_v_m_sq = (v_m1)^2 - (v_m2)^2 ;

% height for that pressure drop (ft)
delta_h = (144*(p2-p1)-
roh_m*delta_v_m_sq/(2*gc))/(roh_m+(f*q_l^2*m_t^2)/(7.41e10*dt^2*roh_m
));
```

inputs.m

```
% Inputs

WOR = 0;                                    % unitless?
GLR = 300;                                  % scf/bbl

%q_l = 480;                                 % bpd
liquid flowrate
q_w = 0;                                    %                                   water
flowrate
d = 1.9995;                                 %
% in                                        well tubing diameter
r = d/2/12;                                 % ft                                well
tubing radius
r_to = r + 1.5;                             % ft
radius tubing outside
dt = d/12;                                  % ft                                well
tubing diameter
At = 3.142*dt^2/4;                          % ft2                               well
tubing corss sectional area
e = 0.00015;                                % pipe roughness (unitless?)
gc = 32.2;                                  % field unit factor
p_TH = 200;                                 % psig
tubing head pressure
T_TH = 80;                                  % degree F
tubing head temperature
h_well = 9700;                              % ft                                well
height
T_BH = 212;                                 % degree F
bottomhole temperature

p1 = p_TH;
T1 = T_TH;
```

interpolateT.m

```
% calculates the iterated T for HB calculation

for i = 1:kmax
    if xh(i, Titeration-1)== h
        h1 = xh(i, Titeration-1);
        h2 = xh(i, Titeration-1);
        i1 = i;
        i2 = i;
        break
    else if xh(i, Titeration-1)<h
        h1 = xh(i, Titeration-1);
    else
        h2 = xh(i, Titeration-1);
        i1 = i-1;
        i2 = i;
        break
        end
    end
end
    if i1 == 0
        i1 = 1;
        h1 = xh(i1, Titeration-1);
    end

dh = h1-h2;

if dh == 0
    dh = 2;
end

if h>h2
    h2=h;
end


% calculating T_h
T_h = T_hi(i1, Titeration-1) + (T_hi(i1, Titeration-1)-T_hi(i2,
Titeration-1))*(h-h1)/(dh);

if h2==h
    T_h = T_BH;
end
```

main.m

```
% clc;
% clear;

%q_l=480;
inputs;

h = 0;
delta_h = 0;

fprintf('\n\ndepth(ft)\twell T(F)\twell P(psig)\tholdup\t\tflow
type\n')
fprintf('%.2f\t\t%.2f\t\t%.2f\n', h, T1, p1)



for Titeration=1:10

    k=1;
    p1 = p_TH;
    T1 = T_TH;

    if Titeration ~= 1
        calculateT;
    end


    while h < h_well

        k = k + 1;
        guessT2p2;

        T2check=1;
        while T2check==1
            HB;
            h = h + delta_h;
            checkT2match;
        end

        fprintf('%.2f\t\t%.2f\t\t%.2f\t\t\t%.2f\t\t%s\n', h, T2, p2,
HL, type)

        xT1(k, Titeration) = T1;
        xh(k, Titeration) = h;
        xT2(k, Titeration) = T2;
        xp2(k, Titeration) = p2;
        xHL(k, Titeration) = HL;
        xQ(k, Titeration) = v_m2*At;
        xdelta_h(k, Titeration) = delta_h;
        xdelta_p(k, Titeration) = -1*(p1-p2);
        xdelta_v_m_sq(k, Titeration) = delta_v_m_sq;

        p1 = p2;
```

```
        T1 = T2;

    end

    fprintf('\n\n\n')

    ks (Titeration) = k;


    if Titeration == 1
        kmax = k;
    else
        if kmax<ks(Titeration)
            xh(kmax:ks(Titeration), 1) = xh(kmax,1);
            xT2(kmax:ks(Titeration), 1) = xT2(kmax,1);
            xp2(kmax:ks(Titeration), 1) = xp2(kmax,1);
            xHL(kmax:ks(Titeration), 1) = xHL(kmax,1);
            xQ (kmax:ks(Titeration), 1) = xQ(kmax,1);
            xv (kmax:ks(Titeration), 1) = xv(kmax,1);
        else
            xh(k:kmax, Titeration) = h;
            xT2(k:kmax, Titeration) = T_BH;
            xp2(k:kmax, Titeration) = p2;
            xHL(k:kmax, Titeration) = HL;
            xQ (k:kmax, Titeration) = v_m2*At;
            xv (k:kmax, Titeration) = v_m2;
        end
    end

