Artificial Lift in Geothermal Wells:
A Study to Binary Cycle Geothermal Power Plants with Gas Lift in the Production Well

F.W.J. Niewold

Master of Science Thesis
Report Number P&E-2738

TU Delft Delft University of Technology

if Creating energy.

ARTIFICIAL LIFT IN GEOTHERMAL WELLS

A STUDY TO BINARY CYCLE GEOTHERMAL POWER PLANTS WITH GAS LIFT IN THE PRODUCTION WELL

by

F.W.J. Niewold

in partial fulfillment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Delft University of Technology,
to be defended on Monday February 27, 2017 at 10:00 AM

Thesis committee:	Prof.dr.ir. T.J.H. Vlugt,	TU Delft
	Dr.ir. C.A. Infante Ferreira,	TU Delft, supervisor
	Dr.ir. S.C. Jansen,	TU Delft
	J.H. Kleinlugtenbelt, MSc	IF Technology, supervisor

TU Delft Delft University of Technology

if Creating energy.

ABSTRACT

An alternative method for generating electric power from hot geothermal reservoirs (200 – 250 ºC) has been proposed. This study was specifically concerned with the effect of artificial lift in geothermal wells on the thermodynamic performance of geothermal power plants. The idea is to prevent flashing of geothermal fluid and consequently replace conventional single-flash power plants with binary cycle power plants. Three positive effects are to be expected. Previous studies have shown that thermal efficiencies of binary cycle power plants are generally higher than single-flash power plants. Secondly, non-condensable gases (NCG) can stay in the solution and subsequently they can be reinjected in the geothermal reservoir. At the present in common geothermal power plants almost all NCG are vented to the atmosphere. Finally, calcite scaling in the wellbore is reduced. The major objective of this study was to investigate the technical feasibility of artificial lift in a geothermal well connected to a binary cycle power plant and to compare its thermodynamic performance to a self-flowing flashing geothermal well connected to a single-flash power plant. Additionally, the CO_2 emission of these two power plants were calculated and compared.

In this work different methods to pressurize wells and lift geothermal fluids, used in geothermal- and petroleum applications, were examined. Gas lift was considered the most appropriate application for geothermal fluids in the range of 200 – 250 ºC. A comprehensive steady-state mathematical model was developed in MATLAB for the single-flash power plant and the binary cycle power plant, covering the system of a reservoir, a production well, a geothermal power plant and an injection well. Additionally, a geothermal fluid property (GFP) model found in literature was implemented. In this work modifications to this GFP model were carried out to simulate the liquid phase and two-phase flow. It was assumed that geothermal fluid is a ternary system consisting of H_2O – NaCl – CO_2. Thermo-hydraulic numerical models were developed for the wells. The drift-flux approach was used to simulate two-phase flow within the wellbore. The geothermal power plant models included all equipment generating or demanding power and all equipment causing a phase change. The different systems of the mathematical model were quantitatively validated with data from literature. The simulated pressure and temperature profiles as a function of well depth of the production well without gas lift (self-flowing) were validated with experimental data of six randomly chosen existing production wells. The production well model with gas lift was validated qualitatively in the results section, because of the novelty of this technology in geothermal production wells and the absence of experimental data.

To compare the two power plant facilities a hypothetical well was designed of 2000 m in depth. The reservoir pressure and temperature were 159 bar and 250 ˚C. The mass flow rate was 30 kg s^{-1}. Multiple simulations were performed for geothermal fluids with various NaCl (2.5 – 5 wt%) and CO_2 (0 – 3.4 wt%) mass fractions. Also, two gas lift mass flow rates (0.5 – 1.0 kg s^{-1}) were simulated for every case. The injected gas to accommodate gas lift was pure CO_2. Additionally, variations in injected gas mass flow rate (0 – 4.5 kg s^{-1}) and variations in geothermal fluid injection temperature (43 – 150 ˚C) were examined to optimize the binary cycle power plant. This was performed for a fluid containing 5 wt% NaCl and 1 wt% CO_2, because this composition shows the highest potential related to net power, utilization efficiency and CO_2 emission differences in favor of the binary cycle power plant system. This binary plant was compared to two single-flash power plant setups, when it comes to the non-condensable gas extraction system, one with a steam ejector/condenser and one with a centrifugal compressor.

The results of this hypothetical case show, for geothermal fluids with a CO_2 content > 0.5 wt% and a binary cycle injection temperature of 70 ˚C, the net power and utilization efficiency of a binary cycle power plant connected to a production well equipped with a gas lift system is higher compared to a single-flash power plant with a self-flowing well. Also, the mass fraction of CO_2 emitted per produced MWh is generally lower for the binary cycle system compared to the single-flash system. The optimized binary cycle power plant shows maximum performance (for the 5 wt% NaCl and 1 wt% CO_2 case) for a gas lift mass flow rate of 1.1 kg s^{-1} and an injection temperature of 43 ˚C. The net power, utilization efficiency and CO_2 emission is 3.0 MW, 47% and 306 kg MWh^{-1} for the binary system compared to 1.5 MW, 24% and 697 kg MWh^{-1} for the single-flash power plant with a steam ejector/condenser gas extraction system and compared to 2.1 MW, 32% and 505 kg MWh^{-1} for the single-flash power plant with a centrifugal compressor gas extraction system.

According to this study, it has been concluded that gas lift in geothermal wells is thermodynamic feasible and combined with a binary cycle power plant has high potential on thermodynamic and environmental grounds. The net power can be 1.5 – 2 times as high and the CO_2 emission can be 1.6 – 2.3 times as low compared to a basic single-flash power plant. Still, future research should be performed on the technical feasibility of gas lift and the comparison with other systems, e.g. other gas extraction systems for CO_2 removal in single-flash plants. Additionally, economic feasibility should be assessed and is highly recommended to complete the comparison with a single-flash power plant. It is also recommended to study scaling potential at the gas lift valve location in the production well and scaling potential due to low injection temperatures for the binary cycle. Finally, real base cases have to be executed. Because every geothermal system is unique and with optimization of the model parameters there is much to gain.

ACKNOWLEDGEMENT

First of all I would like to thank my supervisor Dr. Ir. C.A. Infante Ferreira of the Faculty of Mechanical, Maritime and Materials Engineering at Delft University of Technology. His constructive criticism and eye for detail during our regular meetings kept me sharp and focused. Advice on how to write and structure my master thesis was very valuable. Secondly, I must express my sincere gratitude to my supervisor J.H. Kleinlugtenbelt MSc of IF Technology at Arnhem. His guidance and comments has been an important component in the realization of this thesis. It was a privilege to perform scientific research within a commercial environment, but still having the freedom of defining my own path towards the goal of this study. Additionally, warm thanks to all my colleagues in recent months at IF Technology for making me feel welcome and part of the team or with helping to relax with a beer or a pingpong game every now and then. A special mention to Drs. Guus Willemsen for the possibility to do my thesis work at IF Technology. His enthusiasm and passion towards the subject was truly inspiring. A special thanks to my girlfriend, Annemarie, for her infinite encouragements and faith in me. Finally, I must express my very profound gratitude to my parents. This accomplishment would not have been possible without their unconditional support in the paths I have chosen up to now. Thank you.

F.W.J. Niewold
Arnhem, February 2017

CONTENTS

1

INTRODUCTION

1.1. Background and Motivation

With increasing oil depletion and the demand for energy production with reduced negative environmental impact worldwide, sustainable energy development is a popular theme. Binding targets for greenhouse gas (GHG) emissions for industrialized countries are an important incentive, while for developing countries the economic gain as a consequence of sustainable energy development stimulates the development. Renewable energy use is more labor-intensive, for each unit of electricity generated more jobs are created compared to electricity generated from fossil fuels. Renewable energy projects also keep money circulating in the local economy and countries become less dependent on oil and natural gas import (Gomberg, 2016). One of those renewable energy sources is geothermal energy, which has promising potential for generating heat and electricity for certain specific locations. Geothermal energy is energy generated in the core of the earth. Figure 1.1 presents a schematic of a geothermal power plant system. Hot (red) geothermal fluid is extracted from the reservoir by a production well. In the geothermal power plant energy is transferred from the geothermal fluid to generate electrical power. The cold (blue) geothermal fluid is reinjected into the reservoir via an injection well. In 2015 the total installed geothermal power plant capacity was 12.7 GW_e worldwide. The expected geothermal targets for 2050 are 70 GW_e, implying an exponential growth in the upcoming decades and which is an estimated 8.3% of the total world electricity production. This includes hydrothermal resources, enhanced geothermal systems (EGS) and other non-conventional resources (Bertani, 2016; Olasolo et al., 2016).

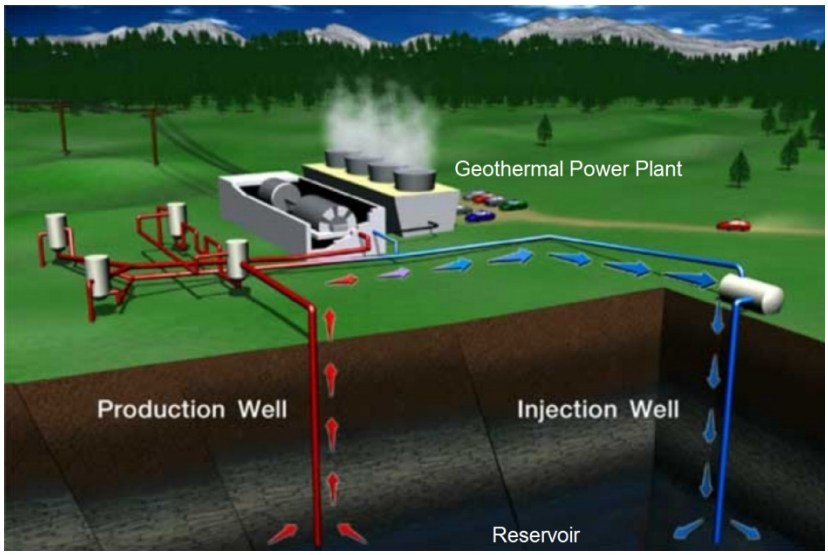


Figure 1.1: Geothermal power plant schematic. The hot (red) geothermal fluid flows from the reservoir via the production well to the geothermal power plant. In the power plant energy is transferred from the geothermal fluid to generate electrical power. The cold (blue) geothermal fluid is reinjected via the injection well into the reservoir.

Indonesia benefits from sustainable energy development, because of its large amount of geothermal sources. Indonesia is traversed by the world's ring of fire; across the country 117 active volcanoes are spread. Indonesia's geothermal electricity potential is estimated about 28 GW_e, which is 40% of world's geothermal energy potential in 2050. Currently, it utilizes only 4.5%, which is 1344 MW_e. Since recently, Indonesian government increases its installed geothermal power plant capacity. In 2025 it is targeted to have an installed capacity of 9500 MW_e (Nasruddin et al., 2016). In 2014 the Geothermal Capacity

Building Programme – Indonesia-Netherlands (GEOCAP) started, which is a collaboration between Indonesian and Dutch entities. The goal is to increase the capacity in developing, exploring and utilizing geothermal energy sources, and to assess and monitor its impact on the economy and environment (Hecker, 2016).

IF Technology, a leading geothermal consulting/engineering company in the Netherlands, is involved in GEOCAP. One of the objectives of this program is to study the technical and economic feasibility of artificial lift in geothermal wells to prevent flashing and using organic Rankine cycle (ORC) technology in the binary cycle geothermal power plant. The production well leads the geothermal fluid from the deep reservoir to the geothermal power plant. The geothermal fluid is in liquid or two-phase state. Solid particles, such as minerals or sand, can be entrained in the fluid flow. The reservoir contains the hot geothermal fluid. Flashing is the process of a saturated liquid undergoing a reduction in pressure resulting in partial evaporation. This is a phenomenon experienced in production wells and it is induced by pressure losses. Three positive effects as a result of preventing flashing are:

1. For flashing geothermal power plants to maximize power output additional flashing of the geothermal fluid is often necessary. This process reduces temperature and pressure of the stream. Consequently, it decreases the maximum efficiency of the power plant. Preventing flashing in the well by keeping the pressure above the boiling point increases the efficiency and maximizes power output (van der Hoorn et al., 2012).

2. Flash-steam geothermal power plants produce gaseous emissions from the non-condensable gases (NCG) that are dissolved in the geothermal fluid. Carbon dioxide (CO_2) is the most common NCG, furthermore gases such as methane (CH_4), hydrogen (H_2), hydrogen sulfide (H_2S) or ammonia (NH_3) can also be present. Currently, in every commercial plant all NCG, with the exception of the toxic H_2S, are removed from the condenser by some means and wasted to the atmosphere. Typical NCG concentrations range from 0.5-1.0 wt% of the total stream, from which approximately 95% consists of CO_2.. Although currently, there are no restrictions on the discharge of CO_2 for geothermal power plants, ideally gaseous emissions should be zero. With binary cycle power plants and artificial lift in the wells, NCG could stay in the solution and reinjected into the geothermal reservoir (DiPippo, 2012).

3. Flashing increases scaling potential in the well casings and the geothermal power plant significantly. The geothermal fluid is a solution of salts in water. Most minerals exhibit a higher solubility in water with increasing temperature. One exception is calcium carbonite ($CaCO_3$), which varies inversely with temperature. However, it is not only dependent on temperature, but also on partial pressure of CO_2, pH, salinity and calcium ion concentration. The deposition of $CaCO_3$ is often observed just above the flash horizon in the well casing. Since CO_2 is released during flashing, the pH of the liquid part increases significantly. This results in supersaturated geothermal fluid with respect to $CaCO_3$ and precipitation in the well casing. Silica (SiO_2) precipitation has greater probability in the flash vessel, piping, injection wells and formation (reservoir), because the SiO_2 concentration increases during flashing and the temperature of the fluid decreases. The injection well is used for enhancing reservoir pressure and geothermal fluid recirculation. SiO_2 precipitation can affect functionality of plant equipment or even decrease the permeability of the reservoir. With artificial lift in wells scaling potential can be significantly reduced, because flashing is prevented in the well. Also SiO_2 concentration in the geothermal fluid stays constant, which decreases the potential for precipitation (DiPippo, 2012).

1.2. Organic Rankine Cycle Technology

The standard classical geothermal power plant classification comprises binary cycle, single-flash, double-flash, dry-steam and back pressure power plants. Figure 1.2 presents two pie charts with the number of units and the installed capacity for the classical power plant types. In 2014, the number of binary cycle power plants in operation was almost half (46%) of the total number of units in operation worldwide. The installed capacity on the other hand is only 14% of the total installed capacity worldwide. It has been shown that the average power rating per unit for binary cycle power plants is relatively small, approximately 6.3 MW_e/unit (Bertani, 2016).

The installed capacity of binary cycle power plants in Indonesia was only 8 MW_e in 2014, against 460 MW_e of dry-steam plants and 873 MW_e of single-flash plants (Bertani, 2016). The large amount of high enthalpy/high temperature geothermal wells present in Indonesia, which produce dry steam or a combination of liquid and steam, causes this distribution. Binary cycle power plants with ORC technology

are mainly used for low-temperature resources, where it is unlikely that wells will flow spontaneously. For geothermal fluid temperatures below 150°C, it becomes difficult to operate a flash plant efficiently and economically (DiPippo, 2012). Since there are currently no restrictions on CO_2 emissions for geothermal power plants, flash plants for high enthalpy/high temperature sources are attractive for its relatively high maturity of the technology, low investment costs, high safety and low complexity (van der Hoorn et al., 2012).

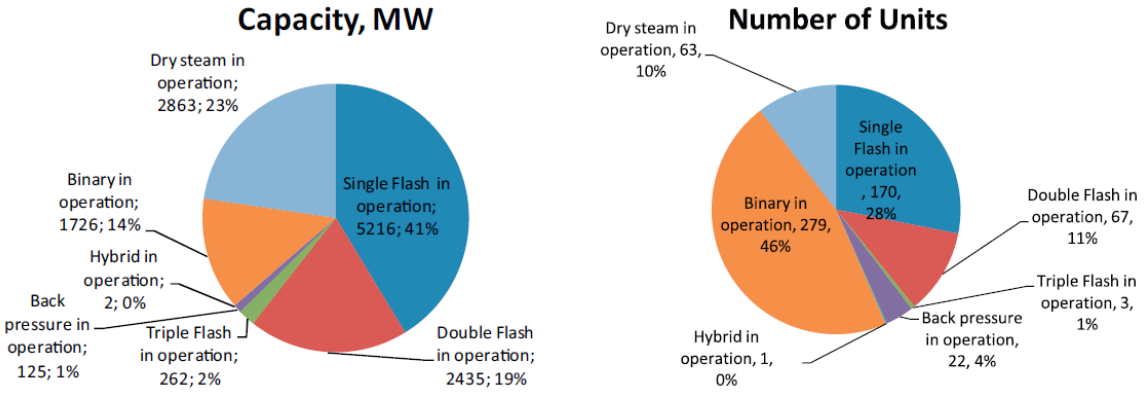


Figure 1.2: Number of units and installed capacity in MW_e for each typology worldwide (Bertani, 2016).

For high temperature geothermal fields, mixed-steam binary plants are powered with steam and liquid, but these plants are scarce. In 2010, Te Huka geothermal power station in New-Zealand was opened consisting of one unit and with an installed capacity of 24 MW_e. The wellhead temperature is 250 ˚C and the mass flow rate of the geothermal two-phase fluid is approximately 210 kg/s. Together with the Ribeira Grande geothermal power plant in Portugal (The Azores), these are the only known binary cycle power plants in literature with an outlet temperature at the wellhead above 250 ˚C (Zarrouk and Moon, 2014).

1.3. Artificial Lift in Geothermal Wells

It is common practice in binary cycle power plants working with low temperature geothermal fluids to install a downhole pump in the production well that pressurizes the fluid below the flash depth to prevent $CaCO_3$ scaling. The flash depth represents the front where boiling starts. It depends among others on reservoir properties and mass flow rate. Downhole pumps are also used in non-spontaneously flowing wells, which have often low temperature geothermal fluid and insufficient reservoir pressure to stimulate the fluid production. Consequently, pressurizing the well increases the mass flow rate (DiPippo, 2012).
In the petroleum industry, it is common practice to stimulate oil production by means of downhole pumps or gas lifting techniques (Renpu, 2011).

1.4. Research Objectives

With the everlasting demand for more sustainable energy production and the high potential of binary cycle geothermal power plants, the objective of this study is to analyze the possibility of artificially lifting high temperature (< 250 ˚C) geothermal wells in order to prevent flashing of the geothermal fluid in the well. A technology that has not been used in commercial geothermal power plants yet for temperatures > 200 ˚C and which has not been explored according to available literature. This involves numerical modeling of mass, heat and momentum transfer in the production well. In addition to this objective, a binary cycle geothermal power plant is modeled and the thermodynamic performance is computed. It is aimed for to couple the numerical model of the production well with mathematical models of the injection well, the reservoir model and the geothermal power plant. The total mathematical model is able to calculate and optimize the thermodynamic performance of the power plant for different reservoir conditions, geothermal fluid properties, well dimensions and atmospheric conditions. A standard technology geothermal power plant (single-flash power plant) is incorporated in the model as well in order to compare the thermodynamic performance. The production well connected to the single-flash power plant is a self-flowing well, which means it flows spontaneously without any form of artificial lift.

Hence in this thesis, an attempt is made to answer the following main research question:

What is the technical and thermodynamic feasibility of artificial lift in geothermal wells connected to binary cycle power plants compared to single-flash power plants connected to self-flowing flashing geothermal wells?

1.5. Thesis Outline

In Chapter 2, a comprehensive literature survey is performed. The state of the art of standard geothermal power plants is reviewed. A review of existing literature within geothermal industry on the topic of lifting fluids from large depths in combination with binary cycle geothermal power plant technology is performed. Additionally, literature within the petroleum industry on lifting fluids from large depths is looked into. Available correlations and/or models describing the thermodynamic and transport properties of geothermal fluids are sought. Finally, the fundamentals of reservoir flow, well flow and geothermal power plant thermodynamics are discussed.

Chapter 3 describes the development and implementation of the mathematical model in MATLAB. It includes the modeling approach, sub models, boundary conditions, relevant assumptions and phenomena, conservation laws and constitutive equations. Then in Chapter 4, the validation of the mathematical model with field data obtained from literature is treated. Additionally, a sensitivity analysis on the model input parameters is presented.

Chapter 5 proposes a hypothetical case. Simulations of the binary cycle power plant system and the single-flash power plant system are involved. The results of both power plants are compared and discussed. Finally, in Chapter 6 conclusions are drawn and recommendations are proposed for future research.

2

DESCRIPTION OF SYSTEMS & THEORETICAL BACKGROUND

In the present chapter, Section 2.1 discusses the history and thermodynamic cycles of the most common geothermal power plants in Indonesia and worldwide. Section 2.2 provides an overview of lifting techniques for fluids from large depths for both geothermal applications and petroleum applications. In Section 2.3, relevant literature related to thermodynamic and transport properties of geothermal fluids are presented. The theory behind flow characteristics and relevant phenomena in the reservoir and geothermal wells is explained in Section 2.4. Also, thermodynamics of the relevant geothermal power plants are discussed.

2.1. Types of Geothermal Power Plants

The first geothermal power plant built in Indonesia was a pilot project in Kamojang in 1978 with an installed capacity of 0.25 MW_e (DiPippo, 2012). In 2014, the total installed capacity of geothermal power plants for electricity production was 1340 MW_e, divided over ten plants and locations. Only 8 MW_e was generated by one binary cycle geothermal power plant. Furthermore, multiple dry-steam power plants generated 460 MW_e and multiple single-flash power plants generated the remaining 873 MW_e (Bertani, 2016).

Until recently, the type of geothermal power plant corresponded to the type of geothermal system. The type of geothermal system can be classified into five categories based on the thermodynamic state of the fluid in the geothermal reservoir (Table 2.1). The thermodynamic state mainly depends on temperature, pressure and composition of the fluid. Additionally, fluid flow through the reservoir and permeability of the reservoir affects the thermodynamic state of the system, because it relates to the accompanying pressure drop in the system.

Figure 2.1 presents the geothermal power plant operating enthalpy range based on published data from 89 geothermal power plants (Zarrouk and Moon, 2014). It shows that binary cycle plants are utilized mainly at sites with hot-water and low-enthalpy two-phase liquid dominated reservoirs. On the other hand, single-flash and dry-steam plants are generally built at sites with high-enthalpy liquid-dominated and vapor-dominated systems. Besides these four standard geothermal power plants, there are also advanced geothermal energy conversion systems in operation. Hybrid flash-binary geothermal power plants exploit both the steam and the remaining liquid of the geothermal fluid to increase power output and efficiency. In the remaining subsections of Section 2.1, the operation, schematic and main equipment of the basic single-flash power plant and the basic binary cycle power plant are presented. The operation, schematic and main equipment of the double-flash power plant, dry-steam power plant and hybrid flash-binary power plant can be found in Section A.1.

Table 2.1: Types of geothermal systems based on thermodynamic state of the reservoir (Rivera Diaz et al., 2016).

Category		Temperature (T)	Production enthalpy (h)
Hot-water		$T < 220$ °C	$h < 943$ kJ/kg
Two-phase, liquid dominated	Low-enthalpy	220 °C $< T <$ 250 °C	943 kJ/kg $< h <$ 1100 kJ/kg
	Medium-enthalpy	250 °C $< T <$ 300 °C	1100 kJ/kg $< h <$ 1500 kJ/kg
	High-enthalpy	250 °C $< T <$ 330 °C	1500 kJ/kg $< h <$ 2600 kJ/kg
Two-phase, vapor-dominated		250 °C $< T <$ 330 °C	2600 kJ/kg $< h <$ 2800 kJ/kg

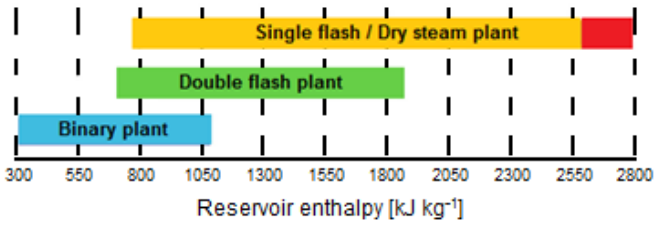


Figure 2.1: Geothermal power plant operating enthalpy range based on published data from 89 geothermal power plants (6 dry-steam, 34 single-flash, 18 double-flash and 31 binary cycle) (Zarrouk and Moon, 2014). Red presents the operating range of the dry-steam power plants.

2.1.1. Single-Flash Steam Power Plant

The largest share of power production worldwide with 41% of the total generated power is attributed to single-flash power plants (Bertani, 2016). It is often the first power plant built at a new site with a liquid-dominated system. This is due to its relatively simple cycle, much operational experience and relatively low investment costs. Single-flash indicates that the geothermal fluid undergoes a process of partial phase change, which means a transition from pressurized liquid to a liquid-vapor mixture. The transitioning is induced by a pressure drop below the saturation pressure for the corresponding temperature. The flash process itself can generally take place at three different locations (DiPippo, 2012):

1. Reservoir: the geothermal fluid flows to the bottom and inlet of the production well through the permeable formation with an accompanying pressure drop.
2. Production well: the pressure decreases due to frictional, hydrostatic and accelerational pressure losses.
3. Power plant: a throttling valve produces steam by decreasing the pressure.

Figure 2.2 presents a simplified single-flash power plant schematic (DiPippo, 2012). The liquid geothermal fluid or liquid-vapor mixture is typically controlled and monitored by a silencer (S), valves (WV) and pressure/temperature gauges once it leaves the production well (PW) and before it reaches the cyclone separator (CS). At the inlet of the cyclone separator, the liquid or liquid-vapor mixture is throttled to an optimum pressure by a throttling valve. In the cyclone separator the steam is separated from the liquid. The steam travels through a moisture remover (MR) before it is supplied to the steam turbine, in order to reduce scaling and erosion potential in the piping and turbine components. After the turbine, the low-pressure steam is condensed with cooling water from the cooling tower (CT) in the condenser (C). Finally, the condensed steam in the cooling tower can be added to the remaining liquid geothermal fluid that is separated from the steam in the cyclone separator or injected separately. The steam ejector/condenser (SE/C) extracts non-condensable gases (NCG) present in the geothermal fluid from the condenser.

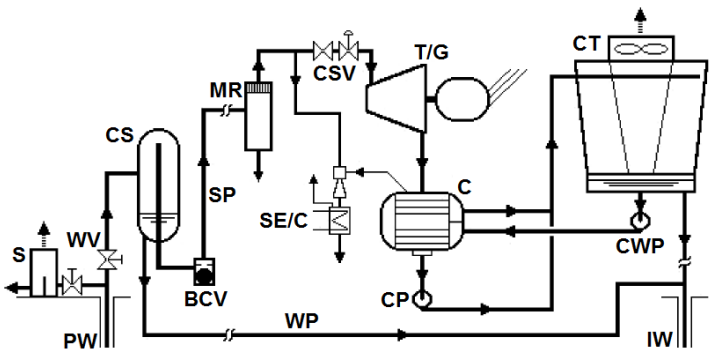


Figure 2.2: Simplified single-flash power plant schematic (DiPippo, 2012).

The liquid is reinjected in the reservoir mainly for two purposes (Rivera Diaz et al., 2016):

1. Recirculation of geothermal fluid improves the resource recovery by keeping the water level content in the reservoir sufficient. Additionally, the pressure in the reservoir is boosted to compensate the pressure drop in the permeable formation. Consequently, the geothermal fluid production is maintained or even increased compared to geothermal power plants without reinjection.

2. Waste water disposal is limited to the reservoir, where the geothermal fluid originally came from, instead of wasting it to the environment. The geothermal fluid contains a fairly amount of minerals and NCG.

Negative aspects of the single-flash steam power plant are the risk of scaling in the production well, cyclone separator and the moisture remover and the relatively low efficiency. Typical utilization efficiencies are in the range of 30 – 35% (DiPippo, 2012).

2.1.2. Binary Cycle Power Plant

The binary cycle power plant is characterized by a working fluid undergoing a closed cycle. At the birth of geothermal power plants, binary cycles were utilized for steam fields, because the steam was too contaminated with minerals and dissolved gases. In that case, clean water was used as working fluid. Currently, binary cycle power plants are mainly employed at hot water or liquid-dominated low-enthalpy sources. Organic fluids are chosen as working fluid due to their favorable thermodynamic properties at lower temperatures. Typical utilization efficiencies are in the range of 25 – 45% (DiPippo, 2012).

Positive aspects of binary cycle power plants are its broad operational experience (Figure 1.2), low maintenance cost, reliability and high availability. As long as there is geothermal fluid in liquid phase, binary cycles are suitable for operation. As the geothermal fluid remains in the liquid phase, higher efficiencies can be achieved compared to flash plants. On the negative side, the investment costs are generally higher than for single-flash power plants. The power cycle needs additional precautionary measures if the working fluid is toxic or flammable (Van der Hoorn et al, 2012).

Figure 2.3 presents a simplified schematic of a basic binary cycle geothermal power plant (DiPippo, 2012). The binary cycle power plant can basically be divided into three subsystems: the power conversion cycle, the geothermal fluid cycle and the cooling system for the removal of heat. Binary cycle plants are often utilized at sites with non-spontaneous flowing wells. Therefore, a downhole pump (P) is mounted in the production well. The geothermal fluid flows from the production well to the sand remover (SR) to prevent scouring and erosion of the piping and heat exchanger tubes. Heat is transferred to the working fluid in a preheater (PH) and evaporator (E) typically, before the geothermal fluid is injected with an injection pump (IP) in the injection well (IW). The geothermal fluid is kept above the boiling pressure to prevent $CaCO_3$ scaling and above the temperature at which SiO_2 scaling becomes an issue. The working fluid undergoes a closed cycle, in which it receives heat from the geothermal fluid, evaporates, expands in the turbine (T) and condensates before it returns to the preheater by means of a condensate pump (CP). During condensation of the working fluid heat is transferred to the cooling system. The turbine drives a generator (G) to generate electricity.

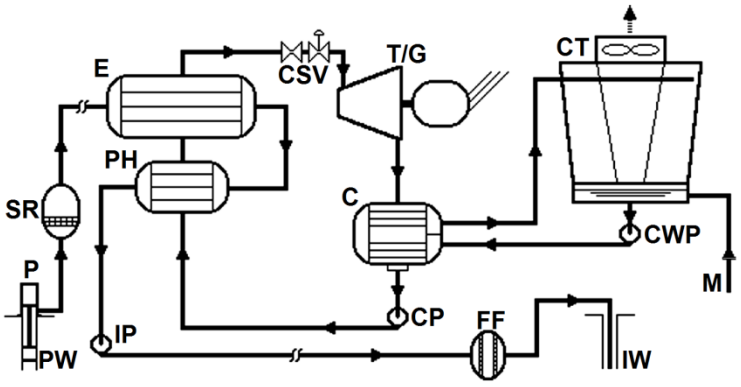


Figure 2.3: Simplified schematic of a basic binary geothermal power plant (DiPippo, 2012).

2.1.3. Summary and Conclusion

The power plant type data is summarized in Table 2.2. It shows the number of binary cycle units is largest, but the total capacity is only ranked 4th due to its low average capacity. This in turn is caused by the low-temperature fields where binary cycle power plants are mostly deployed. The objective of the present work

is to examine if binary cycle power plants can be deployed for high temperature reservoirs by artificially lifting the geothermal fluid in the production well.

Table 2.2: Summary of power plant characteristics according to published data (Ref. a. Bertani, 2016; Ref. b. Zarrouk and Moon, 2014; Ref. c. DiPippo, 2012).

Power plant type	Number of units	Total capacity, MW_e	Average capacity, MW_e/unit	Enthalpy range, kJ/kg
Single-flash	170 (28%)[a]	5216 (41%)[a]	30.4[a]	780-2783[b]
Double-flash	67 (11%)[a]	2435 (19%)[a]	37.4[a]	697-1910[b]
Dry-steam	63 (10%)[a]	2863 (23%)[a]	45.4[a]	2650-2797[b]
Binary cycle	279 (46%)[a]	1762 (14%)[a]	6.3[a]	306-1100[b]
Hybrid flash-binary	47[c]	368.6[c]	7.8[c]	306-2789[b]

Currently, in Indonesia no commercial geothermal binary plant is in operation. However, a demonstration plant will be installed at the Lahendong geothermal field (Frick et al., 2015). Furthermore, there have been situated only single-flash and dry-steam power plants. It is practically impossible to replace dry-steam power plants by a full-binary power plant, because the steam is already present in the geothermal reservoir. Therefore, the present work focusses on the comparison of a binary cycle power plant and a single-flash power plant.

2.2. Artificial Lift in Wells

The current section discusses artificial lift for geothermal applications. Additionally, relevant artificial lifting techniques from the petroleum industry are presented. In the oil and gas industry, downhole pumps are a proven technology for the pumping of fluids. However, geothermal applications differ from these established applications due to the higher temperatures and larger volumetric flow rates (Frick et al., 2011). Finally, on the basis of certain selection criteria the most suitable lifting technique is selected.

2.2.1. Theory of Pressurizing the Production Well

Downhole pumps for geothermal application could serve a number of purposes, three relevant purposes are:

1. The pumping of geothermal fluid from non-spontaneously flowing production wells, which is relevant for hot-water systems.
2. The pressurizing of the fluid in the production well to prevent flashing and the accompanied non-favorable phenomena, e.g. scaling and/or release of NCG. This is relevant for liquid-dominated systems, where flashing occurs in the production well.
3. Stimulation of the fluid flow and increasing the production of geothermal fluid. Consequently, the power output of the plant can be increased.

The basic idea is to install a pump below the dynamic fluid level (Figure 2.4). On the right, the pressure of the geothermal fluid is increased by the downhole pump. The pressure decline is mainly caused by the pressure loss due to friction and gravitation. The objective is to keep the pressure at least above the saturated liquid pressure at the wellhead. The static fluid level indicates the head for a non-spontaneously producing well. The dynamic fluid level differs from the static fluid level due to the additional frictional pressure losses in the reservoir and production well owing to the flowing geothermal fluid during production. Besides the drawdown of the fluid level during operation, the installation depth also depends on the necessary intake pressure to avoid cavitation and the release of NGC (Frick et al., 2011).

2.2.2. Artificial Lift in Geothermal Wells

Downhole pumps for geothermal applications are distinguished by the mode of power transmission. Currently, it is done by the use of line shaft pumps (LSP) or electrical submersible pumps (ESP) (Xie et al., 2005; SANDIA, 2008; Frick et al., 2011; DiPippo, 2012). While a hydraulic turbine pump (HTP) could be a valuable candidate for these particular environments (Harrison et al., 1990; EGEC, 2012).

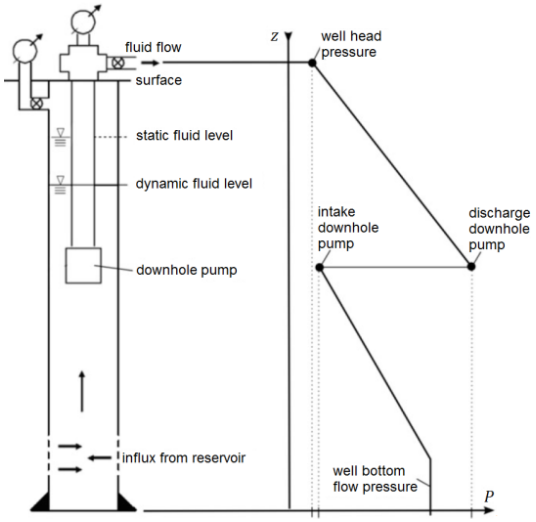


Figure 2.4: Schematic representation of a downhole pump in a production well (left) and the pressure curve as a function of depth (right) (Frick et al., 2011).

2.2.2.1. Line Shaft Pump

Line shaft pumps are powered by an electrical motor above ground and driven by a straight shaft down the production well (Figure 2.5). The shaft is equipped with vanes or impellers, which are mounted inside the well. The geothermal fluid is carried by the tubing that surrounds the shaft (Harrison et al., 1990). The major drawback of this system is the installation depth, which is limited to vertical wells (Frick et al., 2011). Additional cons of this type of pump are the delicate handling during installation and removal, coating materials for enclosing tubing, make-up lubricating fluid for bearings and relatively large production well casing diameter. Pros are attributed to the absence of electric parts in the well, high efficiency, long lifetime, attractive cost and withstanding relatively high temperatures (< 200 ˚C) (EGEC, 2012).

2.2.2.2. Electrical Submersible Pump

Figure 2.6 shows the schematic of an electrical submersible pump (ESP). An ESP is driven by an electrical motor installed in the production well. The motor is powered by an electric cable connected to the grid above the surface. The seal section protects the shaft and rotating parts from the geothermal fluid. The turbine pump is mounted inside the tubing that carries the geothermal fluid to the surface (Harrison et al., 1990). The major drawback of this system is the problem of cooling the motor, which must be done with the hot geothermal fluid (Frick et al., 2011). More disadvantages are related to relatively high cost, electric insulation shortcomings and lower efficiencies in practice. The advantages are large installation depths, long lifetime, high flow rates in limited casings, solution gas handling and much operational experience (EGEC, 2012). Flowserve (2011), one of the world's largest manufacturers of pumps, builds ESP's that can withstand 160 ˚C. Whereas EGEC (2012) published that the maximum operating temperature is approximately 180-200 ˚C.

2.2.2.3. Hydraulic Turbine Pump

A hydraulic turbine pump (HTP) system is presented in Figure 2.7. The pump is driven by a turbine, which is also installed in the production well. Above the ground, part of the geothermal fluid is recirculated and filtered, where after the geothermal fluid is sent through a turbine by using a booster pump. The turbine, which is powered by the high pressure fluid, drives the downhole pump (Harrison et al., 1990). For geothermal applications, there is no literature reporting the use of HTP's. Although there are companies that offer HTP's for geothermal application currently. HTP's gain in interest, because of the ability to operate at high temperature (> 200 ˚C) and high salinities. Additionally, there are no electric parts in the production well and long lifetimes are ascribed. On the other hand, an HTP has relatively low efficiency, it is bounded to the vertical section of the production well, it needs large diameter wells, there is limited operational experience and the costs are high. Furthermore, packer anchoring problems are reported (EGEC, 2012).

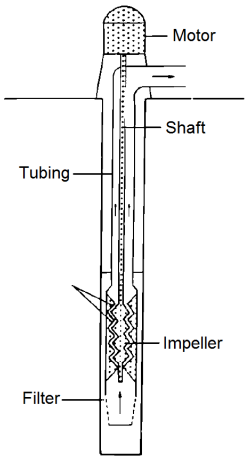


Figure 2.5: Line shaft pump (LSP) (Harrison et al., 1990).

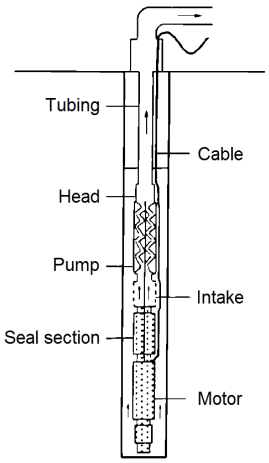


Figure 2.6: Electrical submersible pump (ESP) (Harrison et al., 1990).

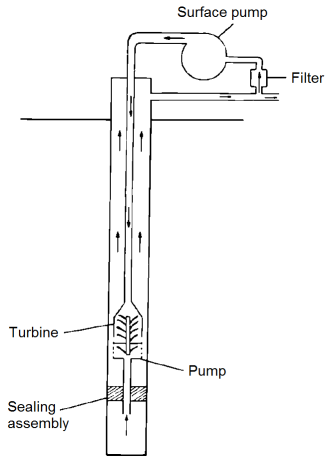


Figure 2.7: Hydraulic turbine pump (HTP) (Harrison et al., 1990).

2.2.3. Artificial Lift in Petroleum Wells

In the oil and gas industry artificial lift is deployed when the pressure in the reservoir is not sufficient to produce at its most economical rate. Basically, five artificial lifting techniques can be distinguished: 1. plunger lift and sucker rod pump (SRP), 2. hydraulic pump, 3. electrical submersible pump (ESP), 4. progressing cavity pump (PCP) and 5. gas lift (Cholet, 2008). These methods are discussed below, except the ESP, which is identical to the geothermal ESP's.

2.2.3.1. Plunger Lift and Sucker Rod Pumps

There are many different types of beam pumping systems. Figure 2.8 gives the schematic of a basic plunger lift and sucker rod pump (SRP). The prime mover is the motor of the system. It provides power to the system by transferring rotating motion to the surface pumping equipment. This equipment converts rotation into an oscillating linear motion. The sucker rod is attached to the plunger in the production well. Fluids are lifted up the tubing by the reciprocation strokes of the plunger. SRP's are simple to operate and maintain. However, capital costs are high and it is not suited for deep deviated wells, because of the sucker rod string (Baldwin et al., 2000; Cholet, 2008).

2.2.3.2. Hydraulic Pump

In the oil and gas industry, there are generally three types of hydraulic pumps deployed: the piston type (Figure 2.9), the jet pump (Figure 2.10) and the turbine pump. The principle of the latter one has already been discussed in Section 2.2.2.3. The hydraulic piston pump is installed below the fluid level. High pressure power fluid (oil or water) is forced through the engine causing it to reciprocate. The engine drives the pump, which pumps the mixture of spent power fluid and well production fluid to the surface. With the jet pump the power fluid enters the pump from the top. The total pressure is converted almost completely in dynamic pressure in the nozzle. After the nozzle the power fluid mixes with the production fluid and passes momentum and energy to the production fluid. In the diffuser the velocity head is converted to static pressure head to lift the fluid to the surface. Advantages of these systems are the ability of working at deep depths, in deviated wells; it can handle heavy viscous fluids. Disadvantage are related to fire hazards if oil is used as a power fluid and the difficulty of handling fluids with high solid content or gas content (Baldwin et al., 2000; Cholet, 2008).

2.2.3.3. Progressing Cavity Pump

Figure 2.11 shows a progressing cavity pump (PCP). It consists of a rotor which rotates in an elastomeric stator to let cavities, filled with the production fluid, move upward. The pump is connected to an engine above the surface by a rotating sucker rod. These PCP's cannot be handled in deviated wells and the stator is sensitive to high temperatures. On the other hand, PCP's can handle crude oils excellently (Cholet, 2008).

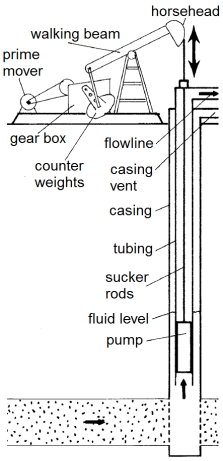


Figure 2.8: Sucker rod pump (SRP) (Conaway et al., 2000).

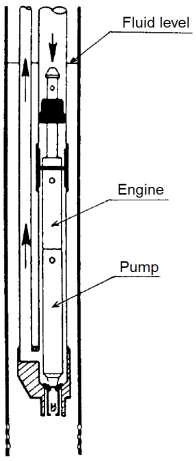


Figure 2.9: Hydraulic pump piston type (Cholet, 2008).

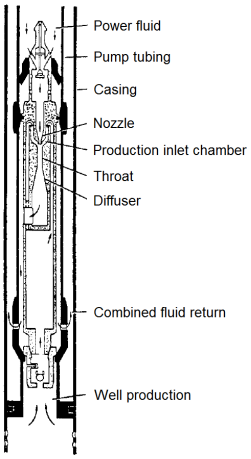


Figure 2.10: Jet pump (Cholet, 2008).

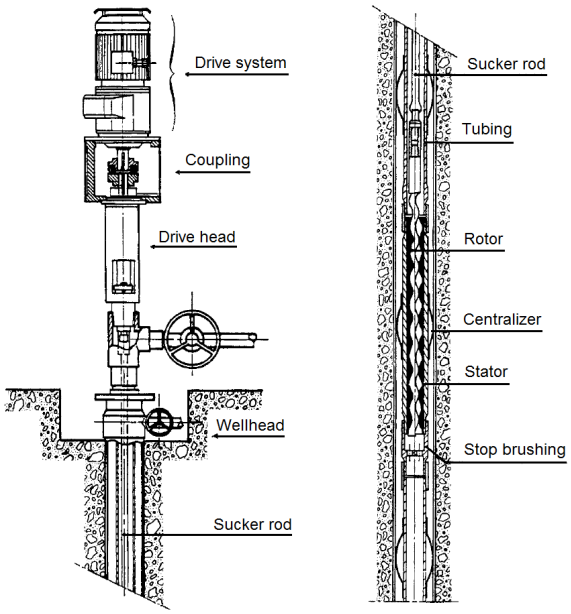


Figure 2.11: Typical configuration of a progressing cavity pump (Cholet, 2008).

2.2.3.4. Gas Lift

Gas lift systems are an alternative to lifting techniques with pumps and can be divided into two types of injection: continuously or intermittently. Only continuous injection is discussed, since 95% of the production wells use continuous gas injection and the volume flow rate in continuous injection is much higher, which is necessary in geothermal applications. Figure 2.12 presents a schematic of the working principle of a gas lift system. Instead of increasing the pressure with a pump, the pressure loss rate is decreased from a certain level. This is done by the injection of a gas, e.g. co-produced natural gas, down the casing annulus into the tubing string at a certain pressure, flow rate and depth in the production well.

The density of the fluid decreases, which decreases the hydrostatic pressure loss above the injection point, causing the production well to flow. A negative aspect of this system is the requirement of a compressor at the surface to compress the gas, which is an inefficient process in comparison to pumps. On the other hand, besides the compressor other large size equipment is not necessary making it a suitable application for offshore industry. It works also well in sand-producing wells, which can cause significant erosion to pump type systems. Additionally, very deep deviated wells are generally equipped with gas lift for its flexibility and low operating costs (Baldwin et al., 2000; Cholet, 2008).

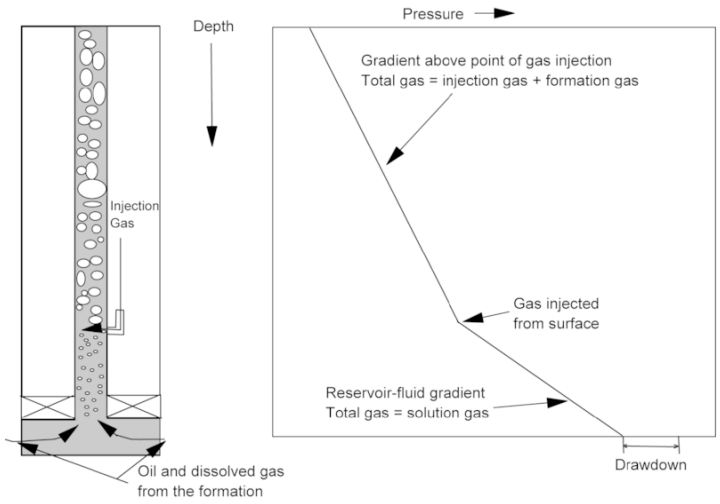


Figure 2.12: Schematic of gas lift principle (left) and pressure vs. depth (right)

2.2.4. Overview and Selection

Table 2.3 and Table 2.4 present an overview of characteristics of different lifting techniques for geothermal applications and oil & gas applications, respectively. The properties are ranked according to importance. According to temperature, it looks like none of the pumps for geothermal applications is suitable to operate at temperatures in the range of 200 – 250 ˚C. In oil & gas industry SRP, hydraulic piston, hydraulic jet and gas lift systems have good opportunities for operating at temperatures up to 250 ˚C. From these four lifting techniques, only gas lift shows high volumetric flow rates, which is an important feature for geothermal applications in order to make the geothermal power plant economically feasible.

Table 2.3: Comparison of different lifting techniques from published data in literature and by manufacturers for geothermal applications. (Ref. 1. EGEC, 2012; Ref. 2. Flowserve, 2011; Ref. 3. Lienau et al., 1991; Ref. 4. Clyde Pumps Ltd., 2008; Ref. 5. Harrison et al., 1990; Ref. 6. Frick et al., 2011).

	LSP	ESP	HTP	Ref.
Operating temperature, ˚C	120-204	< 200	< 218	1, 2, 3
Flow rates, l/s	138	70-250	8-166	1, 2, 4
Head, m	700	750	300-1500	2, 4
Installation depth, m	350-600	1000-3600	1500-3000	2, 3, 5
Efficiency, %	50-65	50-65	40	1, 3
Costs	+	+/−	+/−	1, 3
Lifetime and maintenance	+	+/−	+	1, 3, 6
Maturity of technology	+ +	+ +	− −	1, 5

Table 2.4: Comparison of different lifting techniques for oil and gas applications. (Ref 1. Baldwin, 2000; Ref 2. Cholet, 2008; Ref. 3. New Mexico Tech, 2005; Ref. 4. Clegg et al., 1993).

	SRP	ESP	Hydraulic Piston	Hydraulic Jet	PCP	Gas lift	Ref.
Operating temperature, ˚C	< 288	< 205	< 260	< 260	< 120	No limit	2, 3
Flow rates, l/s	1-11	74-93	7-9	< 28	8-9	55-93	2, 3
Installation depth, m	< 4800	< 4500	< 5200	< 4500	< 1800	< 4500	3
Efficiency, %	30-60	35-60	30-55	10-30	40-80	10-32	2, 3
Costs	+ +	−	+/−	+/−	+ +	+	4
Lifetime and maintenance	+ +	+/−	+	+	+ +	+ +	4
Maturity of technology	+ +	+	+	+	+	+	4

There is not one lifting technique that can be selected unanimously. It depends on the characteristics of the reservoir (e.g. porosity, depth) and the properties of the geothermal fluid (e.g. temperature, pressure, composition). From Table 2.3 and Table 2.4, only gas lift matches the requirements of high temperature and high flow rates. For lower temperatures, ESP's are the better solution compared to LSP and HTP for its high installation depth, high maturity and low costs. Therefore, in the continuation of this study gas lift is modeled for geothermal fluid temperatures in the range of 200 – 250 ˚C.

2.3. Geothermal Fluid Properties

In this section literature on geothermal fluid properties are discussed. The relevant properties to be examined are: saturation pressure, density, viscosity, enthalpy, entropy, heat capacity, thermal conductivity and solubility. It has been aimed for to create a geothermal fluid property (GFP) model for both liquid and two-phase state for temperatures up to 250 ˚C, pressures up to 1000 bar and salinities up to 350 g l^{-1}. The published GFP models over the years have not been unambiguous, which makes it more difficult to model the system accurately (Adams and Bachu, 2002; Duan and Sun, 2003; Champel, 2006; Francke and Thorade, 2010).

2.3.1. Introduction

For simplicity, water properties are often used for the flow characteristics in reservoirs and wells, for the evaluation of artificial lift methods in geothermal production wells and for power plant performance (Xie et al., 2005; IF Technology, 2012). However, in order to model the behavior of the production well, injection well, reservoir and power plant more accurately, implementation of the thermodynamic and transport properties of geothermal fluids in the model is necessary. Relevant properties of geothermal fluids, like density and viscosity, are controlled by pressure, temperature and composition. Temperature and pressure can vary from atmospheric conditions to temperatures and pressures > 300 ˚C and > 100 MPa, respectively. The composition varies depending on the type and amount of dissolved solids and NCG. The total dissolved solids (TDS), often referred as the salinity, can reach in excess 350 g l^{-1} (Adams and Bachu, 2002). The TDS decreases the boiling point of the geothermal fluid, whereas the dissolved NCG increases the boiling point. The determination of the boiling point is crucial for e.g. determining pump setting depths or gas lift valves in the production well (Aksoy, 2007). The properties of geothermal fluids containing dissolved solids can vary by more than 25% for density and by one order of magnitude for viscosity in comparison to fresh water properties. Therefore, neglecting the effect of dissolved components on the fluid properties introduces significant errors on flow behavior in the injection well, production well and reservoir (Adams and Bachu, 2002).

2.3.2. Chemical Composition

Glassley (2014) reviewed the chemical composition of various geothermal systems, mostly volcanic areas, in New Zealand, Mexico, Philippines, Iceland and the USA. The chemical composition of geothermal fluids consists of the dissolved solids Na, K, Ca, Mg, Cl, B, SO_4, HCO_3 and SIO_2, and (dissolved) gases CO_2, H_2S, CH_4, H_2 and NH_3. All systems show > 80 wt% Na and Cl TDS, and > 80 wt% CO_2 of total gases.

The most common ions in low to moderate saline geothermal fluids are Cl^- and Na^+; therefore, geothermal fluids are frequently modeled as an aqueous sodium chloride (NaCl) solution. Adams and Bachu (2002) reviewed several published algorithms to calculate density and viscosity as a function of temperature, pressure and salinity. In these publications the differences between the density reached up to 20% and for the viscosity the maximum difference was even 50%, which indicates a significant discrepancy between the available algorithms. Francke and Thorade (2010) studied the sensitivity of the volumetric flow rate of a downhole pump in a geothermal production well for different density and viscosity algorithms, with a maximum deviation of 3% and 2.5%, respectively. The different density algorithms caused a deviation in pressure heads from the average pressure head at the pump of 7.3% in steady state operation, which consequently caused a deviation in the volumetric flow rate of 14.5%. During start-up conditions the deviation for volumetric flow rate was even 52% at its maximum. However, the influence of the viscosity function was negligible. It was concluded that viscosity related frictional pressure loss was small compared to density related gravitational pressure loss. Champel (2006) studied the influence of geothermal fluid density from five different functions on pumping requirements. The maximum deviation was more than 20%. It was stated that the discrepancy between density functions leads to inaccuracy in the buoyancy

effect, which could result in under- or over-dimensioning of the downhole pump. It has been recommended to conduct new measurements on geothermal fluid densities, particularly for temperatures up to 250 ˚C, pressures up to 50 MPa and molalities in the range of 1 – 5 mol kg^{-1}. Figure 2.13 (left) shows the depth distribution of the main elements Na, Cl and Ca found in sedimentary basins and (right) TDS found in sedimentary basins and crystalline rocks across the world. In general, the salinity increases with depth and can vary from a few g l^{-1} to 200 g l^{-1} in most geothermal systems, with extremes to 643 g l^{-1} (Huenges and Ledru, 2010). Mahon et al. (2000) reviewed the chemistry of geothermal fluids in Indonesia. It was found that the highest published TDS has been found at Wayang Windu, with approximately 40 g kg^{-1}, which is equivalent to a molality less than 0.75 mol kg^{-1}.

Concerning the dissolved gases, CO_2 is the most abundant gas encountered in geothermal systems. In Indonesian geothermal systems 95 – 98 wt% of gases constitutes of CO_2, 2 – 3 wt% of H_2S and other gas constituents are even less abundant (Mahon et al., 2010; Yuniarto et al., 2015). Hosgor et al. (2015) studied the effects of dissolved CO_2 on reservoir production performance. In liquid dominated reservoirs mass fractions of CO_2 can reach up to 5 wt%. Khalifa and Michaelides (1978) studied the effect of NCG on the power plant performance, where NCG have been replaced by CO_2 equivalent mass fraction, because NCG consisted of ~80 wt% CO_2. According to Gokcen and Yildirim (2008), who studied power plant performance affected by CO_2 presence as well, CO_2 mass fractions encountered are even 25 wt%. Figure 2.14 shows the pressure-enthalpy diagram of a binary H_2O – CO_2 system having a mass fraction $w_{CO2} = 0.015$. It is clearly visible that the bubble point pressure shifts upwards compared to pure water systems. Once degassing starts in the production well, initially the gas phase contains mainly CO_2. While pressure declines further, initially the fluid behaves almost isothermally. Once most of the CO_2 is released from the liquid phase, H_2O starts dominating the gas phase. Then geothermal fluid starts behaving like pure water, because almost all CO_2 is present in the gas phase. This phenomenon can be seen by the isobar and isotherm coinciding more or less near gas saturation (Hosgor et al., 2015).