    xh(1, Titeration) = 0;
    xT2(1, Titeration) = xT2(2, Titeration);
    xp2(1, Titeration) = xp2(2, Titeration);
    xHL(1, Titeration) = xHL(2, Titeration);
    xQ (1, Titeration) = xQ (2, Titeration);

    h=0;

    if Titeration ~= 1
        if max(abs(xT2(k, Titeration)-xT2(k, Titeration-1)))<0.1
            break
        end
    end


end


fprintf('\n\n')
T_hi(1:kmax, Titeration) = T_hi(1:kmax, Titeration-1);

plotHB;
figure;
```

mainQ.m

```
clc;
clear;

Td = 10;

for q_l = 80000:20000:160000; %100000:1000:105000
    main;
    xp2Q(q_l/20000-3)=xp2(length(xp2),Titeration);%1000-
99)=xp2(kmax,Titeration);
    %xQQ(q_l/50-5)=xQ(1,Titeration);
end

q_l = 80000:20000:160000; %100000:1000:105000;

plot(xp2Q/14.5,q_l/6.3)
xlabel('pressure(bara)')
ylabel('q_l(m^3/d)')
```

mainSINGLE.m

```
clc;
clear;

q_l=600;      % Liquid flow rate [bpd]
Td =10;       % Diimensionless time
inputs;

h = 0;        % Declating variables
delta_h = 0;

fprintf('\n\ndepth(ft)\twell T(F)\twell P(psig)\tholdup\t\tflow
type\n')
fprintf('%.2f\t\t%.2f\t\t%.2f\n', h, T1, p1)


for Titeration=1:10

    k=1;                          % Declaring counter
    xh(1, Titeration) = 0;        % Initial values stored in matrix
    xT2(1, Titeration) = 0;
    xp2(1, Titeration) = 0;
    xHL(1, Titeration) = 0;
    xQ (1, Titeration) = 0;


    p1 = p_TH;                    % Declating p1 and T1 for the 1st
Hagedorn and Brown iteration
    T1 = T_TH;

    if Titeration ~= 1
        calculateT;               % Solving Ramey's temeprature model
    end


    while h < h_well

        k = k +1;

        guessT2p2;                % Declaring value for p2 and T2 for
Hagedorn and Brown method

        T2check=1;
        while T2check==1
            HB;                   % Running the Hagedorn and Brown
method
            h = h + delta_h;      % Calulating depth
            checkT2match;         % Checking if iteration is needed for
the Hagedorn and Brown method
        end
```

```
        fprintf('%.2f\t\t%.2f\t\t%.2f\t\t\t\t%.2f\t\t%s\n', h, T2, p2,
HL, type)

        xT1(k, Titeration) = T1;     % Storing values from this run
        xh(k, Titeration) = h;
        xT2(k, Titeration) = T2;
        xp2(k, Titeration) = p2;
        xHL(k, Titeration) = HL;
        xv(k, Titeration) = v_m2;
        xQ(k, Titeration) = v_m2*At;
        xdelta_h(k, Titeration) = delta_h;
        xdelta_p(k, Titeration) = -1*(p1-p2);
        xdelta_v_m_sq(k, Titeration) = delta_v_m_sq;

        p1 = p2;                    % Declaring value for p1 and T1 for
next Hagedorn and Brown iteration
        T1 = T2;

    end

    fprintf('\n\n\n')

    ks (Titeration) = k;


    if Titeration == 1
        kmax = k;
    else
        if kmax<ks(Titeration)
            xh(kmax:ks(Titeration), 1) = xh(kmax,1);
            xT2(kmax:ks(Titeration), 1) = xT2(kmax,1);
            xp2(kmax:ks(Titeration), 1) = xp2(kmax,1);
            xHL(kmax:ks(Titeration), 1) = xHL(kmax,1);
            xQ (kmax:ks(Titeration), 1) = xQ(kmax,1);
            xv (kmax:ks(Titeration), 1) = xv(kmax,1);
        else
            xh(k:kmax, Titeration) = h;
            xT2(k:kmax, Titeration) = T_BH;
            xp2(k:kmax, Titeration) = p2;
            xHL(k:kmax, Titeration) = HL;
            xQ (k:kmax, Titeration) = v_m2*At;
            xv (k:kmax, Titeration) = v_m2;
        end
    end

    xh(1, Titeration) = 0;
    xT2(1, Titeration) = xT2(2, Titeration);
    xp2(1, Titeration) = xp2(2, Titeration);
    xHL(1, Titeration) = xHL(2, Titeration);
    xQ (1, Titeration) = xQ (2, Titeration);

    h=0;

    if Titeration ~= 1
        if max(abs(xT2(k, Titeration)-xT2(k, Titeration-1)))<0.1
```