Following the findings in the present section, it is assumed in this work that H_2O, NaCl and CO_2 are the only components present in the geothermal fluid.

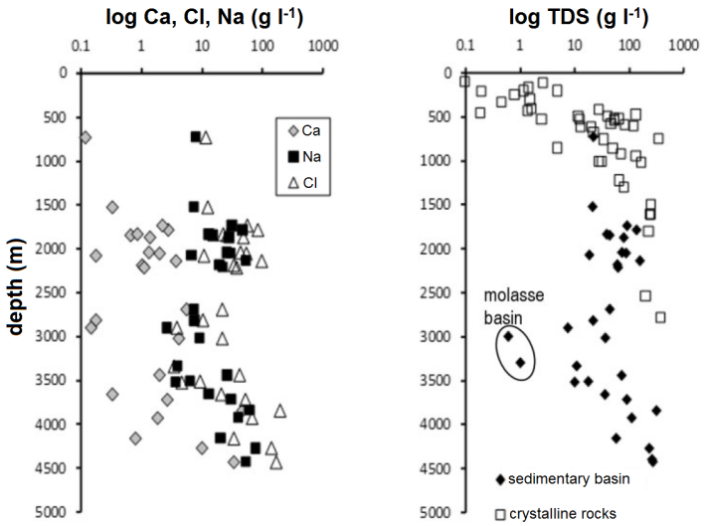


Figure 2.13: (Left) Depth distribution of Na, Cl and Ca of sedimentary basins fluids. (Right) Depth distribution of TDS of 76 samples (Huenges and Ledru, 2010).

2.3.3. Binary System H_2O – NaCl

During the literature survey of the present work, an extensive study on a binary system H_2O – NaCl was conducted. The examined fluid properties are elaborated in Section A.2. When this thesis progressed, it became clear that a ternary system discussed in Section 2.3.4 is essential to model the thermo-hydraulic behavior of the geothermal fluid inside the wellbore accurately. Nevertheless, it is believed that the completeness of the consulted literature on binary system H_2O – NaCl contributes to this thesis and to possible future research. Therefore, it has been appended as supplementary theory.

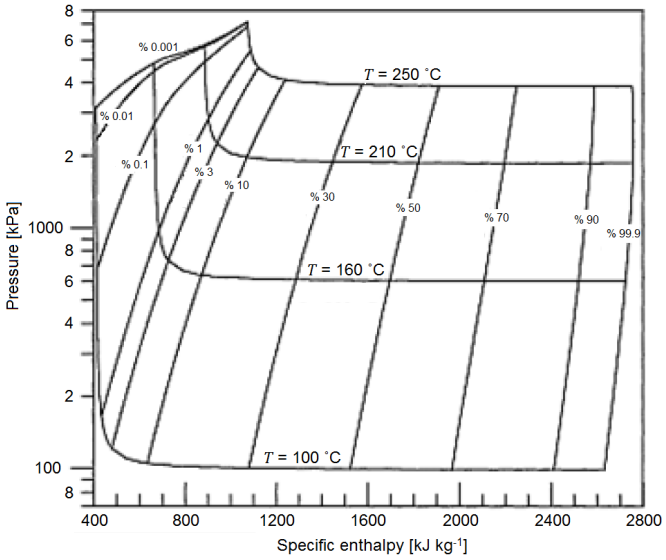


Figure 2.14: Pressure-enthalpy diagram of H_2O-CO_2 system, $\boldsymbol{w_{CO2} = 0.015}$ kg kg^{-1} (Hosgor et al., 2015).

2.3.4. Ternary System H_2O – NaCl – CO_2

In case of a significant amount of CO_2 dissolved in the geothermal fluid, there is the possibility of degassing in the production well during operation. This phenomenon has a major influence on the behavior of the geothermal fluid above the flashing point and in the geothermal power plant, e.g. decreasing density of two-phase flow induces less pressure loss, geothermal power plant performance, scaling potential (Khalifa and Michaelides, 1978; Duan and Sun, 2003; Kelessidis et al., 2007; Gokcen and Yildirim, 2008; DiPippo, 2012; Francke, 2014; Hosgor et al., 2015).

Many experimental studies on the solubility of CO_2 in pure water and NaCl(aq) have been conducted. Also, extensive effort has been done in modeling this phenomenon. Several models have been published, but few can accurately predict CO_2 solubility in a wide $T - P - m$ range. Li and Nghiem (1986) presented a model based on the Peng-Robinson equation of state (EOS), Henry's Law and the scaled-particle theory for temperatures up to 200 ˚C and molalities up to 4 mol kg^{-1}. But it is not accurate for NaCl(aq). Harvey and Prausnitz (1989) developed an EOS for the CO_2 solubility in NaCl(aq) at elevated pressures. However, it overestimates CO_2 solubility by 10 – 20% compared to experimental data. The EOS developed by Zuo and Guo (1991) underestimates CO_2 solubility significantly for a NaCl mass fraction of 20 wt% and high pressures and overestimates solubility at 6 wt% NaCl mass fractions and moderate pressures by more than 12% (Duan and Sun, 2003).

Duan and Sun (2003) presented an improved thermodynamic model for the solubility of CO_2 in water and NaCl(aq). Francke (2014) studied the thermo-hydraulic behavior of geothermal fluids at the research site in Gross Schoenebeck, Germany. A GFP model has been developed to calculate geothermal fluid properties (Francke et al., 2013). These two models are discussed in Sections 2.3.4.1 and 2.3.4.2, respectively.

2.3.4.1. Thermodynamic Model Duan and Sun (2003)

Duan and Sun (2003) developed an improved model calculating CO_2 solubility in pure water and NaCl(aq) valid for temperatures in the range of 273 – 533 K, pressures in the range of 0 – 2000 bar and molalities in the range of 0 – 4.3 mol kg^{-1}. The EOS was developed by applying a specific interaction theory for the liquid phase and an accurate EOS for the vapor phase, based on the EOS of Duan et al. (1992) and the theory of Pitzer (1973). The model is able to predict CO_2 solubility close to experimental uncertainty, which is approximately 7% in CO_2 solubility.

Additionally, a model for the phase equilibrium for $T - P - m$ range of 273 – 523 K, 0 – 2000 bar and 0 – 4.3 mol kg^{-1} and the density for $T - P - m$ range of 273 – 573 K, 0 – 1000 bar and 0 – 4.3 mol kg^{-1} has been published (Duan and Sun, 2003; Duan et al., 2006; Duan et al., 2007; Duan et al., 2008). The models are publically available on the internet (Zhenhao Duan Research Group, 2006).

2.3.4.2. Thermodynamic Model Francke et al. (2013)

Francke et al. (2013) developed a geothermal fluid property (GFP) model, called BrineProp, which has been used in the dissertation Francke (2014). The model has been developed specifically for the research site Gross Schoenebeck, Germany. BrineProp is free software available as a Modelica package. A VBA MS Excel version is available as well. The VBA MS Excel model is in the present work referred to as the "GFP Excel model". The model version, BrineProp_0.5.xlsm, has been made available by Heineken (2016).

The geothermal fluid has been modeled as a mixture of H_2O, salts (NaCl, KCl and $CaCl_2$) and NCG (CO_2, N_2 and CH_4), which are the main components at the Gross Schoenebeck site. The GFP Excel model calculates, for a given $P-T$ state, the gas mass fraction χ. Subsequently, the state variables h and ρ are calculated for separated phases and an effective homogeneous value is calculated according to χ. The relevant assumptions that were made for the model are outlined below.

Assumptions:

1. The geothermal fluid is a mixture of H_2O, NaCl, KCl, $CaCl_2$, CO_2, N_2 and CH_4 and the composition is set with mass fractions.
2. There are two possible phase states: liquid or two-phase with liquid and gas. The gas phase is an ideal mixture of water vapor and gases.
3. If the two-phase state is satisfied, thermodynamic equilibrium is instantly reached. H_2O and gases exchange between liquid and gas phase by degassing/dissolution or evaporation/condensation.
4. Salts are dissolved in the liquid phase and do not precipitate or evaporate.
5. Dissolution of gases in water is modeled according to their solubility as if that particular gas is present in its own. Interaction between gases is neglected. In two-phase state partial pressures equal degassing pressures according to Raoult's Law. The water vapor pressure is calculated with Raoult's Law. Degassing pressure of the gases is calculated with correlations depending on Henry's coefficient describing the non-ideal solution behavior at high pressures.
6. Evaporation enthalpy is considered. Boundary surface enthalpies, gas solution enthalpies and dilution enthalpies are neglected.
7. Dalton's Law is applied to calculate the total pressure of the gas phase.

For the exact calculation procedure, considering equations, correlations and algorithms is referred to Francke (2014). Relevant for the present work, CO_2 solubilities in the GFP Excel model were obtained from Duan et al. (2006). Effective specific volume (and subsequently density), enthalpy and specific heat capacity of a two-phase mixture were calculated as a mass-weighted average according to the mass fraction. Effective viscosity has not been considered.

The GFP Excel model was validated against literature data and field measurements. The density model predicts the density with a calculation error of less than 1.4%. The calculated viscosity has a relative error of approximately 6% according to the measured field data. This has been accepted, because of the minor importance of frictional pressure losses in comparison to hydrostatic pressure losses. Gas solubility and gas volume fraction have matched experimental data rather good for low salinities and low pressures. At high salinity nitrogen solubility has been overestimated and methane solubility has been underestimated significantly. The production well has been hydraulically and thermally validated as well. The boundary conditions of the GFP Excel model are presented in Table 2.5. Validity range of gas mass fractions have not been specifically given, because dissolved gas mass fractions are significantly low at the Gross Schoenebeck site, where N_2 is the most abundant NCG, but with only 0.744 g kg^{-1}.

Table 2.5: Boundary conditions GFP Excel model Francke (2014).

Quantity	Boundary conditions
P	1 – 1000 bar
T	0 – 260 °C
w_{NaCl}	0 – 6 mol kg^{-1}
w_{KCl}	0 – 4.5 mol kg^{-1}
w_{CaCl2}	0 – 3 mol kg^{-1}

2.3.5. Conclusions

The GFP Excel model of Francke et al. (2013) is incorporated in the GFP model developed in this work, because of the usefulness and high applicability. Duan and Sun (2003) is used to validate the GFP model of this work.

2.4. Flow Characteristics & Thermodynamics

The present section discusses the theory of reservoir flow (Section 2.4.1) and well flow (Section 2.4.2) in order to form a basis for the mathematical model of the geothermal power plant systems. Section 2.4.3 reviews the effect of artificial lift in geothermal wells on the flow characteristics from a theoretical viewpoint. Finally, the thermodynamics of the conversion process of two geothermal power plants (single-flash and binary cycle), presented in Section 2.1, are outlined in Section 2.4.4.

2.4.1. Reservoir Flow

Reservoir modeling is a complicated process, because the underground patterns of fractures and the porosity in the rock formation are unknown and it can behave dynamically. In this study, it is aimed for to develop a sub model for the reservoir that can interact with the fluid flow in the production well and injection well. Fluid flow in reservoirs and wells has been thoroughly studied, especially for oil and gas applications. It is far too complex to describe the flow in the reservoir analytically, because the flow path is unknown. Therefore, a lumped parameter approach is used to model the reservoir. In that case, the behavior of the flow is simplified and the values are averaged between the boundaries of the system. In order to simplify the flow the following three assumptions are made (Dake, 1978).

Assumptions:

1. The permeability of the reservoir is considered isotropic and the rock properties are homogeneous throughout the reservoir.
2. The production well is completed across the entire formation thickness and therefore assuming fully radial flow.
3. The pores in the rock formation are completely saturated with a single phase fluid.

Figure 2.15 represents a simplified reservoir-well system. Fluid flows radially and horizontally from the boundary of the reservoir towards the boundary of the well, from where it flows vertically inside the well (DiPippo, 2012).

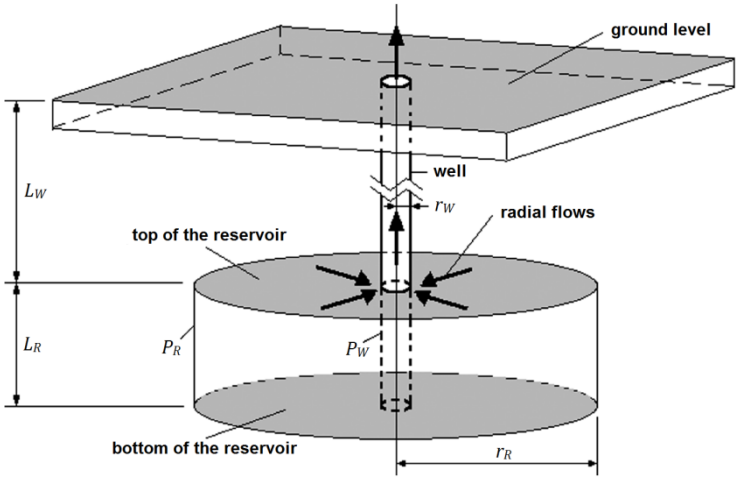


Figure 2.15: Schematic of a simplified reservoir-well system (DiPippo, 2012).

The basic differential equation for the radial flow of a fluid in a homogeneous porous medium, which is also referred to as the pressure diffusion equation, is used to calculate the pressure in the reservoir at a certain distance r from the production well as a function of time. It can be derived from the principle of mass conservation by substituting Darcy's Law for radial, horizontal flow and the basic thermodynamic definition of isothermal compressibility (Dake, 1978).

In order to derive the pressure diffusion equation a simplified model of the reservoir is presented in Figure 2.16, which corresponds to the reservoir model of Figure 2.15 with the associated assumptions. The conservation of mass inside an arbitrary cylindrical shell of thickness dr around the production well is defined by eq. (2.1).

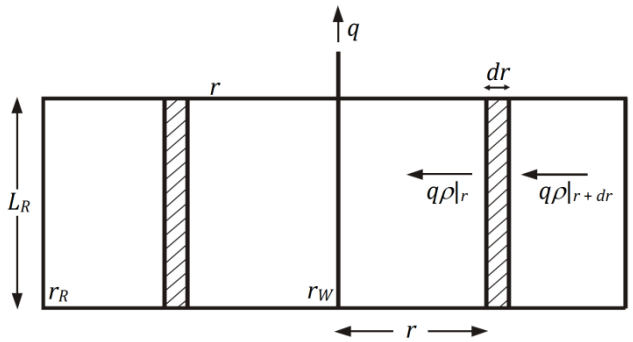


Figure 2.16: Radial flow of a single phase fluid in the vicinity of a producing well (Dake, 1978).

$$(q\rho)|_{r+dr} - (q\rho)|_r = 2\pi r L_R \Phi dr \frac{\partial \rho}{\partial t} \tag{2.1}$$

Where q is the volumetric flow rate [$m^3\ s^{-1}$], ρ is the density [kg m^{-3}], r is the radius [m], L_R is the vertical length or thickness of the reservoir [m], Φ is the porosity and t is the time [s]. The volume of the fluid inside the cylindrical shell is represented by $2\pi r L_R \Phi dr$. The left hand side of eq. (2.1) can be expanded to eq. (2.2), which simplifies to eq. (2.3).

$$(q\rho)|_r + \frac{\partial (q\rho)}{\partial r} dr - (q\rho)|_r = 2\pi r L_R \Phi dr \frac{\partial \rho}{\partial t} \tag{2.2}$$

$$\frac{\partial (q\rho)}{\partial r} = 2\pi r L_R \Phi \frac{\partial \rho}{\partial t} \tag{2.3}$$

Darcy's Law describes the flow of a fluid through a porous medium and relates the volumetric flow rate q across a surface to the pressure gradient $\partial P/\partial r$ across a section (Dake, 1978). The radial form is given by eq. (2.4).

$$q = 2\pi r L_R \frac{K}{\mu} \frac{\partial P}{\partial r} \tag{2.4}$$

Where K is the permeability [m^2], μ is the dynamic viscosity [Pa s], P is the pressure [Pa]. Now by substituting Darcy's Law, eq. (2.4), into the simplified form of the principle of mass conservation, eq. (2.3), eq. (2.5) is obtained.

$$\frac{\partial}{\partial r}\left(\frac{2\pi L_R r K}{\mu} \rho \frac{\partial P}{\partial r}\right) = 2\pi r L_R \Phi \frac{\partial \rho}{\partial t} \tag{2.5}$$

Where the reservoir thickness L_R is not a function of r, this can be simplified to eq. (2.6).

$$\frac{1}{r}\frac{\partial}{\partial r}\left(\frac{K\rho}{\mu} r \frac{\partial P}{\partial r}\right) = \Phi \frac{\partial \rho}{\partial t} \tag{2.6}$$

The time derivative of density in eq. (2.6) can be expressed as a time derivative of pressure by differentiating the basic thermodynamic definition of isothermal compressibility, given by eq. (2.7), with respect to time resulting in eq. (2.8).

$$c = -\frac{1}{V}\left(\frac{\partial V}{\partial P}\right)_T = \frac{1}{\rho}\left(\frac{\partial \rho}{\partial P}\right)_T \tag{2.7}$$

$$c\rho\frac{\partial P}{\partial t}=\frac{\partial \rho}{\partial t} \tag{2.8}$$

Where c is the isothermal compressibility [Pa^{-1}] and V is the volume [m^3]. Finally, by substituting eq. (2.8) into eq. (2.6), eq. (2.9) is obtained.

$$\frac{1}{r}\frac{\partial}{\partial r}\left(\frac{K\rho}{\mu}r\frac{\partial P}{\partial r}\right)=\Phi c\rho\frac{\partial P}{\partial t} \tag{2.9}$$

Eq. (2.9) is non-linear, because the coefficients on both sides are functions of pressure. In order to obtain an analytical solution it must be linearized by assuming that the single fluid flowing through the formation is liquid. Then by assuming μ and c are independent of pressure and therefore constant, eq. (2.9) reduces to eq. (2.10), which is referred to the basic equation for the radial flow of a fluid in a homogeneous porous medium or pressure diffusion equation (Dake, 1978).

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial P}{\partial r}\right)=\frac{\Phi\mu c}{K}\frac{\partial P}{\partial t} \tag{2.10}$$

Where $\Phi\mu c/K$ is now a constant. Depending on the initial and boundary conditions an infinite number of solutions can be obtained. The three most common solutions are transient, semi steady state and steady state. For the present work, it is assumed that the steady state solution describes the reservoir properties sufficiently. Figure 2.17 shows the radial flow of liquid fluid under steady state flow conditions. Steady state implies that the well produces at a constant volumetric flow rate q and that the pressure profile in the reservoir remains constant over time, so that $\partial P/\partial t=0$. Additionally, this assumption is allowed if the outer boundary pressure or reservoir pressure P_R remains constant, which can only be accomplished by natural influx of water at the outer boundary or by the injection of fluid through an injection well. The latter is the case in the present work, where it is assumed that the injection well is placed outside the boundary indicated by Figure 2.17.

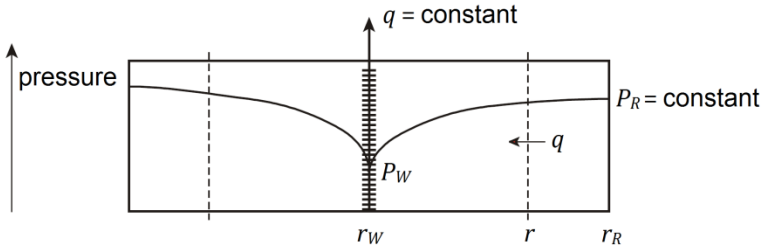


Figure 2.17: The radial flow pressure distribution of liquid geothermal fluid under steady state flow conditions (Dake, 1978).

The steady state condition of radial, horizontal flow in a reservoir is presented by eq. (2.11), which is the radial form of the Laplace equation with pressure only as function of the radius of the reservoir.

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial P}{\partial r}\right)=0 \tag{2.11}$$

The solution of this steady state diffusion equation is given by eq. (2.12), which can also be obtained by integrating eq. (2.4), which is Darcy's Law for radial, horizontal flow of a liquid through a porous medium. It can be seen that the pressure loss in the reservoir is a logarithmic function, which can also be seen in Figure 2.17.

$$\Delta P=P(r=r_R)-P(r=r_W)=\frac{\mu q}{2\pi KL_R}\ln\frac{r_R}{r_W} \tag{2.12}$$

Where ΔP is called the drawdown [Pa], which is the difference between the pressure at the far-field and at the well face. r_R is the radius of the reservoir [m] and r_W is the radius of the well at reservoir depth [m].

Finally, the skin factor is introduced due to the fact that during drilling, depletion or production of the well the pores can be partially plugged with drilling mud. It can lead to a reduction in permeability and therefore

an increased pressure drawdown in the vicinity of the wellbore. The additional pressure loss ΔP_{skin} near the wellbore, defined by Van Everdingen (1953), is considered to be caused by a skin and is given in eq. (2.13)

$$\Delta P_{skin} = \frac{\mu q}{2\pi K L_R} S \tag{2.13}$$

Where S is the skin factor. Substituting eq. (2.13) into eq. (2.12) gives the total pressure drop in the reservoir between the undisturbed flow in the far-field and the well.

$$\Delta P = P(r = r_R) - P(r = r_W) = \frac{\mu q}{2\pi K L_R}\left(ln\frac{r_R}{r_W} - S\right) \tag{2.14}$$

A geothermal reservoir will never be as ideal as described above and the non-uniform reservoir properties are difficult to establish. It is common practice to merge the reservoir properties and the fluid properties within the reservoir to a productivity index PI [kg s^{-1} Pa^{-1}] and an injectivity index II [kg s^{-1} Pa^{-1}]. The PI and the II can be determined by well tests. Consequently, eq. (2.14) reduces to eqs. (2.15) and (2.16) for PI and II, respectively.

$$PI = \frac{\dot{m}}{\Delta P} \tag{2.15}$$

$$II = \frac{\dot{m}}{\Delta P} \tag{2.16}$$

Where ΔP in eq. (2.15) is the pressure drawdown from the far field to the inlet of the production well. In eq. (2.16), ΔP is the pressure drawdown from the outlet of the injection well to the far-field. It has been widely accepted in reservoir engineering to calculate reservoir inflow and outflow conditions with the PI and II conditions (Dake, 1978; Cholet, 2008; Pruess, 2010; Grant and Bixley, 2011; Francke, 2014).

2.4.2. Well Flow

All equations in the current subsection can be used for both the production well as the injection well unless stated otherwise. The First Law of Thermodynamics for an open system in steady state flow can be applied to a well, given in eq. (2.17) (DiPippo, 2012).

$$\dot{Q} - \dot{W} = \dot{m}\left[(h_2 - h_1) + \frac{1}{2}(u_2^2 - u_1^2) + g(z_2 - z_1)\right] \tag{2.17}$$

Where $\dot{Q}$ is the rate of heat flow supplied to the system by its surroundings [W], $\dot{W}$ is the rate of work done by the system [W], $\dot{m}$ is the mass flow rate [kg s^{-1}], h is the enthalpy [J kg^{-1}], u is the velocity [m s^{-1}], g is the gravitational acceleration [m s^{-2}] and z is the elevation [m]. States 1 and 2 represent the bottom and top of the well, respectively. Garcia-Gutierrez et al. (2002) used an analytical solution for the heat flow rate $\dot{Q}$, which has been found accurate for geothermal applications, given in eq. (2.18).

$$\dot{Q} = \frac{4\pi k_r(T_{gf} - T_g)}{\ln\left(\frac{4\alpha_r t}{\gamma r_W^2}\right)} \tag{2.18}$$

Where k_r is the rock thermal conductivity [W m^{-1} K^{-1}], T_{gf} is the temperature of the geothermal fluid [°C], T_g is the geothermal temperature [°C], α_r is the rock thermal diffusivity [m^2 s^{-1}], t is the time [s], γ is Euler's constant (1.78) and r_W is the inner radius of the well [m].

In order to know the pressure as a function of height the momentum equation of fluid mechanics (eq. (2.19)) is applied, which is an application of Newton's Second Law of Motion (DiPippo, 2012).

$$-dP - \frac{dF}{A} - \rho g dz = \rho u du \tag{2.19}$$

Where the left hand side represents the forces per unit area acting on an elemental fluid body of length dz and the right hand side gives the mass times acceleration per unit area. The elemental friction force dF is derived from the dimensionless expression for momentum transfer to a wall given in eq. (2.20) (Mills, 1998).

$$\tau_{wall} = \frac{1}{2}\rho u^2 C_f \tag{2.20}$$

Where τ_{wall} is the wall shear stress [N m^{-2}]. C_f is the skin friction factor, which is simply related to the Darcy friction factor for fully developed flow according to eq. (2.21) (Mills, 1998).

$$f = 4C_f \tag{2.21}$$

Now by substituting eq. (2.21) into eq. (2.20) and rewriting the wall shear stress to an elemental friction force divided by the elemental wall surface, eq. (2.22) is obtained.

$$dF = \frac{1}{2}\rho u^2 \frac{f}{4} C dz \tag{2.22}$$

Where $C = \pi D_i$, is the circumference of the well interior [m].
By integrating eq. (2.19), after substituting eq. (2.22), from the bottom to the top of the well the pressure difference can be expressed by eq. (2.23) (DiPippo, 2012).

$$P_1 - P_2 = \int_{u_1}^{u_2} \rho(z)u(z)du + \frac{1}{2D_i}\int_{z_1}^{z_2} f\rho(z)u^2(z)\,dz + g\int_{z_1}^{z_2} \rho(z)\,dz \tag{2.23}$$

2.4.2.1. Liquid-Only Flow

If the pressure of the hot geothermal fluid falls below the saturated liquid pressure flashing occurs inside the well. The point in the wellbore where flashing starts is called the flash horizon or flash point. According to Ryley (1980), it is convenient to integrate eq. (2.23) separately for liquid phase flow below the flash point and the two-phase flow above the flash point. For liquid flow, the friction factor can be found from the Swamee-Jain equation (eq. (2.24)) (Swamee and Jain, 1976).

$$f = \frac{0.25}{\left\{\log_{10}\left[\frac{\varepsilon/D_i}{3.7} + \frac{5.74}{\mathrm{Re}^{0.9}}\right]\right\}^2} \tag{2.24}$$

Where ε is the absolute pipe roughness [m] and Re is the Reynolds number given by eq. (2.25).

$$\mathrm{Re} = \frac{\rho u D_i}{\mu} \tag{2.25}$$

Where the velocity u is calculated by eq. (2.26).

$$u = \frac{\dot{m}}{\rho A_{CS}} \tag{2.26}$$

For liquid-only flow the acceleration term is assumed to be negligible due to negligible compressibility of the liquid. Additionally, velocity and density can therefore be taken as constants. After integration eq. (2.23) reduces to eq. (2.27).

$$P_1 - P_{FP} = \frac{f\rho u^2 L_{1-FP}}{2D_i} + g\rho L_{1-FP} \tag{2.27}$$

Where P_1 is the pressure at the bottom of the well [Pa], P_{FP} is the pressure at the flashpoint [Pa], f is the Darcy friction factor and L_{1-FP} is the distance from the bottom of the well to the flash point [m]. It must be noted that the sign of the frictional pressure loss in eq. (2.27) depends on the direction of motion, giving it a negative sign for the injection well.

2.4.2.2. Two-Phase Flow

DiPippo (2012) adopted the lumped-parameter approach suggested by Ryley (1980) for the two-phase flow in the production well above the flash point. In that case, mean effective values were used for the two-phase ρ, f, u and eq. (2.23) is integrated to eq. (2.28).

$$P_{FP} - P_2 = \frac{\bar{\rho}_m}{2}(u_2^2 - u_{FP}^2) + \frac{1}{2D_i}\bar{f}_m\bar{\rho}_m\bar{u}_m^2(z_2 - z_{FP}) + g\bar{\rho}_m(z_2 - z_{FP}) \tag{2.28}$$

Where m stands for liquid-gas mixture. Ryley (1980) concluded that the calculation of $\bar{u}_m$ led to difficulties if there is a large velocity change in the pipe section from the flash point to the top of the well. The problem eased progressively by subdivision in smaller segments. Figure 2.18 shows the analysis of a pipe segment to solve the well flow numerically. Then eq. (2.28) is rewritten to eq. (2.29) for the pressure change in a pipe segment.

$$P - (P + \Delta P) = \rho_m u_m\big(u_m - (u_m + \Delta u_m)\big) + \frac{1}{2D_i}f_m\rho_m u_m^2\big(z - (z + \Delta z)\big) + g\rho_m\big(z - (z + \Delta z)\big) \tag{2.29}$$

Where ρ_m is calculated by eq. (2.30) and u_m is calculated by eq. (2.31).

$$\rho_m = \rho_g\varepsilon_g + \rho_l\big(1 - \varepsilon_g\big) \tag{2.30}$$

$$u_m = \frac{\dot{m}}{\rho_m A_{CS}} \tag{2.31}$$

Where ρ_g and ρ_l are the densities of the gas phase and liquid phase [kg m^{-3}], respectively, and ε_g is the cross-sectional void fraction [m^2 m^{-2}], $\dot{m}$ is the mass flow [kg s^{-1}] and A_{CS} is the cross-sectional area of the pipe [m^2].

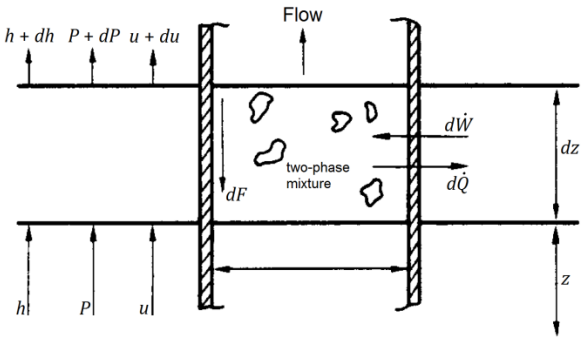


Figure 2.18: Analysis of a pipe segment (modified from Ryley (1980)).

The definition for the two-phase friction factor f_m differs widely in published literature. DiPippo (2012) states that the average friction factor for two-phase flow cannot be expressed in terms of other mean effective quantities. Therefore, multipliers are used in the range of 2 – 3 applied to the liquid-only friction factor. Wallis (1969) used constant values for the two-phase friction factor in wellbores of 0.025, which was also adopted by Garcia-Gutierrez et al. (2002) for their study on flow production characteristics in deep geothermal wells. Wisman (1975) developed a simple correlation consistent for all flow regimes for

adiabatic two-phase vertical flow given in eq. (2.32) and (2.33), which was practical from an engineering purpose and more favorable than the well-known Lockhart-Martinelli correlation.

$$f_m = 0.0056 + \frac{0.5}{\mathrm{Re}_m^{0.32}} \tag{2.32}$$

$$\mathrm{Re}_m = \frac{\rho_l u_l D_i}{\mu_l}(1-\varepsilon_g)\left(1-\sqrt{\varepsilon_g}\right) \tag{2.33}$$

Where ρ_l, u_l and μ_l are the density, velocity and dynamic viscosity of the liquid phase respectively and ε_g is the void fraction. Chadha et al. (1993) developed a more comprehensive model, but less practical, for two-phase flow in a geothermal well, in which the friction factor depends on the flow regime of the two-phase flow. Hasan et al. (2002) used a modification of the correlation from Chen (1979) to model two-phase flow in wellbores given in eq. (2.34) and (2.35), which is an explicit equation.

$$f_m = \frac{1}{4\log_{10}\left[\frac{\varepsilon}{3.7065 D_i} - \frac{5.0452}{\mathrm{Re}_m}\log_{10}\left(\frac{1}{2.8257}\left(\frac{\varepsilon}{D_i}\right)^{1.1098} + \frac{5.8506}{\mathrm{Re}_m^{0.8981}}\right)\right]^2} \tag{2.34}$$

$$\mathrm{Re}_m = \frac{\rho_m u_m D_i}{\mu_m} \tag{2.35}$$

Where ε is the pipe roughness [m], ρ_m is given by eq. (2.30), u_m by eq. (2.31) and μ_m is the mass weighted average of u_l and u_g.

2.4.2.3. Gas Flow in Gas Lift Duct

Figure 2.19 is a schematic of a production well equipped with a gas lift system. It is assumed that the gas lift duct is annular. The First Law of Thermodynamics (eq. (2.17)) also applies to the gas flow in the gas lift duct. The heat flow rate $\dot{Q}_g$ between the surrounding rock formation and the gas in the gas lift duct is given by eq. (2.18). The heat flow rate between the geothermal fluid flowing upwards in the production tubing and the gas flowing downwards in the annular duct is given by eq. (2.36).

$$\dot{Q}_{gf} = UA\left(T_{GL} - T_{gf}\right) \tag{2.36}$$

The calculation procedure of the overall heat transfer coefficient U is discussed in Section A.3. The corresponding equations are given by eqs. (A.23) – (A.41). The First Law of Thermodynamics for an open system in steady state flow for the gas lift duct changes to eq. (2.37).

$$\dot{Q}_{gf} + \dot{Q}_g - \dot{W} = \dot{m}_{GL}\left[(h_2 - h_1) + \frac{1}{2}(u_2^2 - u_1^2) + g(z_2 - z_1)\right] \tag{2.37}$$

State 1 represents now the top of the gas lift duct and state 2 represents the bottom of the gas lift duct. The bottom of the gas lift duct corresponds to the depth of the gas lift valve. The First Law of Thermodynamics for the production tubing in this particular case is still described by eq. (2.17), only $\dot{Q}$ is replaced by $\dot{Q}_{gf}$, where state 1 is the bottom of the production tubing and state 2 is the top of the production tubing according to the flow direction.

The pressure loss in the gas lift duct is calculated according to eq. (2.23). Furthermore, eqs. (2.24), (2.25) and (2.26) also apply to the gas in the gas lift duct for the friction factor, Reynolds number and velocity, respectively.

2.4.3. State of the Art - Modeling Artificial Lift in Wells

Basically, the physics behind the lifting techniques can be divided in two different phenomena as has been discussed in Section 2.2: 1. pressurizing the geothermal fluid with a pump or 2. decreasing the density by

mixing the geothermal with a gas or mixture of gases (gas lift). The effect of these phenomena on flow characteristics inside the wellbore and the way it can be modeled are discussed in the present section.

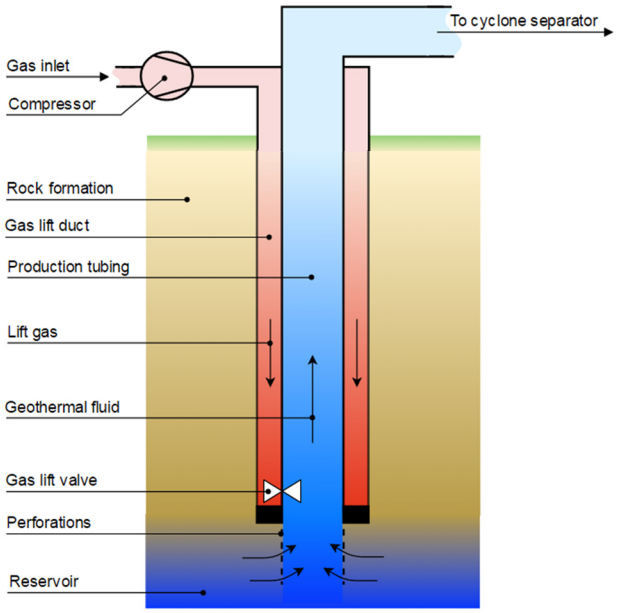


Figure 2.19: Schematic of a production well equipped with a gas lift system.

2.4.3.1. Pump Model

Downhole pumps have been installed in many low-enthalpy geothermal systems to prevent the fluid from boiling and to increase energy production. The installation depth of the pump is crucial to avoid cavitation, which is the boiling of liquid, forming vapor cavities in the liquid. This is caused by local static pressure decrease due to the increase of flow velocity around propeller blades. Cavitation causes reduction in flow rate and efficiency and it can heavily damage pump components. According to Aksoy (2007), parameters having an effect on installation depth are the characteristics of the geothermal fluid (T, P, w_{CO2} and w_{NaCl}) in the reservoir, reservoir permeability, the production flow rate and wellbore characteristics. The pressure drop induced by friction in the pump itself is 1-10 kPa, which is a negligible amount compared to other pressure losses (Lienau et al., 1991).

The required power for the pump depends partly on the required ΔP_p, which is on its turn depending on the characteristics of the geothermal fluid, the production flow rate and the wellbore characteristics above the pump installation depth. The required ΔP_p must be higher than the pressure loss from the pump to the wellhead in order to avoid flashing above the downhole pump. The pumping power is determined by eq. (2.38).

$$\dot{W}_p = \frac{v\Delta P}{\eta_p}\dot{m} \tag{2.38}$$

Subsequently, it is entered as the $\dot{W}$ term in the First Law of Thermodynamics, which is given in eq. (2.17), to calculate the energy increase of the fluid. Francke (2014) modeled the pump as non-isentropic, adiabatic, with no physical height or length. In Table 2.3, it has been shown that the efficiency of an ESP is 50 – 65 %.

2.4.3.2. Gas Lift Model

As it has been discussed in Section 2.2.3.4, in gas lift systems a certain gas(mixture) is injected to decrease the density of the fluids. Subsequently, the gravitational pressure loss above the point of injection is reduced. From the injection point to the wellhead a gas-liquid mixture flows in the wellbore, where the two-phase mixture may flow in a variety of patterns. The flow pattern developed in the conduit depends on the flow rates, the fluid properties and the tube size (Taitel et al., 1980). Generally, four flow patterns for

upward cocurrent flow are commonly distinguished, which can be seen in Figure 2.20 (Guet, 2004; Kelessidis et al., 2007).

1. Bubble flow: this pattern is characterized by a uniformly distributed gas phase in the form of discrete bubbles in a continuous liquid phase. It corresponds to low void fractions. Bubble flow can be separated in two different turbulent subcases.
    a. Bubble flow: low to moderate liquid flow, causes almost no bubble break-up. The bubble size is affected by entrance conditions and devices.
    b. Finely dispersed bubble flow: this regime corresponds to large liquid flow. The bubbles are broken into small bubbles, due to turbulence. The maximum bubble diameter is affected by turbulence conditions and surface tension properties (Guet, 2004)
2. Slug flow: large bullet shaped bubbles (Taylor bubbles) form containing most of the gas, which move uniformly upward. Between Taylor bubbles slugs of continuous liquid containing small gas bubbles arise. Between Taylor bubbles and pipe wall, a liquid film flows downward.
3. Churn flow: more chaotic and disordered form of slug flow. Taylor bubbles are narrower and distorted. The continuity of the liquid between Taylor bubbles is repeatedly destroyed by a high local gas concentration in the liquid slug. Typical is the oscillatory motion of the liquid.
4. Annular flow: a continuous gas phase exists in the core of the pipe. The liquid phase flows partially as wavy liquid film along the pipe wall and the other part as liquid drops entrained in the gas phase.

Multiphase flow effects in wellbores can have an impact on overall system characteristics and performance, e.g. of reservoirs and surface facilities. Therefore, accurate multiphase models describing well flow must be incorporated into the numerical model to optimize the performance of the total system. Commonly, there are three types of well flow models used, which are: empirical correlations, homogeneous models, and mechanistic models. Empirical correlations are obtained from curve fitting of experimental data. Disadvantage of this method is the limited applicability to the range of variables used in the experiments. In homogeneous models the fluid properties are represented by mixture properties. Single-phase techniques can be applied to this mixture. Additionally, slip between phases can be introduced, which requires a set of empirical parameters. These models with slip are in literature referred to as the drift-flux model, which has been proposed firstly by Zuber and Findlay (1965). Finally, mechanistic models are generally considered as the most accurate, because detailed physical equations of different flow patterns describe the behavior of the well flow (Shi et al., 2005).

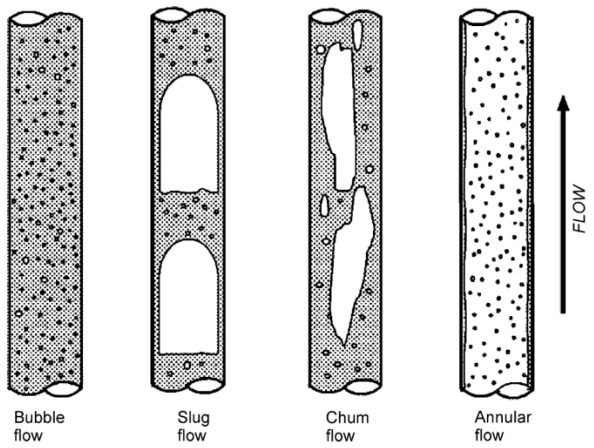


Figure 2.20: Schematic depiction of flow patterns in vertical flow (Kelessidis et al., 2007).

According to Shi et al. (2005), mechanistic models can cause discontinuities in pressure drop and holdup at flow-pattern transitions. From a modeling perspective, these discontinuities can induce convergence problems in a numerical model. One solution to the convergence problems can be smoothing at transitions. Alternatively, a homogeneous model can be applied, because it is relatively simple, continuous, and differentiable. Drift-flux models are therefore a good choice for use in wellbore simulators. Woldesemayat and Ghajar (2007) compared 68 void fraction correlations, and classified them into four categories: slip ratio, $K\varepsilon_H$, drift-flux and general void fraction, for different flow patterns in horizontal and vertical inclined pipes with experimental data. Out of six best performing correlations, five were developed based on the

drift-flux model. Godbole et al. (2011) compared 52 void fraction correlations and concluded that drift-flux correlations are among the most accurate for upward vertical two-phase flow, consistent with the findings of Woldesemayat and Ghajar (2007). Thome (2010) reviewed multiple void fraction correlations and concluded that the drift-flux model must be the preferred choice as well. Therefore, in the present work the drift-flux method for two-phase flow has been adopted.

2.4.3.3. Drift-Flux Correlations

Zuber and Findlay (1965) have been the first to develop the drift-flux model, although Wallis (1969) and Ishii (1977) added substantially to its development (Thome, 2010). Since the introduction of the drift-flux model it has been refined and used many times. The present work focuses on modeling gas-liquid flow in large diameter pipes. Most of the empirical parameters in literature are determined from experiments in small diameter pipes, which are not by definition applicable to flow in wellbores. Shi et al. (2005) conducted experiments in large diameter pipes comparable with wellbores. They have shown that their optimized parameters provide better agreement to the experimental data than existing default parameters. Kelessidis et al. (2007) used the void fraction correlations and flow pattern transitions proposed by Taitel et al. (1980) for simulation of wells with geothermal water containing dissolved CO_2. Their model has been found suitable for bubble, dispersed bubble and slug flow, but less suitable for churn and annular flow. According to Woldesemayat and Ghajar (2007), the best performing drift-flux correlations were Rouhani and Axelsson (1970), Dix (1971) and Toshiba (Coddington and Macian, 2002). It must be noted that these correlations were compared with experimental data for a pipe with an internal diameter of 45.5 mm. Godbole et al. (2011) compared void fraction correlations to different sets of experimental data with various internal diameters up to 76 mm. Dix (1971) and Toshiba (Coddington and Macian, 2002) were not evaluated, Nicklin (1961) and again Rouhani and Axelsson (1970) have shown the smallest error. Hasan et al. (2010) developed a simplified, yet comprehensive, two-phase flow model for wellbores using the drift-flux approach. A comparative study with mechanistic and widely used empirical models was included and they have shown that these models behave in a similar fashion as the proposed simplified model. In the present study, the simplified drift-flux model of Hasan et al. (2010) is applied to incorporate two-phase gas-liquid flow within the numerical model of the wellbore. Shen et al. (2010) and Schlegel et al. (2010) evaluated the drift-flux correlations for large diameters pipes (Hills, 1976; Ishii, 1977; Clark and Flemmer, 1984; Kataoka and Ishii, 1987; Hibiki and Ishii, 2003). It has been concluded that none of the correlations described all flow patterns succesfully. Akbar et al. (2016) developed a finite element model for two-phase flow in geothermal wellbores. Three different drift-flux correlations were implemented. Rouhani and Axelsson (1970) have shown the best fit with field data from the well NWS-1 Sabalan with an internal diameter of 0.24 m, followed by the correlation of Shi et al. (2005).

Based on the literature survey in the present work, the drift-flux correlations of Nicklin (1961), Rouhani and Axelsson (1970), Dix (1971), Toshiba (1989) and Hasan et al. (2010) are further discussed.

The drift-flux model basically applies two parameters: the flow distribution parameter C_0 and the drift-flux velocity u_{gu}. The distribution parameter takes non-uniform flow and concentration profiles into account. Whereas the drift-flux velocity takes into account velocity differences between gas and liquid flow and thereby slip. More specifically, drift-flux velocity is the velocity of the gas phase relative to the mixture velocity. The gas phase velocity is higher than the liquid phase velocity, because of its buoyancy. Furthermore, it has the tendency to flow near the channel center, where the velocity is higher than at the pipe walls. The equations in the present section are based on a one-dimensional flow. The in-situ gas velocity u_g is given by eq. (2.39).

$$u_g = C_0 u_m + u_{gu} \tag{2.39}$$

Where u_m is the average mixture velocity [m s^{-1}]. For cocurrent flow, the mixture velocity u_m is calculated by eq. (2.40).

$$u_m = u_{sg} + u_{sl} \tag{2.40}$$

Where u_{sg} and u_{sl} are the gas and liquid superficial velocities [m s^{-1}], respectively. The superficial velocity is a hypothetical velocity calculated as if the phase was flowing alone in the particular cross sectional area.

The in-situ gas velocity can also be expressed as the ratio of superficial gas velocity u_{sg} and void fraction ε_g $\left(u_g = u_{sg}/\varepsilon_g\right)$, which gives eq. (2.41) after substitution in eq. (2.39).

$$\varepsilon_g = \frac{u_{sg}}{C_0 u_m + u_{gu}} \tag{2.41}$$

The void fraction ε_g can be used to calculate the mixture density, which is given earlier in eq. (2.30). The mixture density is applied in the calculation of the total pressure gradient given by eq. (2.29). Eqs. (2.39) – (2.41) form the basis of the drift-flux model. Table 2.6 gives an overview of the flow distribution parameter and the drift-flux velocity for various correlations considered in the present study.

Table 2.6: Expressions for distribution parameter and drift-flux velocity of various correlations considered in the present work.

Correlation	Distribution parameter	Drift-flux velocity
Nicklin (1961)	$C_0 = 1.2$	$u_{gu} = 0.35\sqrt{gD}$
Rouhani and Axelsson (1970)	$C_0 = 1.1$ for $J > 200$ kg m^{-2} s^{-1} $C_0 = 1.54$ for $J < 200$ kg m^{-2} s^{-1} or $C_0 = 1.0 + 0.2(1-\chi)$	$u_{gu} = 1.18\left[g\sigma\left(\rho_l - \rho_g\right)/\rho_l^2\right]^{1/4}$
Dix (1971)	$C_0 = \frac{u_{sg}}{u_m}\left[1 + \left(u_{sl}/u_{sg}\right)^{\left(\rho_g/\rho_l\right)^{0.1}}\right]$	$u_{gu} = 2.9\left[g\sigma\left(\rho_l - \rho_g\right)/\rho_l^2\right]^{1/4}$
Toshiba (Coddington and Macian, 2002)	$C_0 = 1.08$	$u_{gu} = 0.45$

Hasan et al. (2010) proposed a more comprehensive model for two-phase flow in wellbores. In that study, the flow parameter C_0 and the drift-flux velocity u_{gu} depend on flow pattern, inclination angle, flow direction and phases. Gas lift is restricted to vertical wells in the present study, so well deviation can be neglected. In production wells the flow direction is assumed to be cocurrent. Table 2.7 presents the values of C_0 and u_{gu} for cocurrent flow direction and different flow patterns according to Hasan et al. (2010). The drift-flux velocity is expressed in different correlations for the bubble-rise velocity.

Table 2.7: Parameters for fully developed co-current flow and flow pattern (Hasan et al., 2010).

Flow pattern	Distribution parameter C_0	Drift-flux velocity u_{gu}
Bubble	$C_{0b} = 1.2$	$u_{\infty b}$
Slug	$C_{0s} = 1.2$	$\overline{u}_{\infty}$
Churn	$C_{0c} = 1.15$	$\overline{u}_{\infty}$
Dispersed bubble	$C_{0c} = 1.15$	$u_{\infty b}$
Annular	$C_{0a} = 1.0$	0

The Harmathy correlation, given by eq. (2.42), represents the small bubble rise velocity independent of flow direction, well deviation and annular geometry (Harmathy, 1960).

$$u_{\infty b} = 1.53\left[g\left(\rho_l - \rho_g\right)\sigma/\rho_l^2\right]^{1/4} \tag{2.42}$$

Where g is the gravitational acceleration [m^2 s^{-1}], ρ_l is the density of the liquid phase [kg m^{-3}], ρ_g is the density of the gas phase [kg m^{-3}] and σ is the surface tension [kg m^{-2}]. In slug flow Taylor bubbles are formed and the associated rise velocity is influenced by inclination angle and annular geometry. The Taylor bubble rise velocity is given in eq. (2.43).

$$u_{\infty T} = 0.35\sqrt{gD_i\left(\rho_l - \rho_g\right)/\rho_l}F_\theta \tag{2.43}$$

Where D_i is the diameter of the wellbore [m] and F_θ is the well-deviation factor given by eq. (2.44).

$$F_\theta = \sqrt{\cos\theta}(1+\sin\theta)^{1.2} \tag{2.44}$$

In slug flow, the liquid slug between the Taylor bubbles also contain small bubbles, whose rise velocity can be expressed by eq. (2.42). Consequently, the drift-flux velocity for slug flow, given in eq. (2.45) is calculated as an average of small bubble rise velocity and Taylor bubble rise velocity.

$$u_{gu} = \overline{u}_\infty = u_{\infty b}\left(1-e^{-0.1u_{gb}/(u_{sg}-u_{gb})}\right) + u_{\infty T}\left(e^{-0.1u_{gb}/(u_{sg}-u_{gb})}\right) \tag{2.45}$$

Where u_{gb} is the superficial gas velocity needed for transition from bubble to slug flow [m s^{-1}] and u_{sg} is the superficial gas velocity [m s^{-1}]. Eq. (2.45) is also used for churn flow. Finally, for annular flow eq. (2.41) is applied with a bubble rise velocity of 0, since it has been shown that with annular flow the effects of well orientation, geometry and flow direction are negligible (Hasan et al., 2010). In order to calculate the void fraction ε_g, it is required to establish the transition criteria and subsequently the associated flow pattern. Many researchers have shown that the transition from bubble to slug flow occurs at a void fraction ε_g of 0.25 in vertical systems. The bubble rise velocity and the distribution parameter are modified to account for smooth transition between different flow patterns. For cocurrent upward flow, the superficial gas velocity needed for transition from bubble flow u_{gb} is given by eq. (2.46).

$$u_{gb} = \frac{C_0 u_{sl} + u_{gu}}{4 - C_0} \tag{2.46}$$

Where the flow parameter C_0 associated with transition from bubble to slug flow is expressed by eq. (2.47). The bubble rise velocity u_{gu} is calculated by eq. (2.45).

$$C_0 = C_{0b}\left(1-e^{-0.1u_{gb}/(u_{sg}-u_{gb})}\right) + C_{0s}\left(1-e^{-0.1u_{gb}/(u_{sg}-u_{gb})}\right) \tag{2.47}$$

Where the flow parameters C_{0b} and C_{0s} can be found in Table 2.7. Both eq. (2.45) and (2.47) depend on u_{gb}, which on its turn depend on C_0 and u_{gu}, making an iterative procedure necessary to calculate the values.

Bubble flow cannot exist, turning into dispersed bubble flow, when the mixture velocity u_m is higher than the minimum mixture velocity for dispersed bubble flow u_{ms}, which can be checked by eq. (2.48).

$$2u_{ms}^{1.2}\left(\frac{f}{2D}\right)^{0.4}\left(\frac{\rho_l}{\sigma}\right)^{0.6}\sqrt{\frac{0.4\sigma}{g(\rho_l-\rho_g)}} = 0.725 + 4.15\sqrt{\frac{u_{sg}}{u_m}} \tag{2.48}$$

Where f is the Darcy friction factor for two-phase flow calculated by eq. (2.34). Transition from dispersed bubble to churn flow occurs at high enough gas velocity according to eq. (2.49).

$$u_{sg} > 1.08u_{sl} \tag{2.49}$$

The transition from slug to churn flow occurs when both criteria stated for eq. (2.48) and eq. (2.49) are true. Eq. (2.45) and eq. (2.46) are also valid for this transition, supplemented by eq. (2.50), where the flow parameter for transition from slug to churn flow is calculated.

$$C_0 = C_{0s}\left(1-e^{-0.1u_{ms}/(u_m-u_{ms})}\right) + C_{0c}\left(1-e^{-0.1u_{ms}/(u_m-u_{ms})}\right) \tag{2.50}$$

Where C_{0c} is the fully developed flow parameter for churn flow. Finally, the transition from churn to annular flow occurs when the superficial gas velocity u_{sg} is higher than the superficial gas velocity needed for transition from churn to annular flow u_{gc}, which is given by eq. (2.51).

$$u_{gc} = 3.1\left[g\sigma\left(\rho_l - \rho_g\right)/\rho_g^2\right]^{1/4} \tag{2.51}$$

In Table 2.7 it can be seen that the bubble rise velocity is 0. The flow parameter, given in eq. (2.52), is derived from the fully developed flow parameters of annular flow and the adjoining churn flow. As an additional requirement, annular flow exists if the void fraction $\varepsilon_g > 0.7$.

$$C_0 = C_{0c}\left(1 - e^{-0.1u_{gc}/(u_{sg}-u_{gc})}\right) + C_{0a}\left(1 - e^{-0.1u_{gc}/(u_{sg}-u_{gc})}\right) \tag{2.52}$$

Where C_{0a} is the fully developed flow parameter for annular flow.

2.4.4. Thermodynamics Geothermal Power Plants

The analysis in this section is based on the principle of energy, mass and momentum conservation. First the thermodynamic principles of the single-flash power plant are discussed. Subsequently, thermodynamics related to the binary cycle power plant are presented. Thermodynamics related to other geothermal power plants are discussed in Section A.4.

2.4.4.1. Single-Flash Steam Power Plant

Figure 2.21 shows the temperature-entropy diagram for a single-flash plant (DiPippo, 2012). The operation of single-flash plants has already been discussed in section 2.1.1.