```
            break
        end
    end


end


fprintf('\n\n')
T_hi(1:kmax, Titeration) = T_hi(1:kmax, Titeration-1);

plotHB;
```

mainTd.m

```
clc;
clear;

q_1= 480; %bpd


for Td = 10:5:20
    main
    xT2Td(1:kmax, Td/5-1) = xT2(1:kmax,Titeration);
    xhTd(1:kmax, Td/5-1) = xh(1:kmax,Titeration);
end


for column = 1:3
    for row = 2:length(xT2Td)
        if xT2Td(row, column)==0
            xT2Td(row, column)=xT2Td(row-1, column)
        end
        if xhTd(row, column)==0
            xhTd(row, column)=xhTd(row-1, column)
        end
    end
end



plot((xT2Td-32)*5/9,-0.3048*xhTd)
ylabel('height(m)')
xlabel('T(^oC)')
```

plot_final_run_only.m

```
subplot(1,4,1), plot((xT2(1:length(xT2),3)-32)*5/9,-
0.3048*xh(1:length(xT2),3))
ylabel('height(m)')
xlabel('T(^oC)')

subplot(1,4,2), plot(xp2(1:length(xT2),3)/14.5,-
0.3048*xh(1:length(xT2),3))
xlabel('p(bara)')

subplot(1,4,3), plot(xHL(1:length(xT2),3),-
0.3048*xh(1:length(xT2),3))
xlabel('HL')

subplot(1,4,4), plot(xQ(1:length(xT2),3)/0.00041,-
0.3048*xh(1:length(xT2),3))
xlabel('Q(m^3/d)')
```

plot_progression.m

```
subplot(1,7,1), plot((xT2(1:length(xT2),1)-32)*5/9,-
0.3048*xh(1:length(xT2),1),'g','LineWidth',2)
ylabel('height(m)')
xlabel('T(^oC)')

%subplot(1,7,2)

subplot(1,7,3), plot(xp2(1:length(xT2),1)/14.5,-
0.3048*xh(1:length(xT2),1),'r','LineWidth',2)
xlabel('p(bara)')

subplot(1,7,4), plot((xT2(1:length(xT2),2)-32)*5/9,-
0.3048*xh(1:length(xT2),2),'r','LineWidth',2)
xlabel('T(^oC)')

%subplot(1,7,5)

subplot(1,7,6), plot(xp2(1:length(xT2),2)/14.5,-
0.3048*xh(1:length(xT2),2),'b','LineWidth',2)
xlabel('p(bara)')

subplot(1,7,7), plot((xT2(1:length(xT2),3)-32)*5/9,-
0.3048*xh(1:length(xT2),3),'b','LineWidth',2)
xlabel('T(^oC)')
```

plotHB.m

```
subplot(1,4,1), plot((xT2-32)*5/9,-0.3048*xh)
ylabel('height(m)')
xlabel('T(^oC)')

subplot(1,4,2), plot(xp2/14.5,-0.3048*xh)
xlabel('p(bara)')

subplot(1,4,3), plot(xHL,-0.3048*xh)
xlabel('HL')

subplot(1,4,4), plot(xQ/0.00041,-0.3048*xh)
xlabel('Q(m^3/d)')
```

Wax Model Code

File name	Description
wax.m	Calculates and plots the results of two different wax models

wax.m

```
% Wax Crystallinity Models

for i = 1:length(xT2)

    % Zougari and Sopkow Model Constants
    n = 3;
    c1 =0.1e-8;
    c2 = -9.8e6;
    Tmin = -100;                % degrees C
    Tmax = 80;                  % degrees C
    lambda_eff = 0.1;           % effective cooling rate 0.1C/min
    lambda = 5;                 % cooling rate

    % Zougari and Sopkow Model Calculations
    c1n = c1*(Tmax-Tmin)*((Tmax-Tmin)/lambda_eff)^n;
    c2n = c2/(Tmax-Tmin)^3;
    c3n = Tmin/(Tmax-Tmin)^3;
    theta = (((xT2(i,Titeration)-32)*(5/9))-Tmin)/(Tmax-Tmin);
    K_theta = c1n*exp(-1*c2n/((theta+c3n)*theta^2))*theta^(n+1);
    phi = lambda/lambda_eff;
    crystallinity1(i) = 1 - exp(-1*K_theta/(phi)^n);


    % Begatin et al. Model Constants
    C = 0.0001;
    WAT = 176;                  % degrees F

    % Begatin et al. Model Calculations
    if i == length(xT2)
        dTdz = abs(xT2(i,Titeration)-xT2(i-
1,Titeration))/abs(xh(i,Titeration)-xh(i-1,Titeration));
    else
        dTdz = abs(xT2(i,Titeration)-
xT2(i+1,Titeration))/abs(xh(i,Titeration)-xh(i+1,Titeration));
    end
    crystallinity2(i) = 1 - exp((-1*C*(WAT-
xT2(i,Titeration))^2)/abs(dTdz*xv(i,Titeration)));
```

```
    % Matrices for plotting purposes
    xT2wax(i) = xT2(i,Titeration);
    xhwax(i) = xh(i,Titeration);
end

% Plotting the outputs of the two models in SI units

subplot(1,3,1)
plot((xT2wax(1,1:157)-32)*5/9, crystallinity1(1,1:157))
ylabel('Crystallinity')
xlabel('T(^oC)')

subplot(1,3,2)
plot((xT2wax(1,1:157)-32)*5/9, -0.3048*xhwax(1,1:157))
xlabel('T(^oC)')
ylabel('Depth(m)')

subplot(1,3,3)
plot(crystallinity1(1,1:157), -0.3048*xhwax(1,1:157))
xlabel('Crystallinity')
ylabel('Depth(m)')

figure

subplot(1,3,1)
plot((xT2wax(1,1:157)-32)*5/9, crystallinity2(1,1:157))
ylabel('Crystallinity')
xlabel('T(^oC)')

subplot(1,3,2)
plot((xT2wax(1,1:157)-32)*5/9, -0.3048*xhwax(1,1:157))
xlabel('T(^oC)')
ylabel('Depth(m)')