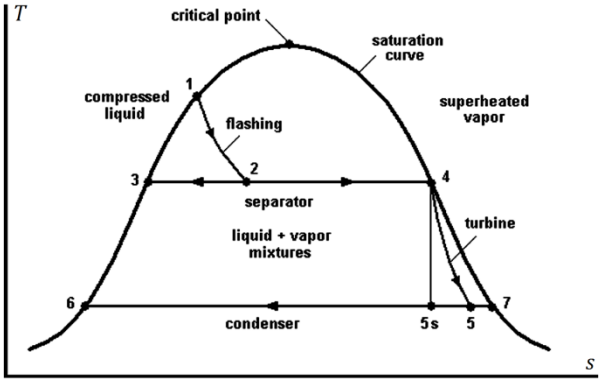


Figure 2.21: Temperature-entropy diagram for single-flash plants (DiPippo, 2012).

At state 1, the geothermal fluid starts flashing isenthalpically, because it happens spontaneously, adiabatically and without work involvement. Additionally, changes in kinetic or potential energy are neglected. This results in eq. (2.53).

$$h_1 = h_2 \tag{2.53}$$

Then the separation process in the cyclone separator occurs isobarically. The quality of the mixture at state 2 is given by eq. (2.54).

$$\chi_2 = \frac{h_2 - h_3}{h_4 - h_3} \tag{2.54}$$

The steam mass fraction goes to the turbine. The power produced by the turbine is given by eq. (2.55).

$$\dot{W}_t = \chi_2\, \dot{m}_2(h_4 - h_5) \tag{2.55}$$

Where the enthalpy at state 5 is calculated by eq. (2.56).

$$h_5 = h_4 - \eta_t(h_4 - h_{5s}) \tag{2.56}$$

Where η_t is the isentropic turbine efficiency and h_{5s} is the enthalpy after isentropic expansion. The gross electrical power is then given by eq. (2.57).

$$\dot{W}_e = \eta_g \dot{W}_t \tag{2.57}$$

The isentropic efficiency of a turbine is affected by the amount of moisture according to Baumann (1921). The Baumann rule says that 1% moisture causes approximately a 1% drop in turbine efficiency. The isentropic efficiency is then given by eq. (2.58).

$$\eta_t = \eta_{tw} = \eta_{td}\left[\frac{\chi_4 - \chi_5}{2}\right] \tag{2.58}$$

Where w and d stand for wet and dry, respectively. The condensing process of the expanded steam after the turbine is expressed by eq. (2.59).

$$\dot{m}_{cw} c_{p,w} \Delta T_{cw} = \chi_2 \, \dot{m}_2 (h_5 - h_6) \tag{2.59}$$

Where cw stands for cooling water. The condenser pump, cooling water pump and injection pump consume power according to eq. (2.60), where the liquid is assumed to be incompressible and thus v is constant.

$$\dot{W}_p = \frac{v \Delta P}{\eta_p} \, \dot{m} \tag{2.60}$$

Where v is the specific volume [m^3 kg^{-1}], ΔP is the pressure difference induced by the pump and η_p is the pump efficiency. The net power is then calculated by eq. (2.61).

$$\dot{W}_{net} = \dot{W}_e - \dot{W}_{ip} - \dot{W}_{cp} - \dot{W}_{cwp} \tag{2.61}$$

In literature there are different definitions for thermal efficiencies and utilization efficiencies. According to DiPippo (2012), for a closed cycle the thermal efficiency is often described by eq. (2.62).

$$\eta_{th} = \frac{\dot{W}_e}{\dot{Q}_{in}} \tag{2.62}$$

Where $\dot{Q}_{in}$ is the rate of heat flow transferred from the geothermal fluid. The utilization efficiency given in eq. (2.63), also referred as the Second Law (exergetic) efficiency, compares the actual power output to the available theoretical power.

$$\eta_u = \frac{\dot{W}_{net}}{\dot{E}} \tag{2.63}$$

Where the theoretical power $\dot{E}$ is calculated by eq. (2.64)

$$\dot{E} = \dot{m}_2 e = \dot{m}_2 \left[h(T,P) - h(T_0,P_0) - T_0[s(T,P) - s(T_0,P_0)]\right] \tag{2.64}$$

Where e is the specific exergy [J kg^{-1}] and 0 stands for dead-state or ambient conditions. The comparison of geothermal power plant performance is often done by their utilization efficiency. That is the thermal efficiency of the single-flash plant is arbitrary, because it does not have a closed cycle and it is difficult to determine the transferred heat.

2.4.4.2. Binary Cycle Power Plant

The temperature-entropy diagram for a binary cycle power plant is shown in Figure 2.22. Starting from state 1, the expansion of saturated or superheated working fluid (wf) is given by eq. (2.65).

$$\dot{W}_t = \dot{m}_{wf}(h_1 - h_2) = \dot{m}_{wf}\eta_t(h_1 - h_{2s}) \tag{2.65}$$

Where $\dot{m}_{wf}$ is the mass flow rate of the working fluid [kg s^{-1}]. Subsequently, the working fluid is condensed with cooling water giving the energy balance in eq. (2.66).

$$\dot{m}_{cw}(h_Y - h_X) = \dot{m}_{wf}(h_2 - h_3) \tag{2.66}$$

Then the working fluid is pressurized by means of a feed pump requiring power expressed in eq. (2.67).

$$\dot{W}_p = \frac{\dot{m}_{wf}(h_{4s} - h_3)}{\eta_p} \tag{2.67}$$

The pressurized working fluid is then heated in a preheater with heat from the geothermal fluid according to the energy balance given in eq. (2.68). Where after it evaporates (and possibly gets superheated) in an evaporator according to the energy balance in eq. (2.69).

$$\dot{m}_{gf}(h_B - h_C) = \dot{m}_{wf}(h_5 - h_4) \tag{2.68}$$

$$\dot{m}_{gf}(h_A - h_B) = \dot{m}_{wf}(h_1 - h_5) \tag{2.69}$$

The power required for the cooling water pump, injection pump, condenser pump and if needed a production pump and make-up pump are calculated using eq. (2.60). The thermal and utilization efficiencies are calculated by eqs. (2.62) and (2.63), respectively.

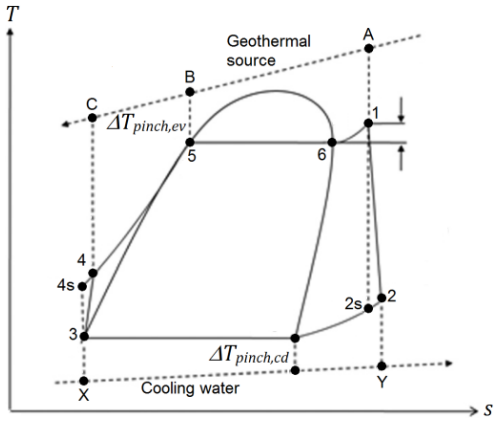


Figure 2.22: Temperature-entropy diagram for binary cycle power plants (modified from Wang et al. (2013)).

In Section 2.2.4, it has been decided to equip the binary cycle power plant with a gas lift system to lift geothermal fluids with temperatures in the range of 200 – 250 ˚C. The basic binary geothermal power plant depicted in Figure 2.3 is modified to Figure 2.23. The geothermal fluid in this study consists of H_2O, NaCl and CO_2 (Section 2.3.2). The geothermal fluid at the wellhead with gas lift is a two-phase fluid. It is assumed the gas mixture contains CO_2 and H_2O. The liquid and gas are separated in the CS. The liquid stream flows to the evaporator. The gas stream flows to a certain CO_2 production system. It is aimed for to utilize the degassed CO_2 from the production well as the lift gas by recycling it in the gas lift system. The CO_2 is then compressed (COMP) and reinjected through the gas lift valve (GLV). In that case, the gas mixture from the wellhead the CO_2 must be separated from the H_2O prior to the compression process, because H_2O would condense in the gas lift duct. The gas lift duct surrounds the production well and contains the lift gas and the GLV. This separation process has not been considered in the present study. Therefore, two extreme scenarios have been proposed. Scenario 1 assumes that the (P, T) state of the CO_2 prior to the compression process (state c1) is equal to the (P, T) state of the gas in the CS (state A). In that case, it is assumed as well that there is no work involved in separating the CO_2 from the H_2O. Scenario 2 assumes atmospheric conditions for the CO_2 fed to the compressor. In that case, the maximum amount of compression work is considered. The compression work is given by eq. (2.70).

$$\dot{W}_{comp} = \frac{\dot{m}_{GL}(h_{c2s} - h_{c1})}{\eta_{comp}} \tag{2.70}$$

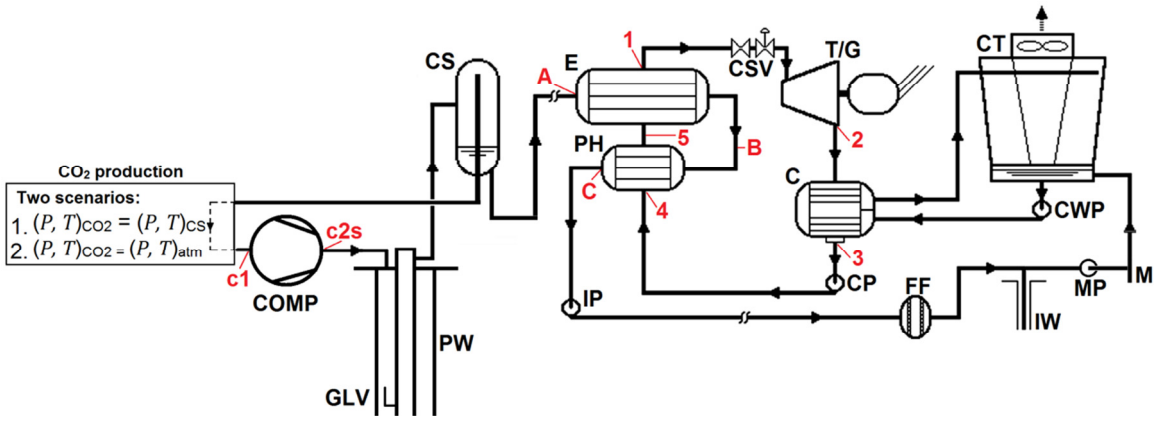


Figure 2.23: Binary cycle power plant with a gas lift system in the production well. CO_2 is separated, compressed and reinjected via the gas lift valve (GLV) (modified from DiPippo (2012)).

The decision on the working fluid has great implications for the thermodynamic performance of geothermal power plants. Candidate working fluids are propane, i-butane, n-butane, i-pentane, n-pentane and ammonia. It is aimed to find the best match between the cooling curve of the geothermal fluid and the heating-boiling curve of the working fluid to decrease the thermodynamic losses. For high temperature geothermal fluids in the range of 200 < T < 250 ˚C, i-pentane (T_c = 187.8 ˚C) is a suitable working fluids of these candidates, because of the relatively high critical temperature (DiPippo, 2012). In a previous study, i-pentane have given maximum net power in binary cycle power plants for these operating conditions (IF Technology, 2012)

2.4.4.3. Effect of NCG on Power Plant Performance & Gas Extraction Systems

Geothermal fluids contain dissolved NCG in the range of 0 – 5 wt%. The major constituent of the NCG is CO_2, which typically constitutes more than 80 wt% of the total dissolved gases. The volatility of the NCG is higher than the volatility of the steam, causing a high mass fraction of NCG at the beginning of the flashing process. When a geothermal liquid is flashed and flows into the cyclone separator (CS) (Figure 2.24), practically all NCG degas from the liquid solution. The presence of NCG in the gas mixture significantly affects the performance of the steam turbine, condenser and steam ejector/condenser. Firstly, the specific expansion work in the turbine decreases, because the recoverable specific energy of NCG is lower than that of steam. Secondly, the presence of NCG decreases the heat transfer coefficient in the condenser and consequently it increases the required heat transfer area. Subsequently, it increases the extraction and/or compression work in the gas extraction system. Additionally, higher costs due to lower heat transfer efficiency, expensive gas extraction equipment and lower turbine power output are involved with increasing NCG content. It is reported that a CO_2 mass fraction of 10 wt% can cause a reduction in net power output of 25% (Khalifa and Michaelides, 1978; Gokcen and Yildirim, 2006).

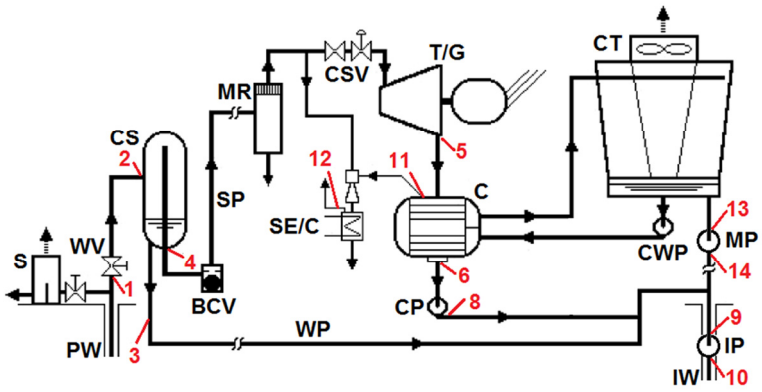


Figure 2.24: Simplified single-flash power plant schematic with state numbers (modified from DiPippo, 2012).

The geothermal gas mixture entering the gas turbine is assumed to be saturated, which is justified by the presence of moisture removers in the steam power plant. Furthermore, the assumption of an ideal gas mixture is adopted according to Dalton's Law, which is allowed due to the relatively low pressures

encountered in geothermal steam turbines and condensers. The enthalpy and entropy of the gas mixture at the entrance of the steam turbine can be calculated by eq. (2.71) and eq. (2.72), respectively. The following equations and state numbers all correspond to the single-flash steam power plant depicted in Figure 2.24.

$$h_{mix,4} = \sum_{i=1}^{n} w_i h_i(P_i, T) \tag{2.71}$$

$$s_{mix,4} = \sum_{i=1}^{n} w_i s_i(P_i, T) \tag{2.72}$$

Where i and n represent the number of a single component and the total number of components, respectively. The partial pressures P_i are determined by the mass fraction w_i and molar mass M_i of the constituents according to eq. (2.73).

$$P_i = \frac{{}^{w_i}/_{M_i}}{\sum_{i=1}^{n} {}^{w_i}/_{M_i}} P \tag{2.73}$$

Since the gas mixture is separated from the liquid in the flash tank and equilibrium between liquid and gas is assumed, the enthalpy and entropy of steam correspond to saturated conditions. In other words, the water vapor present in the gas mixture which coexists with the liquid is in equilibrium with that liquid. The water vapor exists in the gas phase at a partial pressure equal to the saturation pressure of water for the identical temperature. The isentropic expansion process is presented by eq. (2.74).

$$s_{mix,4} = s_{mix,5s} \tag{2.74}$$

Figure 2.21 shows that expansion occurs in the two-phase region, which indicates that moisture will be present in the steam turbine due to partial condensation. It is assumed that this moisture is pure H_2O. The CO_2 solubility in water can be neglected for low temperatures and low pressures. The solubility at 40 ˚C and 0.5 bar is less than 0.5 g/kg_{H2O}, where the temperature corresponds roughly to the condenser temperature. The pressure at the exit of the steam turbine can even be lower, resulting in even lower solubility (Carroll et al., 1991). The enthalpy $h_{mix,5s}$ and entropy $s_{mix,5s}$ are calculated by eq. (2.75) and eq. (2.76).

$$h_{mix,5s} = (1-\chi) h_{H_2O,6}(T) + \chi \sum_{i=1}^{n} w_i h_i(P_i, T) \tag{2.75}$$

$$s_{mix,5s} = (1-\chi) s_{H_2O,6}(T) + \chi \sum_{i=1}^{n} w_i s_i(P_i, T) \tag{2.76}$$

Where χ is the steam quality after isentropic expansion. The enthalpy $h_{mix,5}$ is calculated by eq. (2.77).

$$h_{mix,5} = h_4 - \eta_t \left(h_4 - h_{mix,5s}\right) \tag{2.77}$$

Where η_t and $\dot{W}_t$ are again determined by eq. (2.58) and eq. (2.55).

The expanded gas-liquid mixture is fed to a condenser, which is connected to a gas extraction system. There are basically three gas extraction systems used in geothermal power plant, namely the steam ejector/condenser (SE/C), liquid ring vacuum pumps (LRVP) or centrifugal compressors. Sometimes hybrid gas extraction systems are deployed. In this study, only SE/C and centrifugal compressors are considered. SE/C is relatively cheap, reliable and easy to maintain, because it does not contain moving parts. Centrifugal compressors are relatively expensive and prone to failure, because of its moving parts and the potential aggressive nature of geothermal fluids. Centrifugal compressors are utilized in geothermal power plant where high NCG concentrations are present. Centrifugal compressors are generally more efficient than SE/C.

Steam ejector/Condenser:

SE/C is a supersonic flow induction device. It sucks and compresses the NCG from the condenser by creating a vacuum with an accelerated motive flow drained from the gas stream before the steam turbine. The motive flow decreases the mass expanded in the turbine; therefore, it is not available for power generation anymore. A more comprehensive explanation on the operation of a SE/C is presented in Section A.5. The presence of NCG reduces the heat transfer efficiency of the condenser; therefore it requires an increased heat transfer area. Furthermore, a decrease in turbine outlet pressure, which increases turbine power output, decreases the temperature difference between the mixture and the available cooling medium in the condenser. Additionally, the condensation of the mixture does not proceed isothermally, because of the NCG presence. This results also in an increased required heat transfer area or an increased cooling medium mass flow rate, which both increase the capital and/or operating cost. Moreover, a reduced temperature difference in the condenser decreases the subcooling of the gas mixture, which on its turn increases the necessary motive flow to the SE/C. The availability of the cooling medium depends mainly on the setting of the power plant and can be assumed as a constant. In summary, it can be concluded that decreasing the back pressure of the turbine will increase the mass flow rate of motive flow. There should be found a balance for the optimum power output. The net power output given in eq. (2.61) must be complemented with the power input of the make-up pump $\dot{W}_{mp}$ as in eq. (2.78) to make up the extracted fluids.

$$\dot{W}_{net} = \dot{W}_e - \dot{W}_{ip} - \dot{W}_{cp} - \dot{W}_{cwp} - \dot{W}_{mp} \tag{2.78}$$

The thermodynamic performance may exhibit an optimum at an intermediate turbine back pressure. Furthermore, the economic optimum depends also on the capital expenditure associated with the optimum turbine back pressure, which can deviate from that for the thermodynamic optimum. The loss of turbine power output depends on the mass flow fed to the SE/C, the compression ratio and the composition of the mixture. The partial pressure of H_2O at the inlet of the SE/C (state 11) can be determined by eq. (2.79).

$$P_{H_2O,11} = P_{v,H_2O}(T_{11}) \tag{2.79}$$

The remaining partial pressures of the NCG and subsequently the mole fractions can be calculated with the mass fractions entering the condenser, the molar masses and the assumption that NCG do not dissolve in the liquid so that it satisfies eq. (2.80).

$$P = \sum_{i=1}^{n,NCG} \frac{{}^{w_i}/_{M_i}}{\sum_{i=1}^{n,NCG} {}^{w_i}/_{M_i}} \left(P - P_{H_2O,11}\right) + P_{H_2O,11} \tag{2.80}$$

Now according to the conservation of mass, the mass flow (suction flow) through the SE/C per unit mass of mixture entering the condenser is calculated by eq. (2.81).

$$\dot{m}_{11} = \frac{w_{i,5}}{w_{i,11}} \dot{m}_5 \tag{2.81}$$

The enthalpy and entropy of the mixture at the inlet of the SE/C (state 11) is determined accordingly to eq. (2.71) and eq. (2.72).

Centrifugal compressor:

In case of a centrifugal compressor, the NCG are extracted from the condenser and compressed to atmospheric conditions in order to remove the NCG from the geothermal power plant. In Figure 2.24 the SE/C is replaced by a centrifugal compressor in this case. The drained flow to the SE/C is not required with a centrifugal compressor. The power demanded by the centrifugal compressor is given by eq. (2.82).

$$\dot{W}_{comp} = \frac{\dot{m}_{11}\left(h_{mix,11} - h_{mix,atm}\right)}{\eta_{comp}} \tag{2.82}$$

3

MODEL DESCRIPTION

The present chapter illustrates the model description. The system borders and the relevant input and output parameters are determined for every sub model. Subsequently, relevant phenomena and assumptions are outlined. Then the calculation procedure and implementation, based on the conservation laws and constitutive equations treated in Chapter 2, is discussed. In Section 3.1 the general model is introduced. Other sections comprise the following sub models: 3.2 geothermal fluid property model, 3.3 reservoir model, 3.4 production well model – self-flowing, 3.5 drift-flux model, 3.6 single-flash power plant model, 3.7 injection well model, 3.8 production well model – gas lift and lastly in 3.9 the binary cycle power plant model.

3.1. General Model

3.1.1. Modeling Purpose

The purpose of the model is to study and compare geothermal power plant performance for two power plant designs. This is realized by simulating and studying the hydraulic and thermodynamic behavior of the geothermal fluid in a geothermal power plant system during steady state operation with the ability for off-design simulation. The single-flash power plant comprises a reservoir, a self-flowing production well, a single-flash power plant and an injection well. The binary cycle power plant comprises a reservoir, a production well with gas lift and an injection well.

3.1.2. System Border and I/O Variables

Figure 3.1 shows a schematic of the total mathematical model including the two geothermal power plants. The total model can be divided basically into nine sub models: reservoir model, production well model – self-flowing, production well model – gas lift, injection well model – single-flash, injection well model – binary cycle, single-flash power plant model, binary cycle power plant model, geothermal fluid property (GFP) model and drift-flux model. Even though, the real physical system consists of the reservoir, two production wells, two power plants and two injection wells. The GFP model and the drift-flux model have their own sub model, because of their complex nature. Black arrows within the sub models represent the flow direction of the geothermal fluid. The green and red arrow(s) indicate the input and output variables of the total system. Gray arrows connecting the sub models indicate the input and output variables between the sub models. There are two sub models for the production well: self-flowing and gas lift. There are two sub models for the geothermal power plant: single-flash and binary cycle. The production well model – self-flowing is a part of the single-flash power plant model and production well model – gas lift is a part of the binary cycle power plant model.

Model boundaries are the far-field reservoir for the reservoir, where reservoir properties are assumed constant. Additionally, for the power plant model every device that consumes or produces power, e.g. pumps and generator have a boundary between the device and the grid. Finally, equipment where mass flows to the environment, e.g. the condenser/cooling tower and the (SE/C)/environment interface, is a model boundary.

The well/rock interface for the production wells and injection wells, where heat flows across the boundary, has not been defined as a system boundary. An explanation is given in Section 3.4.1.

The input parameters must be set prior to simulation in order to compute the output parameters at the end. Additionally, a GFP model for the geothermal fluid was implemented that can communicate with the surrounding sub models. The production well models interact with the drift-flux model in order to calculate the void fraction of the fluid flow in the production wells if two-phase flow is present.

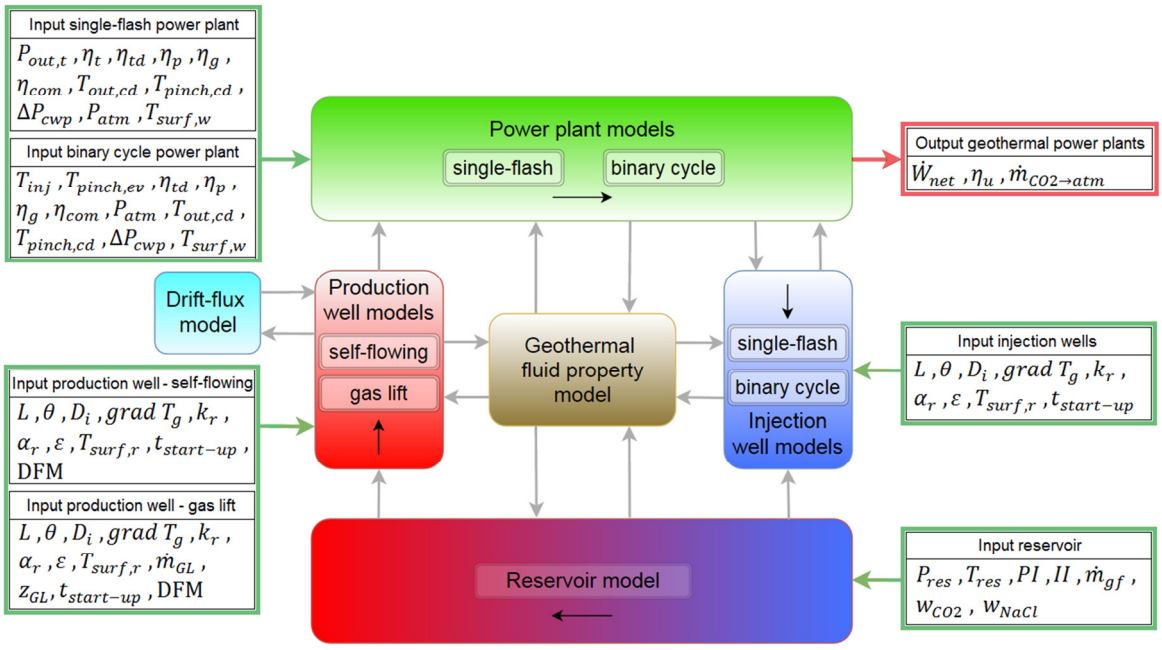


Figure 3.1. Schematic representation of the total mathematical model of the geothermal power plant systems. Black arrows indicate flow direction. Green and red arrows indicate input and output model parameters of the total system, respectively. Gray arrows show the interaction and calculation direction of the sub models. The production well model – self-flowing is a part of the single-flash power plant model and the production well model – gas lift is a part of the binary cycle power plant model.

3.1.3. Calculation Procedure

The reservoir model, production well models, injection well models, power plant models and drift-flux model were developed within the MATLAB environment. The core of the GFP model is a VBA MS Excel model (GFP Excel model) developed in Francke et al. (2013). In the present work, a MATLAB model was developed around the GFP Excel model to calculate the fluid properties. The MATLAB model communicates with the GFP Excel model by creating a component object model (COM) server in order to give input and extract output from the GFP Excel model.

Figure 3.2 presents the calculation procedure of the general model. The MATLAB model is a function based model. Every declared function accepts declared inputs from the previous function and returns demanded outputs for the successive function. The main procedure is briefly described below in the order of simulation.

Start: This is the main script to perform a simulation of a geothermal power plant system containing a reservoir, production well, power plant and injection well. This script contains twelve functions, which are completed in sequence.

The white blocks correspond to auxiliary functions necessary for preparing the simulation. These blocks have not been explained in the present chapter, because it is not linked to the theory discussed in Chapter 2. For the MATLAB code of all these function is referred to Section B.

The twelve yellow blocks correspond to the nine sub models, including four functions for the GFP MATLAB model: reservoir-, two production well-, two power plant-, injection well-, drift-flux- (*fCalc_eps_g*) and GFP MATLAB model(s) (*fCalc_geofprops1, fCalc_geofprops2, fCalc_geofprops3, fCalc_geofprops4*). These functions are explicitly described in the next sections.

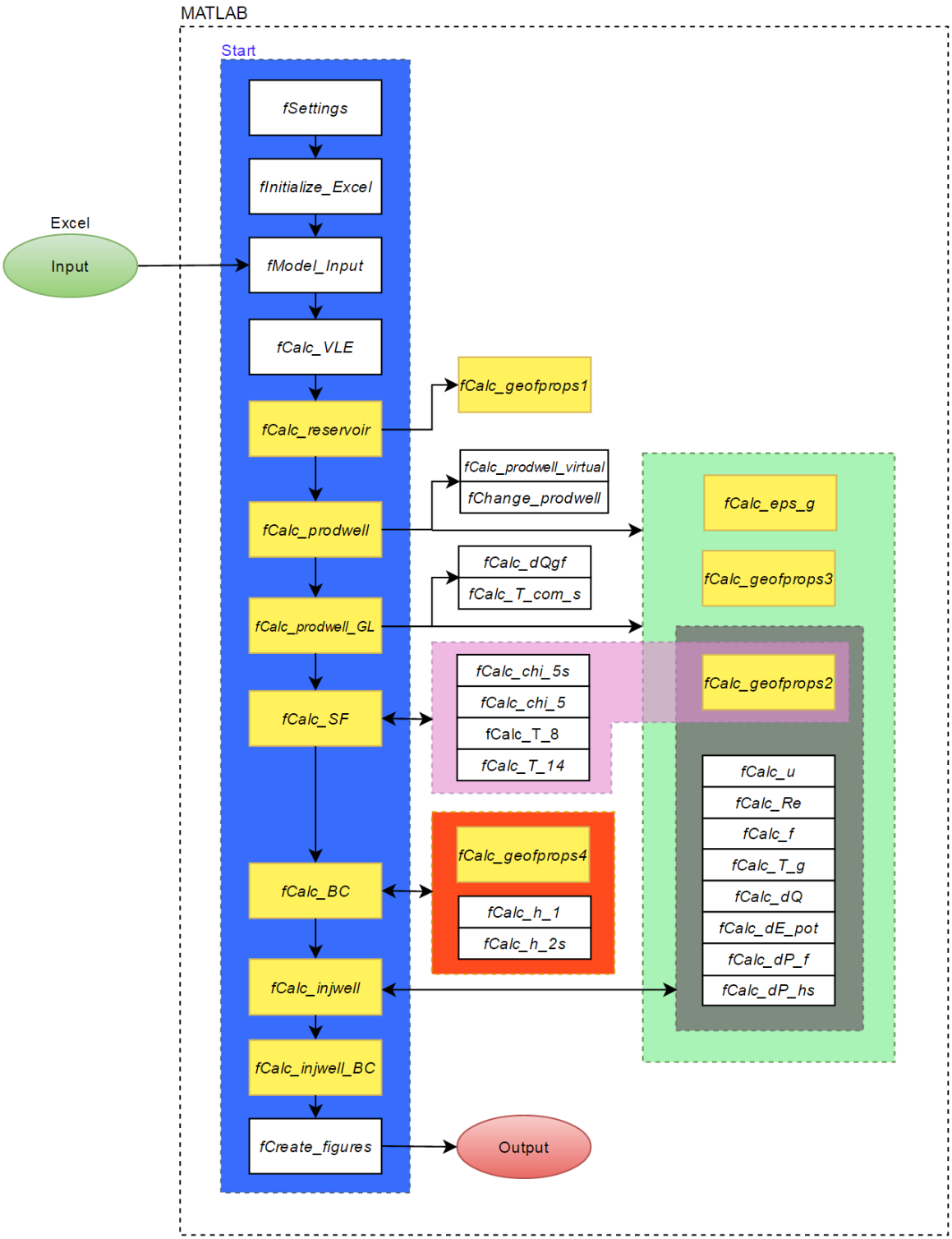


Figure 3.2: Calculation procedure of the general model (MATLAB). Start (blue) is the main script. The yellow blocks correspond to the nine sub models: reservoir, two production wells, two power plants, two injection wells, drift-flux (*fCalc_eps_g*) and four GFP MATLAB models (*fCalc_geofprops1, fCalc_geofprops2, fCalc_geofprops3, fCalc_geofprops4*). The Green block contains functions invoked by production well models. Purple block contains functions invoked by single-flash power plant model. Red block contains function invoked by binary cycle power plant model. Grey block contains functions invoked by injection well model.

fSettings: Variables necessary for computation are declared, e.g. data tables, constants and auxiliary model parameters (iteration parameters).

fInitialize_Excel: The GFP Excel model is initialized. All other Excel workbooks are closed.

fModel_Input: MATLAB draws all user-defined input from an Excel workbook. The user-defined input variables have been given in Figure 3.1. The Excel workbook interface is shown in Section C.1.

fCalc_VLE: The fluid properties at the VLE curve are calculated. The importance of this function is outlined in bullet point 2 in Section 3.2.2.2.

fCalc_reservoir: The geothermal fluid properties at inlet and outlet of the production well and injection well are calculated, respectively (Section 3.3).

GFP MATLAB model is invoked for the first time: *fCalc_geofprops(1, 2, 3, 4)*: these functions correspond to the GFP model. The numbers correspond to different algorithms, which was necessary for different functions. Section 3.2 discusses the GFP model explicitly.

fCalc_prodwell: The fluid properties in the self-flowing production well are calculated (Section 3.4).

fCalc_eps_g: this function corresponds to the drift-flux model and calculates the void fraction (Section 3.5).

fCalc_prodwell_GL: The fluid properties in the production well with gas lift are calculated (Section 3.8).

fCalc_SF: The fluid properties in the single-flash power plant are calculated. Additionally, power plant performance is calculated (Section 3.6).

fCalc_BC: The fluid properties in the binary cycle power plant are calculated. Additionally, power plant performance is calculated (Section 3.9).

fCalc_injwell: The fluid properties in the injection well are calculated in case of a single-flash power plant (Section 3.7).

fCalc_injwell_BC: The fluid properties in the injection well are calculated in case of a binary cycle power plant (Section 3.7).

fCreate_figures: Relevant output is structured and relevant graphical figures are created.

3.2. Geothermal Fluid Property Model

The GFP model has been based on the "Geofluid Model" of Francke et al. (2013), referred to as the "GFP Excel model". For an exact description of the GFP Excel model is referred to Francke et al. (2013) or Francke (2014). The GFP Excel model has been developed in VBA. The input and output is set and obtained from MS Excel, respectively. Figure C.4 presents the interface of the two-phase sheet. Figure C.5 and Figure C.6 show the liquid phase and vapor phase sheet, respectively. In the present work a MATLAB model, referred to as the "GFP MATLAB model", has been developed around the GFP Excel model to calculate the fluid properties. For the assumptions related to the GFP Excel model is referred to Section 2.3.4.2. Additional assumptions are discussed in Section 3.2.2. The GFP MATLAB model can be invoked by four functions: *fCalc_geofprops1*, *fCalc_geofprops2*, *fCalc_geofprops3 and fCalc_geofprops4*.

3.2.1. Purpose and System Border

The purpose of the GFP MATLAB model is to calculate the liquid phase or two-phase properties of the geothermal fluid.

The GFP MATLAB model is not a real physical system. Therefore, it does not contain a real physical system border. The GFP MATLAB model can be invoked by the other surrounding sub models (see Figure 3.1). Input and output variables are exchanged with these sub models.

3.2.2. Model Development and Assumptions

The fluid properties obtained from the GFP Excel model have been found valid for the entire liquid phase. Additionally, the GFP Excel model is valid from a pressure where a significant gas mass fraction starts to develop until the saturation pressure of pure water, associated to the relevant state temperature. Nevertheless, the GFP Excel model also has counted some discontinuities, which can be critical in the current numerical models of the production wells. The shortcomings of the GFP Excel model are described in Section 3.2.2.1. The solutions and assumptions following these shortcomings are explained in Section 3.2.2.2.

3.2.2.1. Shortcomings GFP Excel Model

1. It has been noted that there is a significant difference between the degassing pressures obtained from the online model of Duan and Sun (2003) and GFP Excel model of Francke et al. (2013). In their study, Duan and Sun (2003) validated the degassing pressures with experimental results. On the other hand, the GFP Excel model has shown discontinuities between the degassing pressure of Duan and Sun (2003) and their degassing pressure. It is assumed that the behavior of the geothermal fluid has not been approximated accurately by the GFP Excel model for this particular region.
2. For pure water, the GFP Excel model has been unable to calculate fluid properties below saturation pressures.
3. In addition to shortcoming 2, the GFP Excel model has not accurately approximated the fluid properties for pressures between the saturation pressure of water and the saturation pressure of NaCl(aq). As it is explained in Section A.2.1, the saturation pressure of a NaCl(aq) is lower than that of pure water. The sections of the production well, which experience pressures between these two saturation pressures have shown discontinuities.
4. For low CO_2 mass fractions dissolved in the geothermal fluid, the quality jumps to 1 even before the saturation pressure is reached. This has given large discontinuities in the GFP Excel model.

Consequently, these shortcomings gave rise to discontinuities and errors in the production well models. The discontinuities encountered in the GFP Excel model have all been related to the quality/gas mass fraction. The solutions and assumptions, described in Section 3.2.2.2, for these shortcomings are obtained by manipulating the vapor quality. Subsequently, the associated fluid properties are calculated.

3.2.2.2. Solutions and Assumptions

The solutions and assumptions to the shortcomings (Section 3.2.2.1) of the GFP Excel model are discussed in this section. The region close to the VLE curve for a pure substance is already quite delicate. Similarly, in the numerical model of the production wells, this involves many iteration steps. The existence of NaCl and CO_2 makes it even more prone to errors and discontinuities. The described solutions are graphically clarified in Figure 3.3 and Figure 3.4. The former shows a pressure-enthalpy diagram of a ternary H_2O – NaCl – CO_2 solution with $m_{NaCl} = 3$ mol kg^{-1} and $m_{CO2} = 1.41$ mol kg^{-1}, which is approximately equivalent to a NaCl mass fraction of 0.15 kg kg^{-1} and a CO_2 mass fraction of 0.05 kg kg^{-1}. The latter zooms in on the northwest part of Figure 3.3, to the bubble curve. The solutions are explained on the basis of the 200 ˚C isotherm, depicted as the continuous red line. Generally, this description applies to all temperatures and molalities within the validity ranges given in Table 2.5.

1. The vapor quality of the geothermal fluid in the sections of the wellbore, where the pressure is between the degassing pressure of Duan and Sun (2003) and the degassing pressure calculated in the GFP Excel model is computed by interpolation.

   This particular situation has been enlarged in Figure 3.4. The dashed magenta line represents the degassing pressures between 150 ˚C and 250 ˚C according to the GFP Excel model for the indicated molalities. The dashed black dashed line represents the degassing pressures between 170 ˚C and 250 ˚C according to Duan and Sun (2003). In the GFP Excel model degassing starts at 310 bar, while according to Duan and Sun (2003) degassing starts at 635 bar. On the other hand, the GFP Excel model has shown a sharp increase in quality from 0 to 0.016 at that particular degassing pressure. This is reflected in the sudden enthalpy change on the 200 ˚C isotherm close to the degassing pressure of the GFP Excel model. At that same pressure, Duan and Sun (2003) have shown a quality of 0.017, which is equivalent to an absolute error of approximately 0.1%. Therefore, it is justified to interpolate the quality between the degassing pressure of Duan and Sun (2003) and the GFP Excel model. This

interpolation of quality is represented by the dashed cyan line. The associated fluid properties are calculated with the mass-weighted average of the quality and the single phase fluid properties.

The algorithm was coded in the function *fCalc_geofprops3*. It is invoked from the production well model (*fCalc_prodwell*).

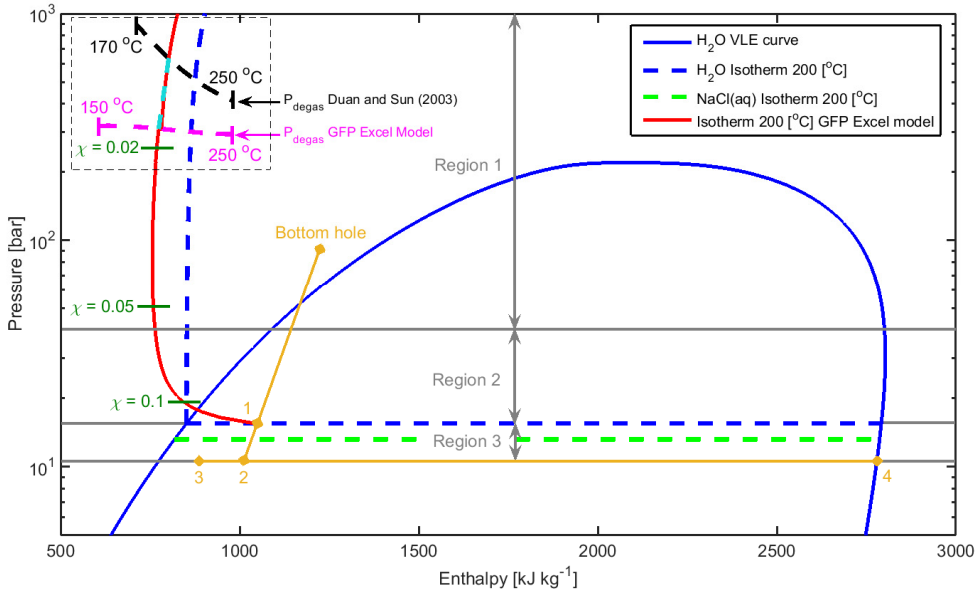


Figure 3.3: Pressure-enthalpy diagram of H_2O – NaCl – CO_2 solution with $m_{NaCl} = 3$ mol kg^{-1} and $m_{CO2} = 1.41$ mol kg^{-1} consisting of the VLE curve and the 200 ˚C isotherm of water (blue (dashed)), the 200 ˚C isotherm of NaCl(aq) (green dashed), the 200 ˚C isotherm of the GFP Excel model (red), degassing pressure GFP Excel model (magenta dashed), degassing pressure isotherm Duan and Sun (2003) (black dashed). Yellow line from bottom hole to 2 is a hypothetical pressure-enhalpy profile in a production well.

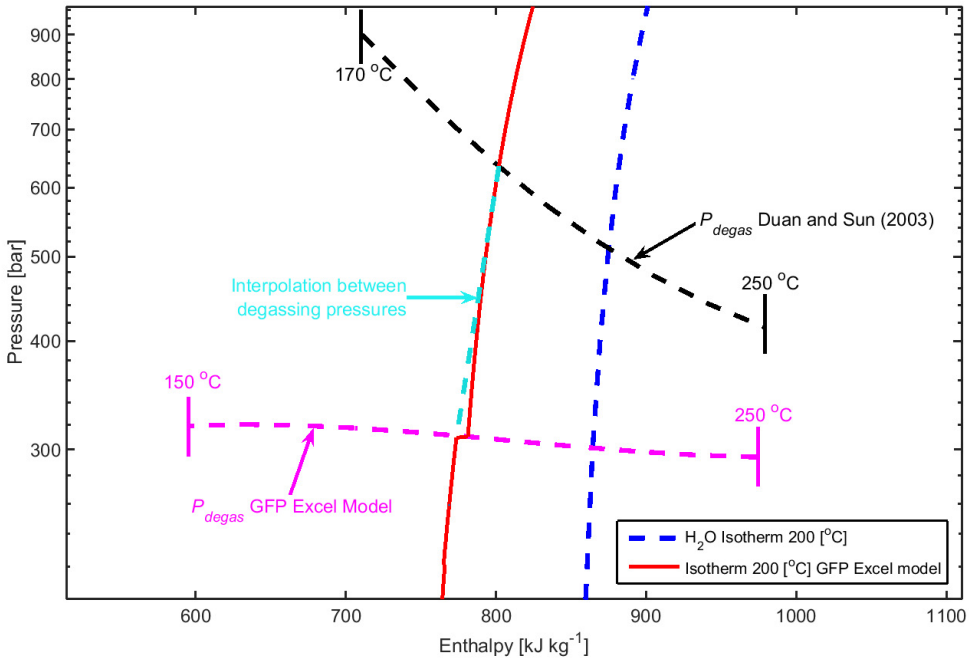


Figure 3.4: Enlargement of the degassing region from Figure 3.3. The magenta dashed line shows the interpolation between the degassing pressure according to Duan and Sun (2003) (black dashed) and Francke et al. (2013) (red dashed).

2/3. The descriptions of the solutions to shortcomings 2 and 3 in Section 3.2.2.1 are joined here. The dashed blue line in Figure 3.3 represents the 200 ˚C isotherm of pure H_2O. The continuous blue line represents the VLE curve of pure H_2O. The dashed green line represents the 200 ˚C of the NaCl(aq) solution for the indicated molality, without CO_2. The continuous yellow line following the points from the bottom hole to 1 and finally 2 represents arbitrary geothermal fluid behavior in a production well. It catches the trend of decreasing pressure and enthalpy while flowing from bottom to top. It can happen, at a certain location in the production well, the pressure falls below the saturation pressure of H_2O for the associated temperature. This is indicated by the step from 1 to 2. The iteration fails at that moment, because the GFP Excel model cannot calculate fluid properties below H_2O saturation. At this particular moment, the properties of the geothermal fluid as a function of P and T coinciding with the VLE curve come into play. These points are indicated by 3 and 4. Point 3 coincides with the isotherm of pure water and the geothermal fluid isotherm for that particular pressure at point 2. The vapor quality at point 2 in the two-phase region is then obtained by interpolation of enthalpies between the fluid properties at points 3 and 4.

   The algorithm was coded in the function *fCalc_geofprops2*. It is invoked from *fCalc_prodwell*, *fCalc_prodwell_GL*, *fCalc_SF*, *fCalc_BC*, *fCalc_injwell* and *fCalc_injwell_BC*. The MATLAB code can be found in Section B.

4. If 4 from Section 3.2.2.1 occurs, the GFP MATLAB model should be able to calculate the right properties. Therefore, several data points for the quality as a function of pressure, temperature and composition have been obtained from Duan and Sun (2003) in order to compute the quality close to saturation pressures. The fluid properties at point 3 in Figure 3.3 are then first interpolated from these data tables.

Additional assumption:

1. In case the pressure falls below the saturation pressure of water and there is CO_2 present in the gas phase, it has been assumed that 100% CO_2 degasses from the liquid phase. It has been verified by Francke et al. (2013) and Duan and Sun (2003) that > 99% of CO_2 degasses at the H_2O saturation pressure, independent of NaCl molality.
2. It has been assumed that evaporation from the saturation pressure of water happens almost isobarically. Therefore, it is linearly interpolated between a quality of 0.20 and 1. This is justified, because the amount of CO_2 degassed from the liquid solution at saturation pressure of water, corresponding to a quality of 0.20, is 99.4% according to Francke et al. (2013). This shows almost all CO_2 is located in the gas phase. This phenomenon has been discussed already in Section 2.3.2.

3.2.3. Calculation Procedure

The present section relates to *fCalc_geofprops2*. It is the most complex function of the four GFP MATLAB sub models, because two-phase geothermal fluid can be involved. The MATLAB code of the functions can be found in Section B.

The flow diagram of the calculation procedure for the geothermal fluid properties is presented in Figure 3.5. The GFP Excel model has been restricted to an input of pressure, temperature and composition. However, the input of the GFP MATLAB model in the present work consists of $P(i)$, $h(i)$, w_{NaCl}, w_{CO2} and $T(i-1)$, where (i) represents the segment number. The production well is divided in segments in the production well model. This will be discussed in Section 3.4. Because the production well model in the present work calculates the pressure and enthalpy at the inlet of a well segment, the corresponding temperature in that segment must be iterated to find a solution. Iteration starts with the temperature of the previous segment $T(i-1)$.

The output of the GFP MATLAB model consists of $T, \chi, \varepsilon_g, v, \rho, c_{p,} \mu, w_{NaCl,l}$, $w_{CO2,l}$, $w_{H2O,l}$, $w_{CO2,g}$, $w_{H2O,g}$ for segment i. Basically, the GFP model can be divided into three parts, with three different algorithms. These parts correspond to three different regions. The regions are marked in gray in Figure 3.3.

1. This region and corresponding algorithm applies to all pressures in the range of 40-1000 bar and temperatures below 250 ˚C. This is the maximum temperature for which the model is still valid. The lower limit of 40 bar corresponds to the saturation pressure at 250 ˚C. The absolute enthalpy error of 0.01 kJ/kg is the default value. Increasing the absolute error will decrease the number of iterations and

computation time, but increases the error of calculated outputs and vice versa. Iteration is performed with the MATLAB function fsolve. Safety measures have been incorporated in the model of the present work if the iteration fails, which is a possible scenario, because the GFP Excel model can show discontinuities. The temperature is then increased or decreased depending on the trend of the enthalpy.

2. A pressure below 40 bar and below T_{sat}, where the GFP Excel model is still valid. In this scenario iteration is performed by adapting the temperature depending on the trend of the enthalpy. The fsolve function is removed, because iteration fails close to the saturation pressure of water. The default temperature step is 0.1 ˚C.

3. A pressure below 40 bar and above T_{sat}, where the GFP Excel model has not been able to calculate the fluid properties. Above T_{sat}, the fluid properties are calculated as described in point 2/3 in Section 3.2.2.2. If $T(i-1) > T_{sat} - 0.05$, then it is assumed that $T(i) = T_{sat} - 0.05$. The 0.05 is the default safety margin to avoid scenario's in which $T(i)$ is above the saturation temperature and the GFP Excel model shows large discontinuities.

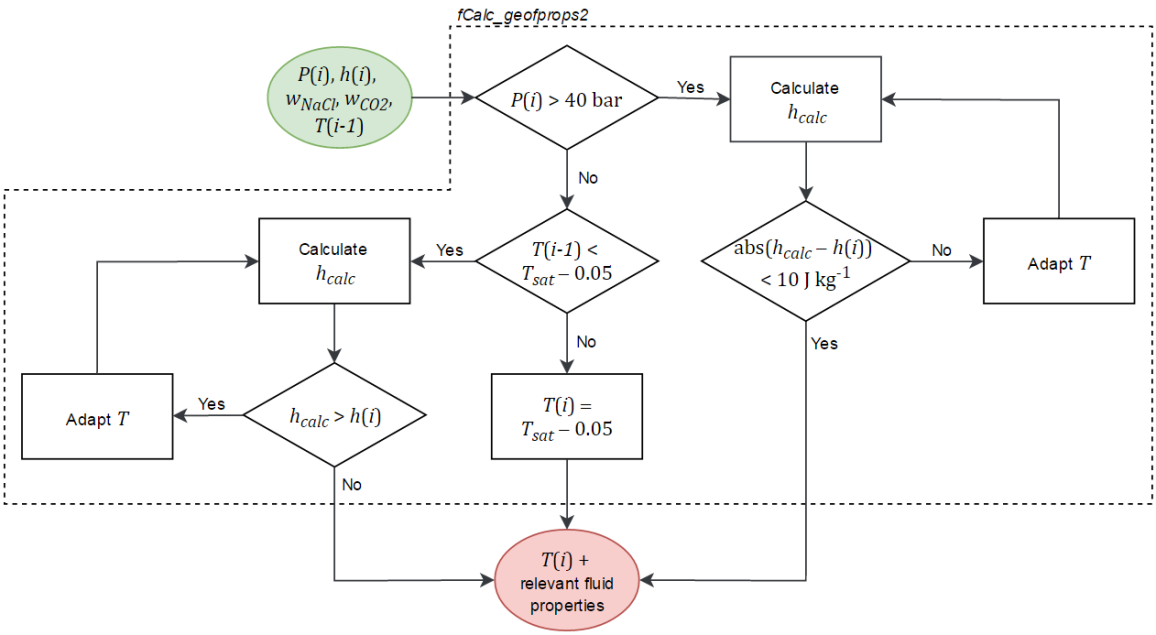


Figure 3.5: Calculation procedure geothermal fluid properties *fCalc_geofprops2* with input variables (green) and output variables (red).

3.3. Reservoir Model

3.3.1. Purpose and System Border

The purpose of the reservoir model is to calculate the geothermal fluid properties at the inlet of the production well and at the outlet of the injection well.

The system borders of the reservoir are the outflow to the production well model, the inflow from the injection well model and the reservoir itself.

3.3.2. Phenomena and Assumptions

For more extensive description of phenomena and assumptions related to reservoir and reservoir flow characteristics is referred to Section 2.4.1. The most relevant assumptions that have been made, are: isotropic permeability, homogeneous rock properties, fully radial flow and saturated rock pores with a single phase fluid. The pressure drawdown from the far-field reservoir can be described by PI and II.

The geothermal fluid inside the reservoir is a ternary system containing H_2O, NaCl and CO_2. According to Duan and Sun (2003), these species are the most common in geothermal fluids. For justification and information on the chemical composition is referred to Section 2.3.2.

Additional assumptions:

1. The temperature of the far-field, referred to as T_{res}, is assumed to be the maximum existing temperature in the reservoir. It means that T_{res} is equivalent to the production well inlet temperature.
2. The pressure of the far-field referred to as P_{res} is assumed to be the hydrostatic pressure of the reservoir if flow is not induced.
3. The geothermal fluid is always present in liquid phase alone in the reservoir. Pressure drawdown should not result in degassing of CO_2 inside the reservoir.
4. The composition of the geothermal fluid is constant.

3.3.3. Calculation Procedure

The flow diagram of the calculation procedure for the reservoir model is presented in Figure 3.6. Inputs to the sub model are defined by the user to the left. The sub model calculates pressure drawdown according to eq. (2.15) and eq. (2.16). Subsequently, it exports output to and imports input from the GFP MATLAB model. Finally, it is checked if the liquid-only state is satisfied. If this is true, output is sent to the production well model and the injection well model. If not, the simulation will be terminated.

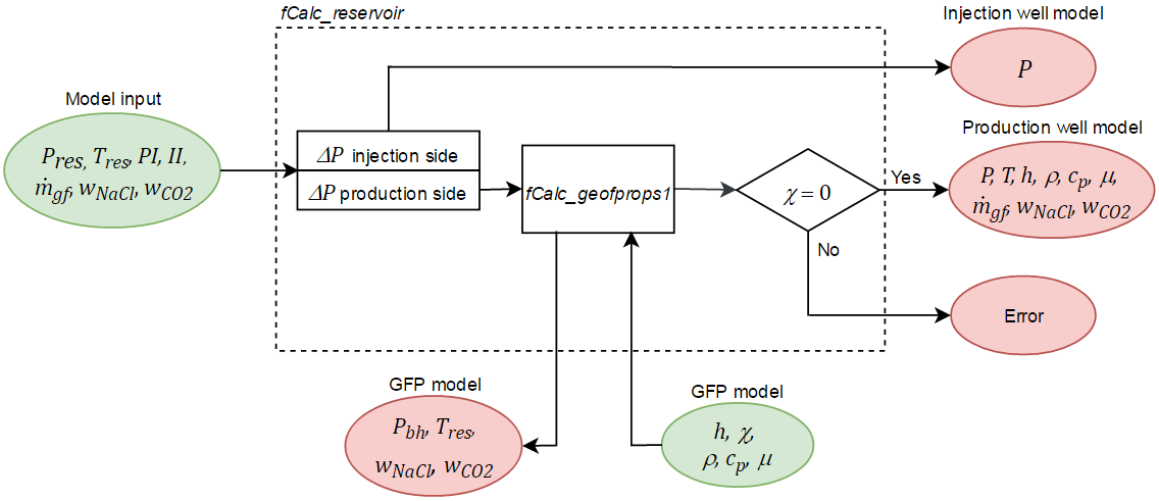


Figure 3.6: Calculation procedure for the reservoir model with system border (black dashed), input variables (green) and output variables (red).

3.4. Production Well Model – Self-Flowing

3.4.1. Purpose and System Border

The purpose of the production well model – self-flowing is to study and simulate the hydraulic and thermal behavior of the geothermal fluid within the well in steady-state conditions. Figure 3.7 presents a schematic of the production well. It is a one-dimensional numerical model, where segments are distributed along the length of the well.

At the bottom the system border is the inflow from the reservoir model. At the top the system border is the outflow to the single-flash power plant model. Additionally, the GFP MATLAB model/production well model interface and drift-flux model/production well model interface are system borders. As one would perhaps expect, the well/rock interface is not defined as a system border. The heat flow rate $\dot{Q}$ is described

by an analytical equation given in eq. (2.18). The geothermal rock, surrounding the production well, is part of the production well and discretized as such. The rock formation is radially assumed to be infinite. The assumptions and the calculation procedure related to this model are discussed in Section 3.4.2 and 3.4.3, respectively.