subplot(1,3,3)
plot(crystallinity2(1,1:157), -0.3048*xhwax(1,1:157))
xlabel('Crystallinity')
ylabel('Depth(m)')
```

APPENDIX B: HEAT BALANCE IN HORIZONTAL WELL COMPLETION

This report describes the concept behind the heat balance calculations utilized in the horizontal oil well simulator developed as part of the research work done by Thanyamanta (2009) to compute temperature changes in the length of the well, which in turn was used to predict asphaltene precipitation.

General equation derivation

Initially, the undisturbed reservoir has a temperature of T_{res}. The temperature of the oil, T_{oil}, is the same as T_{res}, since they are in contact for a long time and hence have reached thermal equilibrium.

$$T_{res} = T_{oil}$$

Once the well is drilled into place and the oil start flowing, the temperature of the oil changes. This is because of the following.

(1) heat conduction from the reservoir through the wall of the well tubing

(2) frictional heat production

(3) expansion (Joule-Thomson effect)

(4) inflow of oil through the slots located over the length of the horizontal well

In order to include all the above aspects into the energy balance, the following general equation is used to calculate the total energy in the system.

$$\dot{e}_{total} = \dot{e}_{conv} + \dot{e}_{cond} + \dot{e}_{w}$$

The total energy is a sum of the energy by convection, conduction and due to work. Convection includes the kinetic energy and internal energy of the molecules comprising the system.

$$\dot{e}_{conv} = \left(\frac{1}{2}\rho v^2 + \rho \hat{u}\right)q$$

Conducted energy includes all forms of energy transfer due to the difference of temperature equilibrium. The difference in the temperature of the fluid and tubing wall is accounted by this, and can be written in the following general form with a general coefficient of heat transfer.

$$\dot{e}_{cond} = -k.\nabla T$$

The work done by the molecules in the system is to overcome tensor stress. Tensor stress is a combination of normal and shear stress.

$$\dot{e}_w = \pi'.q$$
$$\pi' = P\delta + \tau$$

Substituting each of the above components breakdowns into the general total energy equation yields the following.

$$\dot{e}_{total} = \left(\frac{1}{2}\rho v^2 + \rho \hat{u}\right)q + \dot{e}_{cond} + \pi'.q$$

A series of algebraic manipulations are performed on this equation to be able to simplify it. In the first step, the equation is substituted with the value of tensor stress.

$$\dot{e}_{total} = \left(\frac{1}{2}\rho q v^2 + \rho q \hat{u}\right) + \dot{e}_{cond} + \left(Pq + \tau q\right)$$

The term containing pressure is multiplied by density in both the numerator and denominator, such that overall it has the effect of multiplying by 1.

$$\dot{e}_{total} = \frac{1}{2}\rho q v^2 + \rho q \hat{u} + \dot{e}_{cond} + P\rho q \frac{1}{\rho} + \tau q$$

Inverse of density yields specific volume. This term is then rearranged and combined with specific internal energy term.

$$\dot{e}_{total} = \frac{1}{2}\rho q v^2 + \rho q \hat{u} + \dot{e}_{cond} + P\rho q \hat{v} + \tau q$$

$$\dot{e}_{total} = \frac{1}{2}\rho q v^2 + \rho q\left(\hat{u} + P\hat{v}\right) + \dot{e}_{cond} + \tau q$$

From thermodynamics we know that sum of specific internal energy and the product of pressure and internal volume gives the specific enthalpy. This is substituted in the next step.

$$\dot{e}_{total} = \frac{1}{2}\rho q v^2 + \rho q \hat{H} + \dot{e}_{cond} + \tau q$$

For fully developed flow, it is known that enthalpy is the dominating term. Hence kinetic energy and shear stress terms are ignored. Thus the following equation is obtained.

$$\dot{e}_{total} = \rho q \hat{H} + \dot{e}_{cond}$$

This equation describes the energy transfer in the system. It can be applied to each discrete segment in the network/grid of the reservoir solver. The system can be classified by the direction of flow to break the calculations down – in the axial and radial flow directions. This gives the following equations.

$$\dot{e}_{axial} = \rho q \hat{H}$$
$$\dot{e}_{radial} = \rho q \hat{H} + Q$$

These equations are in par with the physical system of the model. Following is a schematic of the system used in model.

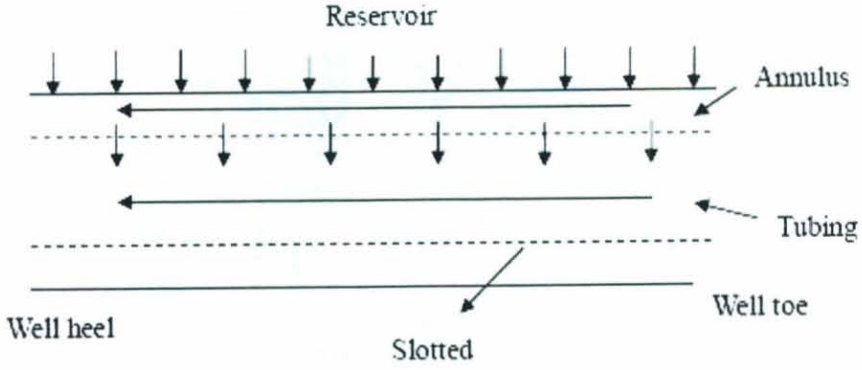


Figure: Reservoir Completion

The equations show that energy flow in the axial direction is due to the change of enthalpy in the fluid and expansion. The energy transfer in the radial direction occurs due to both (1) the enthalpy of the fluid flowing in from the reservoir, and (2) heat transfer from the reservoir to the fluid in annulus, which is denoted by Q.

Q accounts for the entire process of conductive heat transfer through the annulus wall, and then convective heat transfer from the inside wall of annulus into the fluid by convection.

In the sections below, we work on further developing the above axial and radial energy equations.

In the axial energy transfer equation, the enthalpy term can be expanded to show a change in temperature and Joule-Thomson expansion as shown below.

$$\dot{e}_{axial} = \rho q \Delta \hat{H}$$
$$\dot{e}_{axial} = \rho q \hat{C}_p \left(T - T^o\right) + q\left(1 - \beta T\right)\left(P - P^o\right)$$

Joule-Thomson effect describes the change in temperature due to a pressure change. This physical effect is widely used for liquefying gas. For example, in order to liquefy carbon dioxide, the gas is passed though a nozzle from high pressure to low pressure. The expansion causes the gas to cool. When sufficient

pressure drop is applied, we get enough temperature drop to convert to liquid. For other material, the pressure drop can cause increase in temperature. This is determined by the specific Joule-Thomson coefficient, which is described below.

$$K_{JT} = \frac{\beta T - 1}{\rho \hat{C}_p}$$

The above equation can be rearranged to be more suitable for our purpose.

$$\beta T - 1 = -K_{JT} \rho \hat{C}_p$$

The left hand term in the above equation appears in our axial energy transfer equation. If we substitute it in, it yields the following.

$$\dot{e}_{axial} = \rho q \hat{C}_p (T - T^o) + q\left(-K_{JT} \rho \hat{C}_p\right)(P - P^o)$$

$$\boxed{\dot{e}_{axial} = \rho q \hat{C}_p (T - T^o) - \rho q \hat{C}_p K_{JT} (P - P^o)}$$

In similar fashion, the equation for energy transfer in radial direction can be expanded.

$$\dot{e}_{radial} = \rho q \Delta \hat{H} + Q$$

$$\dot{e}_{radial} = \rho q \hat{C}_p (T - T^o) + A_r U (T_o - T_b)$$

Unlike in the case of axial heat transfer, Joule-Thomson expansion is ignored for radial heat transfer. Also, the outside temperature, T^o, is assumed to be the same

as the reservoir temperature. Thus, our final radial heat transfer equation is shown below.

$$\boxed{\dot{e}_{radial} = \rho q \hat{C}_p (T - T^o) + A_r U (T_{res} - T_b)}$$

Equation derivation for model grid

In this section, the general equation derived above is applied to the specific grid network used to model the flows in the producer tubing system. The following is the schematic of the producer grid.

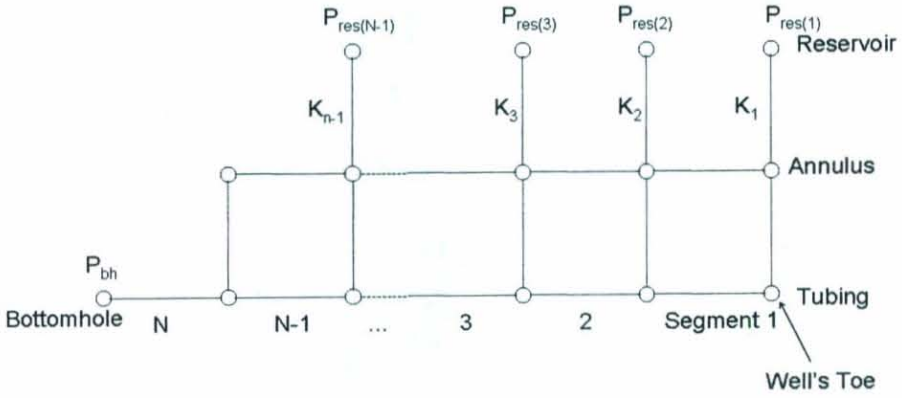