3.4.2. Phenomena and Assumptions

1. At the bottom of the production well, referred to as bottom hole, geothermal fluid in liquid state flows in. At the top of the production well, referred to as wellhead, geothermal fluid flows out of the well in liquid or two-phase state.
2. The geothermal fluid always flows in one direction, which is upward. Liquid and vapor flows co-currently. Exception is there for slug flow, where liquid film flows downward between the Taylor bubble and the pipe wall, this phenomenon is accounted for in the drift-flux model (see Section 3.5).
3. The flow is modeled one-dimensionally and stationary.
4. There is no mass, momentum and energy accumulation.
5. All fluid properties are being modeled homogeneous for liquid and two-phase flow. This means uniform properties on a cross section. Exception is made for the velocity of the gas phase and liquid phase in two-phase flow, where slip must be accounted for (see Section 3.5).
6. The flow is fully developed. The velocity profile is fully developed.
7. The flow is turbulent.
8. The production well is divided into segments. Every segment is radially symmetric, with a constant geometry, constant cross-section, fixed volume and constant wall roughness.
9. Heat flows only radially and it is governed by conduction.
10. Along with the production well segments, the geothermal rock formation is divided in segments of equivalent length (see Figure 3.7). The rock formation is also radially symmetric and has constant properties for the length of the segment.
11. No work is exerted on the geothermal fluid.
12. Liquid flow is assumed to be incompressible.
13. No chemical reactions occur in the geothermal fluid.
14. Rock formation is radially infinite.

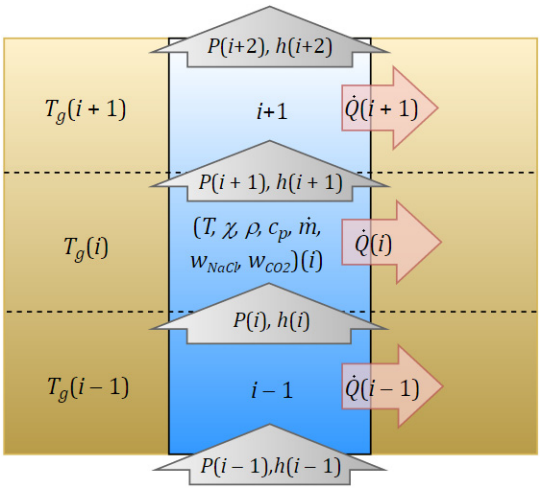


Figure 3.7: Schematic of the production well model. Wellbore and rock formation is divided into multiple segments $(i-1, i, i+1)$.

3.4.3. Calculation Procedure

3.4.3.1. Numerical Model

The analytical approach in wellbore modeling, as in eq. (2.27) and (2.28) for liquid flow and two-phase flow respectively, can give rise to significant errors. With liquid flow this is caused by the temperature dependent density. For two-phase flow, density is also dependent on void fraction, which makes it even more sensitive to errors above the flash point. As stated in Section 3.4.2 the production well is axially discretized. The length of the segments is user-defined, the default value is approximately in the range of 20 – 25 m adopted from Francke (2014). The calculation of the segment properties is performed by applying a forward finite difference scheme. It means that segment properties are constant and equivalent to the inflow conditions ($P, h, \dot{m}\ and\ w$) of the respective segment (i) and the outflow conditions of the previous segment ($i-1$) (see Figure 3.7). The inflow pressure, enthalpy and component mass fractions are known, from which thermodynamic and transport properties of the geothermal fluid are being calculated with the GFP MATLAB model discussed in Section 3.2. In addition, the incoming mass flow rate, segment geometry and rock formation properties are known, from which the relevant variables are calculated to solve the conservation laws and constitutive equations for the respective segment (i) and calculate the outflow conditions of that segment and inflow conditions of the next segment ($i+1$).

3.4.3.2. Model Equations

The conservation laws discussed in Section 2.4.2 are applied to the segments of the production well.

Mass balance:

$$\dot{m}(i) = \dot{m}(i-1) \tag{3.1}$$

Momentum balance:

The momentum balance given by eq. (2.27) for liquid-only flow and eq. (2.29) for two-phase flow is rewritten for a segment by eq. (3.2) and eq. (3.3) for liquid-only flow and two-phase flow. In eq. (3.2) the kinetic part is neglected, because the liquid is assumed incompressible.

$$P(i) = P(i-1) - \left[\frac{f\rho u^2 L}{2D_i}\right](i-1) - [g\rho L\cos\theta](i-1) \tag{3.2}$$

In eq. (3.3), the kinetic contribution to the pressure loss, caused by the acceleration of the fluid due to degassing/evaporation, is taken into account. It can be seen that eq. (3.3) is implicit. Therefore, iterations are necessary to solve the momentum balance.

$$P(i) = P(i-1) - \rho(i-1)\left(u^2(i) - u^2(i-1)\right) - \left[\frac{f\rho u^2 L}{2D_i}\right](i-1) - [g\rho L\cos\theta](i-1) \tag{3.3}$$

Energy balance:

For the energy balance, the same applies as for the momentum balance. The energy balance given by eq. (2.17) is rewritten for a segment in eq. (3.4) for liquid-only flow and to eq. (3.5) for two-phase flow.

$$h(i) = h(i-1) + \left[\frac{\dot{Q}}{\dot{m}}\right](i-1) - [gL\cos\theta](i-1) \tag{3.4}$$

Eq. (3.5) is implicit, which also makes an iterative procedure necessary to solve the equation.

$$h(i) = h(i-1) + \left[\frac{\dot{Q}}{\dot{m}}\right](i-1) - \frac{1}{2}\left(u^2(i) - u^2(i-1)\right) - [gL\cos\theta](i-1) \tag{3.5}$$

The constitutive equations necessary to solve the conservation equations are based on the effective properties derived from the single phase fluid properties and the quality. The friction factor f, the Reynolds

number Re and the velocity u for liquid-only flow are calculated according to eq. (2.24), eq. (2.25) and eq. (2.26), respectively. T_g depends on the geothermal gradient, which is a user-defined variable in the model input. The velocity u, the friction factor f and the Reynolds number Re for two-phase flow are calculated according to eqs. (2.31), (2.34) and (2.35), respectively. The heat flow rate $\dot{Q}$ for every single segment is calculated by eq. (3.6) derived from eq. (2.18).

$$\dot{Q}(i) = \left[\frac{4\pi k_r\left(T_g - T_{gf}\right)}{\ln\left(\frac{4\alpha_r t}{\gamma r_w^2}\right)}\right](i) \tag{3.6}$$

3.4.3.3. Calculation Procedure

Figure 3.8 shows the calculation procedure for the production well model – self-flowing. The starting input variables (left) are obtained from the output variables of the reservoir model at the production well side and the user-defined variables from the model input. Then segment geothermal fluid properties are calculated from the bottom to the top of the production well with *fCalc_geofprops2,* described in Section 3.2. Subsequently, the quality of the fluid is checked. If the condition of $\chi = 0$ is met, calculation proceeds with the constitutive equations. If $\chi > 0$, *fCalc_eps_g* is invoked to export output and import input (right top) from the drift-flux model to calculate the flow pattern *FP* and the void fraction ε_g. Consequently, an adapted density ρ is calculated before the constitutive equations are solved. The model description of the drift-flux model is discussed in Section 3.5. Then the heat flow rate $d\dot{Q}$, potential energy loss $d\dot{Q}_{pot}$, kinetic energy loss $d\dot{E}_k$, frictional pressure loss dP_f, hydrostatic pressure loss dP_{hs} and kinetic pressure loss dP_k are computed. Finally, the conservation equations are solved to obtain pressure P and enthalpy h at the outflow of the respective segment and the inflow of the next segment. These simulations are looped until the last segment and the output variables (right) are known, which are the input variables for the single-flash power plant model.

Figure 3.8 shows an alternative loop, which has to do with the discrepancy between the degassing pressures of the GFP Excel Model and Duan and Sun (2003), which has been discussed in Section 3.2. It basically involves an interpolation of the quality between these two defined degassing pressures, discussed in Section 3.2.2. The degassing pressures of Duan and Sun (2003) were implemented in MATLAB as data tables. Figure C.7 – Figure C.10 show the degassing pressures as a function of T, m_{NaCl} and m_{CO2}. The quality is interpolated for the segments between the different degassing pressures. Then, it involves step 1 to 5 (see blocks in Figure 3.8) in order to recalculate the segment properties. Consequently, it affects also the pressure and enthalpy in these certain segments. Therefore, this interpolation scheme is iterated while $abs\left(P^{old}(i-2) - P^{new}(i-2)\right) > 0.01$, which means the pressure loss is not yet continuous over the segments. The third condition states that iteration is performed until the maximum nr. of iterations is reached, which is a user-defined parameter. If one of the conditions is not satisfied, calculation continues with segment $(i+1)$. Within the alternative loop the GFP model is invoked by *fCalc_geofprops3.* In contrast to *fCalc_geofprops2,* which calculates the fluid properties according to the two-phase model of the GFP Excel model from Figure C.4, *fCalc_geofprops3* calculates the temperature and effective properties with the single phase fluid properties and the quality. As in *fCalc_geofprops2,* an iterative calculation scheme is necessary to find the temperature *T,* that satisfies *P* and *h*. Figure C.5 and Figure C.6 present the liquid phase and vapor phase interface of the GFP Excel model, respectively.

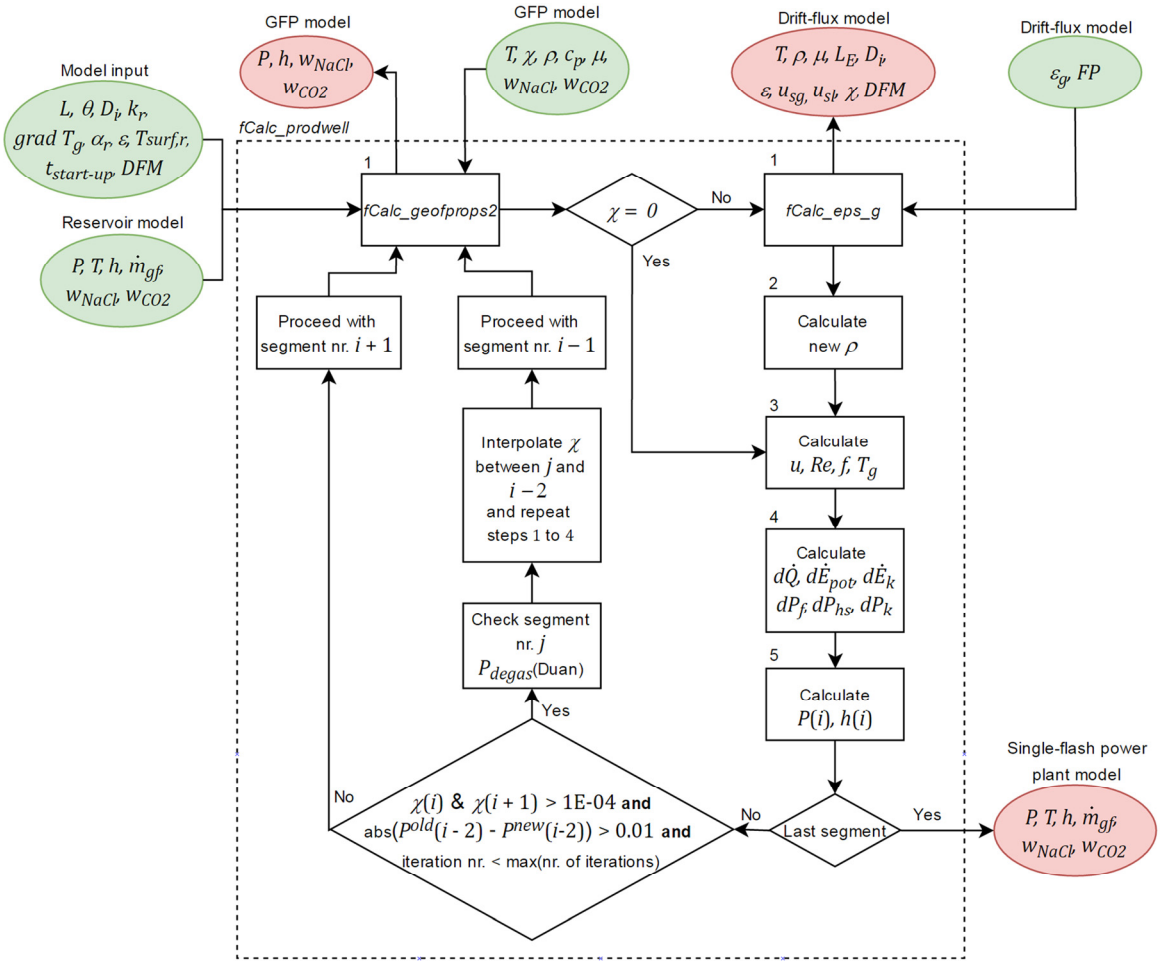


Figure 3.8: Calculation procedure production well model – self-flowing with system border (black dashed), input variables (green) and output variables (red).

3.5. Drift-Flux Model

The purpose of the drift-flux model is to calculate the void fraction in a segment of the production well. The drift-flux model can be seen as a part of the production well, therefore the system borders of the production well model apply also to the drift-flux model. However, the complexity and the importance of the drift-flux model and the production well model made it desirable to split it into two sub models. The consideration to model two-phase flow by the drift-flux model has been discussed in Section 2.4.3.2.

3.5.1. Phenomena and Assumptions

A more comprehensive discussion about the phenomena and assumption can be found in Section 2.4.3.3. There is also overlap with the assumption for the production well discussed in Section 3.4.2. Briefly in summary, there are five flow patterns that describe the two-phase flow: bubble, dispersed bubble, slug, churn and annular. The flow is one-dimensional, cocurrent and vertical upwards. The flow is fully developed and turbulent.

3.5.2. Calculation Procedure

Figure 3.9 presents the calculation procedure of the drift-flux model. The model contains five different drift-flux correlations, namely Nicklin (1961), Rouhani and Axelsson (1970), Dix (1971), Toshiba (Coddington and Macian, 2002) and Hasan et al. (2010), discussed in Section 2.4.3.3. The drift-flux correlation is user-defined in the model input. The default correlation is Rouhani and Axelsson (1970).

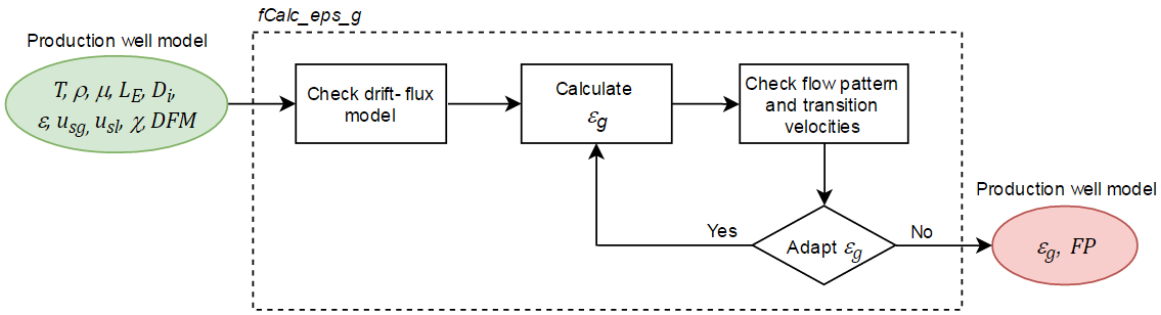


Figure 3.9: Calculation procedure drift-flux model with system border (black dashed), input variables (green) and output variables (red).

The model equations for the drift-flux correlations have been discussed as well in Section 2.4.3.3. The void fraction ε_g is calculated by eq. (2.41). The equations for the flow distribution parameter C_0 and drift-flux velocity u_{gu} can be found in Table 2.6. Except for Hasan et al. (2010), eqs. (2.42) – (2.52) describe the drift-flux correlation.

3.6. Single-Flash Power Plant Model

3.6.1. Purpose and System Border

The purpose of the single-flash power plant model is to study and simulate the energetic and environmental performance of the geothermal power plant in steady-state operation.

The system borders are the inflow from the production well and outflow to the injection well. Additionally, system borders are defined for the generator/grid, condenser pump/grid, injection pump/grid, make-up pump/grid, cooling water pump/grid, condenser/cooling tower and depending on the gas extraction system, the (SE/C)/environment or the centrifugal compressor/(grid/environment). The components can be found in Figure 2.24.

The single-flash power plant components studied in this sub model are the cyclone separator, steam turbine. generator, condenser, SE/C or centrifugal compressor, condenser pump, make-up pump, cooling water pump and injection pump.

3.6.2. Phenomena and Assumptions

All components (Section 3.6.1):

1. The geothermal fluid flowing in the power plant is in liquid phase or two-phase.
2. Pressure losses in piping between components are negligible.
3. There are no chemical reactions between the components.

Cyclone separator:

4. Flash process is isenthalpic.

Steam turbine:

5. At the inlet, gas is saturated.
6. Components in the gas can be H_2O and CO_2.
7. Gas is an ideal mixture of gases.
8. CO_2 dissolution in condensed H_2O is neglected (see Section 2.4.4.3)

Condenser:

9. Component is perfectly insulated and adiabatic.

10. Pressure loss is neglected.
11. CO_2 dissolution in condensed H_2O is neglected (see Section 2.4.4.3)

Gas extraction system:

12. Gas extraction system is SE/C or centrifugal compressor.
13. CO_2 is vented to the environment.
14. SE/C has two stages.

Pumps:

15. There is only liquid flow.
16. Injection pump is isothermal.
17. Liquid is incompressible.

3.6.3. Calculation Procedure

Figure 3.10 presents the calculation procedure of the single-flash power plant model. The state numbers correspond to Figure 2.24. A comprehensive description of this calculation procedure has been discussed in Section D.1.

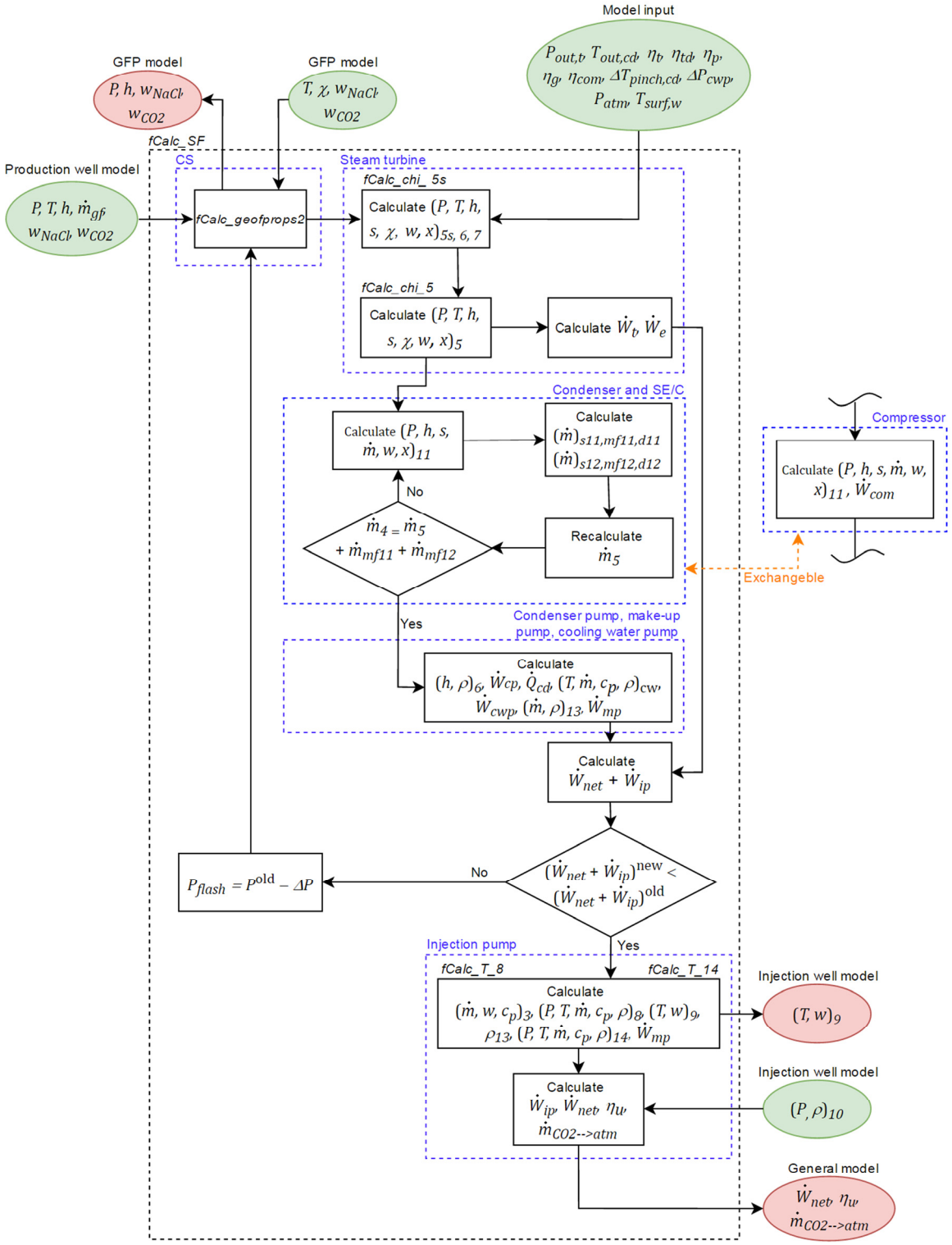


Figure 3.10: Calculation procedure single-flash power plant model with system border (black dashed), component borders (blue dashed), input variables (green) and output variables (red).

3.7. Injection Well Model

The injection well model can be invoked by two functions: *fCalc_injwell* and *fCalc_injwell_BC*.

3.7.1. Purpose and System Border

The purpose of the injection well model is to study and simulate the hydraulic and thermal behavior of the geothermal fluid within the injection well in steady-state conditions. Additionally, it connects the single-flash power plant model and the binary cycle power plant model with the reservoir model and closes the geothermal fluid circuit.

The injection well model shows much resemblance with the production well model – self-flowing discussed in Section 3.4. The system borders are at the top, the inflow from the power plant model, and at the bottom, the outflow to the reservoir model. Additionally, the GFP MATLAB model/production well model interface is a system border. The well/rock interface is not defined as a system border as has been explained in Section 3.4.1.

3.7.2. Phenomena and Assumptions

The phenomena and assumptions 3, 4, 6, 7, 8, 9, 10, 11, 12, 13 and 14 outlined in Section 3.4.2 apply to the injection well model as well. Assumptions 1, 2 and 5 are modified to:

1. Geothermal fluid is always in liquid phase.
2. The geothermal fluid always flows in one direction, which is downward flow.
3. All fluid properties are being modeled homogeneously. This means uniform properties on a cross section.
4. Kinetic pressure losses are neglected.

3.7.3. Calculation Procedure

3.7.3.1. Numerical Model

The numerical model is based on Figure 3.7, only the direction of flow is downwards. The numerical method is equivalent to the method discussed in Section 3.4.3.1 used for the production well model – self-flowing.

3.7.3.2. Model Equations

The conservation laws discussed in Section 2.4.2, are applied to the segments of the injection well. Only-liquid flow is assumed in the injection well. Therefore, the mass, momentum and energy balance of eqs. (3.1), (3.2) and (3.4) are applied to the flow, respectively.

The constitutive equations necessary to solve the conservation equations are based on the effective properties derived from the single phase fluid properties and the quality. The friction factor f, the Reynolds number Re and the velocity u for liquid-only flow are calculated according to eqs. (2.24), (2.25) and (2.26), respectively. T_g depends on geothermal gradient, which is a user-defined variable in the model input. The heat flow rate $\dot{Q}$ for every single segment is calculated by eq. (3.6) derived from eq. (2.18).

3.7.3.3. Calculation Procedure

Figure 3.11 shows the calculation procedure of the injection well model. Starting input variables (left) are imported from the output variables of the reservoir model at the injection well side. The user-defined variables are imported from the model input and the geothermal fluid properties at the top of the production well are imported from the power plant model. Then segment geothermal fluid properties are calculated from bottom to top of the injection well with *fCalc_geofprops2,* which is a function that exports output and imports input (top left) from the GFP MATLAB model described in Section 3.2. Subsequently, the heat flow rate $\dot{Q}$, potential energy loss $\dot{E}_{pot}$, frictional pressure loss dP_f and hydrostatic pressure loss dP_{hs} are

computed. Kinetic energy losses and kinetic pressure losses can be neglected, because the flow is assumed incompressible. Finally, the conservation equations are solved to obtain pressure P and enthalpy h at the outflow of the respective segment and the inflow of the next segment. These simulations are looped until the last segment and the output variables (right) are known, which are the input variables for the single-flash power plant model.

Figure 3.11 shows an alternative loop after the calculation of the last segment. While $abs(T_{wh} - T_9) > 1$ °C, the injection well is recalculated from bottom to top. Where T_{wh} is the fluid temperature of the last segment and T_9 is the fluid temperature before the injection pump, which is equal to T_{10} assuming an isothermal pump. The $T_{res,in}^{new}$ is then calculated by eq. (3.7).

$$T_{res,in}^{new} = T_{res,in}^{old} - (T_{wh} - T_9) \tag{3.7}$$

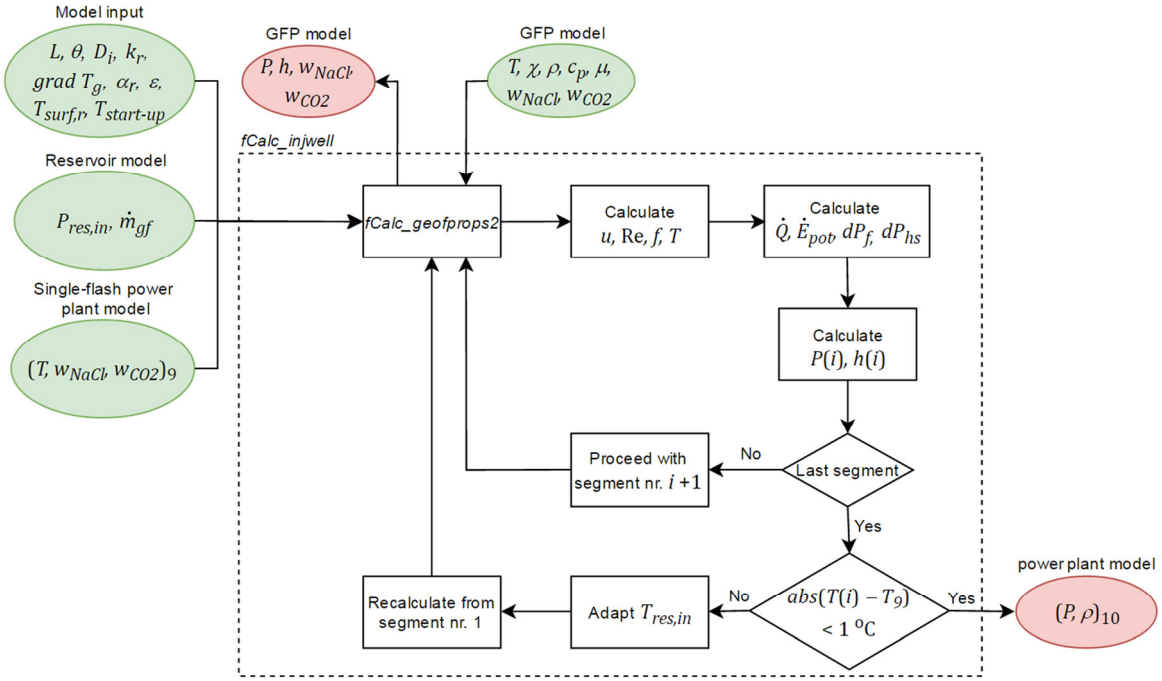


Figure 3.11: Calculation procedure injection well model with system border (black dashed), input variables (green) and output variables (red).