Figure: Completion grid

The heat balance system is applied to each segment of the grid. Looking at the annulus, the sources of heat inflow are (1) the heat in the fluid in the segment, (2) the heat in the fluid coming in from the reservoir (radial flow). The heat outflows due to (1) fluid transfer to the next segment (axial flow), and (2) fluid transfer

through the slots into the tubing (radial flow). This gives the following equation for segment 1.

$$\sum \dot{E}_{segment} = \dot{E}_1 + \dot{E}_I - \dot{E}_2 - \dot{E}_S$$

Previously, equations were made for heat transfer in axial and radial directions. When applying these equations to the above balance, this yields the following.

$$\sum \dot{E}_{segment} = \left(\rho_1 q_1 \hat{H}_1\right) + \left(\rho_I q_I \hat{H}_I + Q_I\right) - \left(\rho_2 q_2 \hat{H}_2\right) - \left(\rho_S q_S \hat{H}_S + Q_S\right)$$

If we do a mass balance at node 2, the equation looks as shown below. It can be rearranged to solve for node 1 parameters, which is then substituted in the above equation.

$$\rho_2 q_2 = \rho_1 q_1 + \rho_I q_I - \rho_S q_S$$
$$\rho_1 q_1 = \rho_2 q_2 - \rho_I q_I + \rho_S q_S$$

Upon substitution, we get the following equation.

$$\sum \dot{E}_{segment} = \left(\rho_2 q_2 \hat{H}_1 - \rho_I q_I \hat{H}_1 + \rho_S q_S \hat{H}_1\right) + \left(\rho_I q_I \hat{H}_I + Q_I\right) - \left(\rho_2 q_2 \hat{H}_2\right) - \left(\rho_S q_S \hat{H}_S + Q_S\right)$$
$$\sum \dot{E}_{segment} = \rho_2 q_2 \left(\hat{H}_1 - \hat{H}_2\right) + \rho_I q_I \left(\hat{H}_1 - \hat{H}_I\right) + \rho_S q_S \left(\hat{H}_1 - \hat{H}_S\right) + Q_I - Q_S$$

The following assumptions are made at this point:

(1) Enthalpy of the fluid at the annulus segment ($\hat{H}_1$) is the same as the enthalpy of the fluid going down the slots ($\hat{H}_s$). This is because part of the fluid at annulus segment goes down the slots.

(2) Q_S is zero, because there is no heat transfer from the walls into the slots.

After applying the above assumptions and using the appropriate expansion for enthalpy, we get the following equation.

$$\sum \dot{E}_{segment} = \rho_2 q_2 \left[\hat{C}_p (T_1 - T_2) + \hat{C}_p K_{JT} (P_1 - P_2) \right] + \rho_I q_I \left[\hat{C}_p (T_1 - T_{res}) \right] + Q_I$$

$$\sum \dot{E}_{segment} = \rho_2 q_2 \left[\hat{C}_p (T_1 - T_2) + \hat{C}_p K_{JT} (P_1 - P_2) \right] + \rho_I q_I \left[\hat{C}_p (T_1 - T_{res}) \right] + A_r U (T_{res} - T_o)$$

This is the final equation that is used for energy balance. In the steady state system we are concerned about, the sum of the input and output would be zero, i.e. the left hand side has a value of zero.

$$0 = \rho_2 q_2 \left[\hat{C}_p (T_1 - T_2) + \hat{C}_p K_{JT} (P_1 - P_2) \right] + \rho_I q_I \left[\hat{C}_p (T_1 - T_{res}) \right] + A_r U (T_{res} - T_o)$$

The equation uses parameters that can be obtained from tables and from data of the reservoir. However, special calculations need to be done to get the value for U. This is described in details in the next section.

A point to be noted is that the final equations were obtained using only annulus segments. Similar equations can be setup for nodes in the tubing as well. Thus for a well with N segments, we get 2N equations. However, we have 2N+1 temperatures to calculate. This is dealt with by assuming that node 1 temperatures are known to be equal to the reservoir temperature.

Also to be noted is that the equations shown in this document are for only single phase. Asphaltene precipitation was described for a two phase system. This can be included by taking into consideration the specific heat capacities and flow rates of each phase in the energy equation.

Calculating U

This method was developed by Dawkrajai et al. (2005). U is the convective heat transfer coefficient from the wall to the fluid. It is described as follows.

$$U_{I-b} = \frac{Q}{(T_b - T_I)A}$$

Where, Q is the heat being transferred, A is the heat transfer surface area. The temperature difference is the only unknown. In the next parts, the objective is to get an expression to calculate the temperature difference between the bulk fluid inside the tube and the initial reservoir temperature. Following is a schematic of the heat transfer layers.

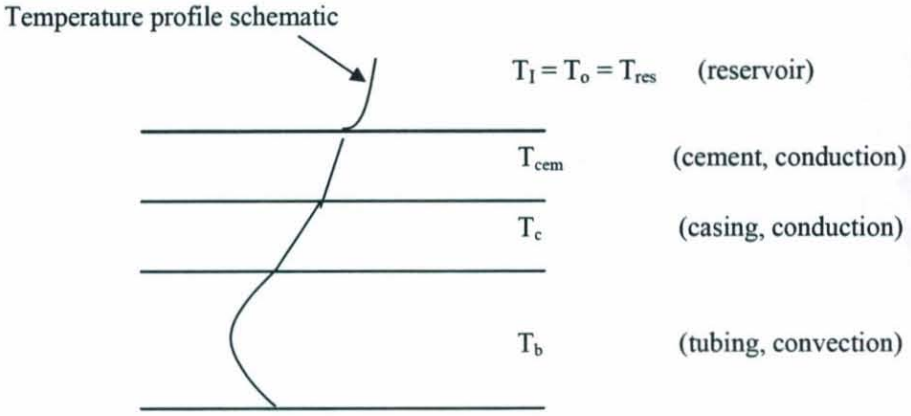


Figure: Temperature profile in the completion and surroundings

The heat being transferred between each layer is assumed to be same, i.e. no heat loss. This heat is denoted by Q. The following equation shows the conduction differential equation (Fourier's law) between the outer wall and casing. It is then integrated.

$$Q = -2\pi(1-\gamma)rk_c\frac{dT}{dr}$$

$$Q\int_{R}^{R_c}\frac{1}{r}.dr = -2\pi(1-\gamma)k_c\int_{T_c}^{T_{cem}}dT$$

$$Q\ln\left(\frac{R_c}{R}\right) = -2\pi(1-\gamma)k_c\left(T_{cem}-T_c\right)$$

$$\boxed{T_{cem}-T_c = -\frac{Q\ln\left(\frac{R_c}{R}\right)}{2\pi(1-\gamma)k_c}}$$

Similarly, the equation for heat transfer between the casing and cement.

$$Q = -2\pi(1-\gamma)rk_{cem}\frac{dT}{dr}$$

$$Q\int_{R_c}^{R_{cem}}\frac{1}{r}.dr = -2\pi(1-\gamma)k_{cem}\int_{T_{cem}}^{T_I}dT$$

$$Q\ln\left(\frac{R_{cem}}{R}\right) = -2\pi(1-\gamma)k_{cem}\left(T_I - T_{cem}\right)$$

$$\boxed{T_I = T_{cem} - \frac{Q\ln\left(\frac{R_{cem}}{R_c}\right)}{2\pi(1-\gamma)k_{cem}}}$$

The conductive heat transfer between the cement layer and bulk fluid is given by the following equation.

$$Q = -2\pi(1-\gamma)Rh\left(T_c - T_b\right)$$

$$\boxed{T_b = T_c + \frac{Q}{2\pi(1-\gamma)Rh}}$$

Now we are able to calculate $T_b - T_I$ as follows.

$$T_b - T_I = T_c + \frac{Q}{2\pi(1-\gamma)Rh} - T_{cem} + \frac{Q\ln\left(\frac{R_{cem}}{R_c}\right)}{2\pi(1-\gamma)k_{cem}}$$

$$T_b - T_I = \frac{Q}{2\pi(1-\gamma)Rh} + \frac{Q\ln\left(\frac{R_{cem}}{R_c}\right)}{2\pi(1-\gamma)k_{cem}} + T_c - T_{cem}$$

Substituting in the equation for $T_c - T_{cem}$ yields the following.

$$T_b - T_I = \frac{Q}{2\pi(1-\gamma)Rh} + \frac{Q\ln\left(\frac{R_{cem}}{R_c}\right)}{2\pi(1-\gamma)k_{cem}} + \frac{Q\ln\left(\frac{R_c}{R}\right)}{2\pi(1-\gamma)k_c}$$

$$T_b - T_I = \frac{Q}{2\pi(1-\gamma)R}\left[\frac{1}{h} + \frac{R\ln\left(\frac{R_{cem}}{R_c}\right)}{k_{cem}} + \frac{R\ln\left(\frac{R_c}{R}\right)}{k_c}\right]$$

Substituting this temperature difference term into the original U equation gives the following.

$$U_{I-b} = \frac{Q}{(T_b - T_I)A}$$

$$U_{I-b} = \frac{Q}{(T_b - T_I)2\pi(1-\gamma)R}$$

$$U_{I-b} = \frac{Q}{\frac{Q}{2\pi(1-\gamma)R}\left[\frac{1}{h} + \frac{R\ln\left(\frac{R_{cem}}{R_c}\right)}{k_{cem}} + \frac{R\ln\left(\frac{R_c}{R}\right)}{k_c}\right]2\pi(1-\gamma)R}$$

$$\boxed{U_{I-b} = \frac{1}{h} + \frac{R\ln\left(\frac{R_{cem}}{R_c}\right)}{k_{cem}} + \frac{R\ln\left(\frac{R_c}{R}\right)}{k_c}}$$

In order to calculate the value for *h* in the above equation for laminar flow, we use the following equation.

$$h = 3.656\frac{k\mu}{2R}$$

For turbulent flow, we need to calculate the *Nu* number before we can calculate *h*.

$$Nu = 0.023\,\mathrm{Re}^{0.8}\,\mathrm{Pr}^{0.33}\left(\frac{\mu}{\mu_{water}}\right)^{0.14}$$

$$h = \frac{Nu \cdot k}{d}$$