3.8. Production Well Model – Gas Lift

3.8.1. Purpose and System Border

The purpose of the production well – gas lift model is to study and simulate the hydraulic and thermal behavior of the geothermal fluid within a well with a gas lift system in steady-state conditions. Figure 3.12 presents a schematic of the production well with a gas lift system. It is a one-dimensional numerical model, where segments are distributed along the length of the well. The gas is compressed at the earth's surface and it is injected at a segment boundary.

The system borders are at the bottom the inflow from the reservoir model and at the top the outflow to the single-flash power plant model. Additionally, the GFP model/production well model – gas lift interface and drift-flux model/production well model – gas lift interfaces are system borders. As one would perhaps expect, the well/rock interface is not defined as a system border. The heat flow rate $\dot{Q}$ is described by an analytical equation given in eq. (2.18). The geothermal rock surrounding the production well is part of the production well and discretized as such. The rock formation is radially assumed to be infinite.

The assumptions and the calculation procedure related to this model are discussed in Sections 3.8.2 and 3.8.3, respectively.

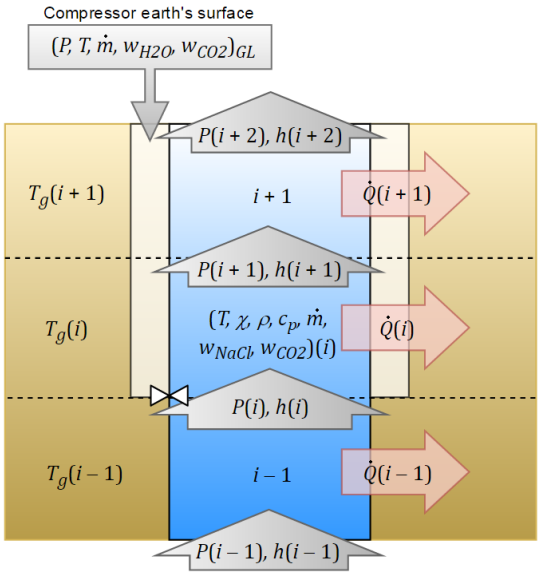


Figure 3.12: Schematic of the production well model – gas lift. Wellbore and rock formation is divided into multiple segments $(i-1, i, i+1)$. The compressed gas is injected at the segment boundary. Compressed gas flows through the annulus duct to the gas lift valve, where it is injected into the production string.

3.8.2. Phenomena and Assumptions

The phenomena and assumptions that have been applied to the production well model – self-flowing in Section 3.4.2, also apply to the production well – gas lift model. Only assumption 9 has been adapted for the part of the production well surrounded by the gas lift duct.

9. Heat flows only radially. It is governed by conduction for the part of the production well surrounded by the geothermal rock. It is governed by convection and conduction for the part of the production well surrounded by the gas lift duct.

Additional assumptions:

1. The gas is injected at a segment boundary.
2. The pressure of the gas is equal to the pressure of the geothermal fluid at the corresponding segment boundary.
3. The gas lift valve does not have a physical length. Additionally, the diameter of the production well is not affected by the gas lift system.
4. The gas flowing downwards in the annulus is fully developed and turbulent.
5. The gas is pure CO_2.
6. The hydraulic diameter of the gas lift duct is 0.05 m. This means that $r_{ao} - r_{Wo}$ = 0.025 m (see Figure 3.13).

3.8.3. Calculation Procedure

3.8.3.1. Numerical Model

The description of the numerical model for the production well model – self-flowing applies also to the production well model – gas lift. Therefore, it is referred to Section 3.4.3.1.

There is one exception for the segment containing the gas lift valve. In Section 3.4.3.1, it has been mentioned that segment properties are constant and equivalent to the inflow conditions $(P, h, \dot{m}\ and\ w)$ of the respective segment (i) and the outflow conditions of the previous segment $(i-1)$. The segment

containing the gas lift valve has different inflow conditions compared to the outflow conditions of segment $(i-1)$. This is further explained in Sections 3.8.3.2 and 3.8.3.3.

3.8.3.2. Model Equations

The production well is basically divided in two parts. One part below gas injection and one part above gas injection. The conservation laws, discussed in Section 3.4.3.2, are also applied to those particular segments of the production well model – gas lift, before the segment with gas injection. This means that eqs. (3.1) – (3.6) apply to those segments. Figure 3.13 shows the schematic of the part of the production well model – gas lift containing the gas lift valve. This figure refers to eqs. (3.8) – (3.13).

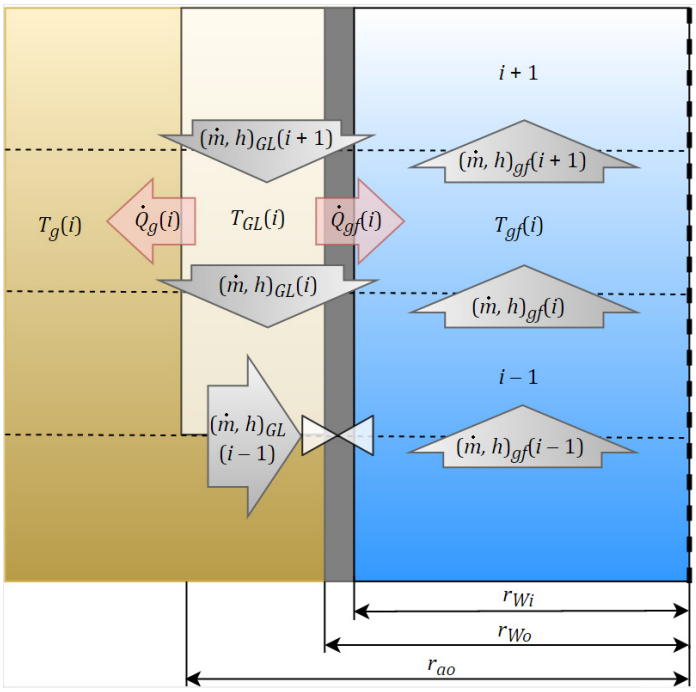


Figure 3.13: Schematic of the part of the production well – gas lift model containing the gas lift valve (not to scale). This figure shows the mass- and heat flow rates for the part of the production well surrounded by the gas lift (GL) annulus. The segment containing the gas lift valve is $i-1$. The segments above the gas lift valve segment start from i.

Mass balance:

The mass balance for the segment $(i-1)$ containing the gas lift valve is given by eq. (3.8). The mass balance for all segments above the gas lift segment is given by eq. (3.9).

$$\dot{m}_{gf}(i) = \dot{m}_{gf}(i-1) + \dot{m}_{GL} \tag{3.8}$$

$$\dot{m}_{gf}(i+1) = \dot{m}_{gf}(i) \tag{3.9}$$

Momentum balance:

The momentum balance given by eq. (3.3) for two-phase flow also applies to the segment with gas injection. According to assumption 2 in Section 3.8.2, the pressure level is not affected by gas injection at the height of the gas lift valve.

Energy balance:

The energy balance for the segment $(i-1)$ of the production string containing the gas lift valve is given by eq. (3.10). The energy balance for all segments above the gas lift segment is given by eq. (3.11).

$$h_{gf}(i) = \left[\frac{(\dot{m}h)_{gf} + (\dot{m}h)_{GL}}{\dot{m}_{gf} + \dot{m}_{GL}}\right](i-1) + \left[\frac{\dot{Q}_{gf}}{\dot{m}_{gf} + \dot{m}_{GL}}\right](i-1) - [gL\cos\theta](i-1) \tag{3.10}$$

$$h_{gf}(i+1) = h_{gf}(i) + \left[\frac{\dot{Q}_{gf}}{\dot{m}_{gf} + \dot{m}_{GL}}\right](i) - [gL\cos\theta](i) \tag{3.11}$$

The heat flow rate $\dot{Q}_{gf}$ in eqs. (3.10) and (3.11) is now given by eq. (3.12), where the variables are a function of segment number logically. The overall heat transfer coefficient is then given by eq. (A.23). The calculation procedure of the overall heat transfer coefficient is given in Section A.3 by solving eqs. (A.24)-(A.41).

$$\dot{Q}_{gf} = UA\left(T_{GL} - T_{gf}\right) \tag{3.12}$$

The energy balance of the segments in the gas lift duct is equal for all the duct segments. It is given by eq. (3.13). The heat flow rate to the surrounding rock is given by eq. (3.6), which was the analytical solution for the heat flow rate from a geothermal well to the surrounding rock proposed by Garcia-Gutierrez et al. (2002).

$$h_{GL}(i) = h_{GL}(i+1) + \left[\frac{\dot{Q}_{gf} + \dot{Q}_{g}}{\dot{m}_{GL}}\right](i) - [gL\cos\theta](i) \tag{3.13}$$

3.8.3.3. Calculation Procedure

Figure 3.14 shows the calculation procedure for the production well model – gas lift. The starting input variables (left) are obtained from the output variables of the reservoir model at the production well side and the user-defined variables from the model input. Until the segment of the gas lift valve is reached the calculation procedure is exactly identical to the procedure of the production well model. From the point of gas injection, two-phase properties and alternative heat transfer calculations are applied, discussed in Section 3.8.3.2.

The discrepancy between the degassing pressures of the GFP Excel Model and Duan and Sun (2003), which has been discussed in Section 3.2 does not apply in this case. It is assumed that gas injection takes place before flashing occurs in the self-flowing production well. The comparison is performed between identical production wells, so in the production well with gas lift there may be assumed that flashing has not occurred at that point.

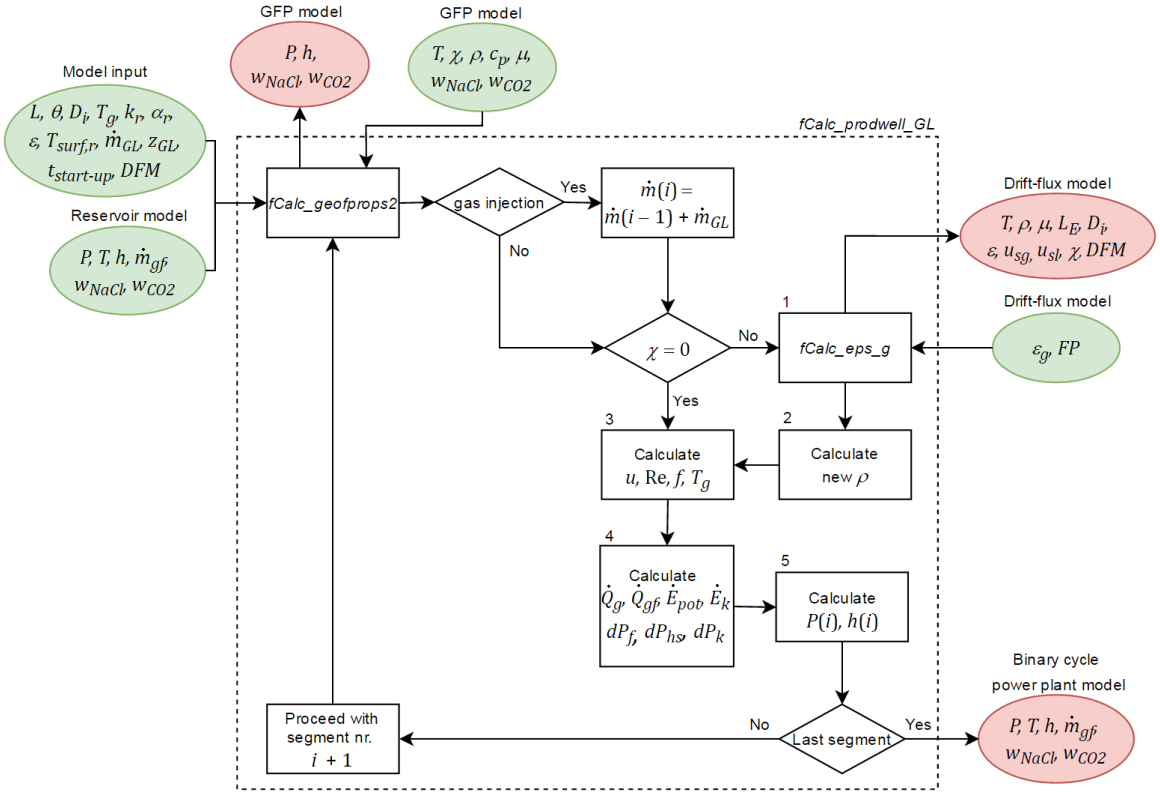


Figure 3.14: Calculation procedure production well – gas lift model with system border (black dashed), input variables (green) and output variables (red).

3.9. Binary Cycle Power Plant Model

3.9.1. Purpose and System Border

The purpose of the binary cycle power plant model is to study and simulate the energetic and environmental performance of the geothermal power plant in steady-state operation.

The system borders are the inflow from the production well and outflow to the injection well. Additionally, system borders are defined for the gas lift compressor/grid, generator/grid, condenser pump/grid, cooling water pump/grid, make-up pump/grid, injection pump/grid and the condenser/environment. The components can be found in Figure 2.23.

The binary cycle power plant components studied in this sub model are the compressor, evaporator, preheater, gas turbine, condenser, condenser pump, make-up pump, cooling water pump and injection pump.

3.9.2. Phenomena and Assumptions

All components (Section 3.9.1):

1. The geothermal fluid flowing in the power plant is in liquid phase.
2. Pressure losses in piping between components are negligible.
3. There are no chemical reactions between the components.
4. The working fluid incorporated is i-pentane (isopentane)
5. Working fluid pressure is subcritical.

Compressor:

6. Compressed gas is pure CO_2.
7. Two conditions are assumed: compression from atmospheric conditions or compression from wellhead conditions.

Preheater:

8. Working fluid is preheated until saturated liquid.
9. Pinch point is 5 K.
10. Pressure drop is neglected.
11. Only liquid part at the wellhead is sent to the preheater/evaporator.

Evaporator:

12. Working fluid is evaporated isothermally and isobarically.
13. Working fluid can be superheated.
14. Pinch point is 5 K.
15. Pressure drop is neglected.

Gas turbine:

16. At the inlet, gas is saturated or superheated.
17. At the outlet, working fluid is in gas phase.

Condenser:

18. Component is perfectly insulated and adiabatic.
19. Pressure drop is neglected.
20. At the outlet, working fluid is saturated liquid.

Pumps:

21. There is only liquid flow.
22. Injection pump is isothermal.
23. Liquid is incompressible.

3.9.3. Calculation Procedure

Figure 3.15 presents the calculation procedure of the binary cycle power plant model. The state numbers correspond to Figure 2.23. Data tables with properties of working fluids isopentane have been incorporated in the model.

Condenser pump:

The calculation starts with the condenser pump (CP) with input from the top of the production well model – gas lift. It uses the outlet pressure of the turbine and the critical pressure of the working fluid as an initial calculation. The required power is calculated according to eq. (2.67).

Preheater/evaporator:

An initial estimate of the working fluid mass flow rate is performed by solving the energy balance given in eq. (2.69). The properties at state C are user-defined, this is the minimum allowed injection temperature. The pinch point between B and 4 is 5 K. Now that P and T are known the enthalpies are calculated in *fCalc_geofprops4*. Subsequently, the heat flow rates are calculated. Then an iterative procedure follows. If the calculated h_1 is lower than the saturated vapor enthalpy, $T_{wf,ev} = T_5$ is decreased. If T_1 is within T_{pinch} of T_A, $T_1 = T_A - T_{pinch}$. Then the working fluid mass flow rate must be adjusted. The energy balance is solved over the entire preheater/evaporator.

Turbine:

The calculation of the turbine is rather straightforward, see eq. (2.65). This whole procedure is iterated until the maximum $\dot{W}_{net} + \dot{W}_{ip}$ is obtained.

Cooling water pump and make-up pump:

The required power of the pumps is calculated by eq. (2.60).

Injection pump:

The calculation of $\dot{W}_{ip}$ and subsequently $\dot{W}_{net}$happens after the injection well is computed.

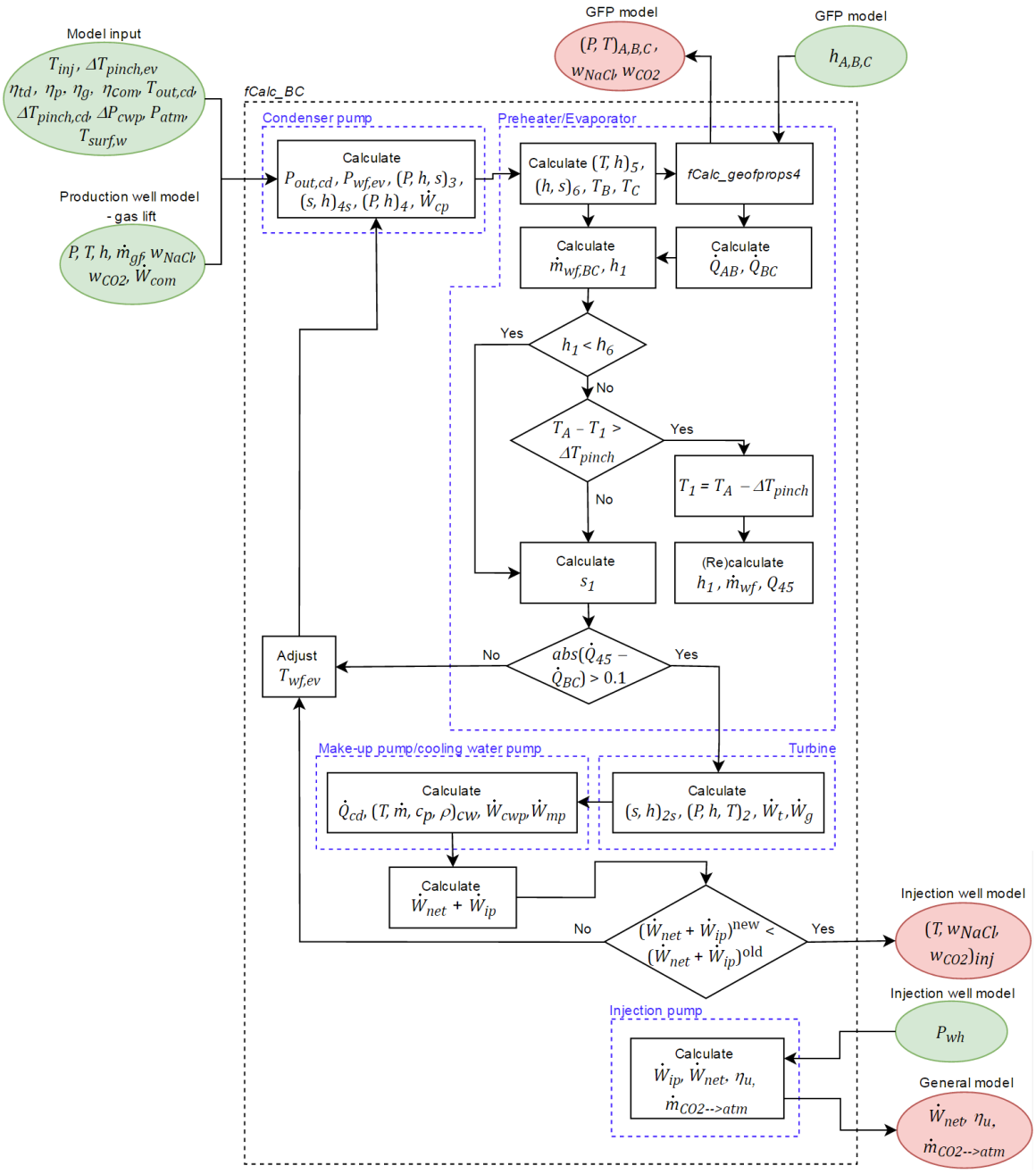


Figure 3.15: Calculation procedure binary cycle power plant model with system border (black dashed), input variables (green) and output variables (red).

4

Model Validation & Sensitivity Analysis

The present chapter presents the validation of the mathematical model described in Chapter 3. Firstly in Section 4.1, the production well model – self-flowing has been hydraulically and thermally validated with field data from literature. Section 4.2 discusses the validation of the GFP MATLAB model by comparison of the vapor quality of the geothermal fluid as a function of temperature, pressure and composition. Then Section 4.3 treats the validation of the drift-flux model. Section 4.4 presents the qualitative validation of the production well with gas lift. Section 4.5 discusses the validation of the single-flash power plant and the binary cycle power plant model.

Besides the model validation, the sensitivity of certain input parameters and assumptions are discussed. The level of detail of the mathematical model described in Chapter 3 is relatively high, especially of the two production well models and the geothermal power plant models. Insight in how certain parameters affect the simulation contributes to further model development. Also, the knowledge obtained from this sensitivity analysis is used to determine the input parameters to examine in Chapter 5. Section 4.6 discusses the sensitivity analysis of the production well model – self-flowing. In Section 4.7, the power plant model is treated.

4.1. Self-Flowing Production Well Model Validation

The self-flowing production well model has been validated hydraulically and thermally with field data from literature. In Section 4.1.1, field data of various geothermal wells are discussed. This field data was used as input for the present model to calculate pressure and temperature profiles. Section 4.1.2 covers the analysis of the results of the simulation. Finally, in Section 4.1.3 conclusions are drawn related to the validation of the production well model.

4.1.1. Field Data and Model Input Parameters

Ambastha and Gudmundsson (1986a) and Ambastha and Gudmundsson (1986b) compared their model calculations for two-phase flow in geothermal wells with pressure profiles from ten different geothermal wells around the world. They assumed pure water in their comparison and model calculations as geothermal fluid. Two wells have been found suitable for validation of the present model: East-Mesa 6-1 (Imperial Valley, CA) and Ngawha 11 (New Zealand). These wells are the only ones out of these ten wells having single-phase fluid (liquid) at the bottom hole of the well with temperatures up to 250 ˚C. Field data of East-Mesa 6-1 was also used by Chadha et al. (1993) to validate their homogeneous two-phase model for the prediction of two-phase flow in vertical wells. Additionally, the field data from the NWS-1 Sabalan well (Iran) from Akbar et al. (2016), who proposed a finite element model for two-phase in wellbores, has been used for validating the present model. They assumed pure water as well. Table 4.1 presents the field data of the East-Mesa 6-1, Ngawha 11 and NWS-1 Sabalan geothermal wells. This is also the input for the present model to calculate the pressure and temperature profiles.

Besides the pure water models, geothermal wells in which salts and NCG played a significant role were evaluated. Barelli et al. (1982) compared four geothermal wells with their two-phase flow model for geothermal wells in the presence of salts in the range of 1 – 10 wt% and NCG in the range of 0 – 15 wt%. These salts and NCG were assumed consisting entirely of NaCl and CO_2, respectively, just as in the present model. Field data of three of these four wells have been found useful for validation of the present model. In

the present work, in accordance with Barelli et al. (1982), these wells have been referred to as W2, W3 and W4. The field data of these wells are presented in Table 4.2.

In Table 4.1 and Table 4.2, it can be seen that thermal conductivity and thermal diffusivity do not apply or a value is assumed in the present study. The properties of the geothermal rock, including geothermal temperature gradient, are difficult to establish. Ambastha and Gudmundsson (1986a) assumed no heat transfer between fluid and rock. In this case, the properties do not apply to the simulation. Akbar et al. (2016) and Barelli et al. (1982) did not report or they assumed average values for these properties. In the present work, average values were obtained from Eppelbaum et al. (2014). Average thermal conductivity of common rocks reported in literature are approximately in the range of 1.5 – 3 W m^{-1} K^{-1} at atmospheric temperature and pressure. The thermal conductivity is also a function among others of temperature and porosity, where thermal conductivity decreases with increasing temperature and porosity. The geothermal gradient in Table 4.2 is a linear relation between bottom hole- and surface temperature. Segment lengths were in the range of 20 – 25 m in all simulations.

Table 4.1: Field data of geothermal wells: East-Mesa 6-1, Ngawha 11 and NWS-1 Sabalan (Ambastha and Gudmundsson, 1986a; Akbar et al., 2016). This field data was used as model input in the present model. For two wells, extra simulations were performed (Ngawha 11-2 and NWS-1 Sabalan-2) with additional model input parameters, which have not been considered by Ambastha and Gudmundsson (1986a) and Akbar et al. (2016). These additional model input parameters are highlighted in blue between brackets.

Well	East-Mesa 6-1	Ngawha 11-1 (Ngawha 11-2)	NWS-1 Sabalan-1 (NWS-1 Sabalan-2)
Mass flow rate, kg s^{-1}	12.9	6.6	30
Bottom hole pressure, bar	93	86.3	60.09
Bottom hole temperature, ˚C	198.5	222.5	225
Production time, h	N/A	N/A (1 year)	100
CO_2 concentration, wt%	0	0 (1.23) [a]	0 (0.4) [c]
NaCl concentration, wt%	0	0 (0.4) [a]	0 (0.5) [c]
Bottom hole depth, m	2134	1002 (902) [a]	1570
Pipe diameter, m	0.2215	0.1987 from 0 – 673.5 m, 0.1504 from 673.5 – 1002 m	0.2444
Inclination angle, deg	0	0	0
Absolute roughness pipe, m	0.00018	0.00018	0.0000015
Surface temperature rock, ˚C	N/A	N/A (20) [b]	11
Geothermal gradient rock, K m^{-1}	N/A	N/A (0.41 from 0 – 500 m, 0 from 500 – 902m) [b]	0.2863 from 0 – 800 m, -0.0195 from 800 – 1570 m
Thermal conductivity rock, W m^{-1} K^{-1}	N/A	N/A (1.5) [d]	N/A (1.5 [d])
Thermal diffusivity rock, m^2 s^{-1}	N/A	N/A (1.2×10^{-6}) [d]	N/A (1.2×10^{-6} [d])
Reference	Ambastha and Gudmundsson (1986a)	Ambastha and Gudmundsson (1986a)	Akbar et al. (2016)

[a] Sheppard (1987), [b] Bromley and Bignall (2016), [c] Moghaddam (2006)

[d] This value was assumed in the present work from values in Eppelbaum et al. (2014), because it has not been given in the field data of the corresponding geothermal well.

4.1.2. Results of Simulations

The pressure and temperature profiles of the geothermal wells given in Table 4.1 and Table 4.2 are presented in Figure 4.1 and Figure 4.2, respectively. The continuous and dashed lines represent the calculated values by the present model and the markers represent the measured values obtained from literature. The quality of the matches of the measured and calculated, pressure and temperature profiles were calculated according to the method proposed by Ambastha and Gudmundsson (1986a). They

estimated the mean error and standard deviation of the mean error, according to eqs. (E.1) – (E.4) described in Section E.1. The results of the comparison of measured and calculated values are presented in Table 4.3 and Table 4.4.

Table 4.2: Field data of various geothermal wells. This field data was used as model input in the present model (Barelli et al. 1982).

Well	W2	W3	W4
Mass flow rate, kg s^{-1}	32.78	50	18.05
Bottom hole pressure, bar	98	106	64.3
Bottom hole temperature, ˚C	225	223	285.5
Production time, h	11	10	20
CO_2 concentration, wt%	3	12	0
NaCl concentration, wt%	1	6.3	10
Bottom hole depth, m	1355	2010	800
Pipe diameter, m	0.3397	0.2445	0.2445
Inclination angle, deg	0	0	0
Absolute roughness pipe, m	0.00018 [a]	0.00018 [a]	0.00018 [a]
Surface temperature rock, ˚C	18 [b]	18 [b]	18 [b]
Geothermal gradient rock, K m^{-1}	0.1528	0.1020	0.2686
Thermal conductivity rock, W m^{-1} K^{-1}	1.5 [c]	1.5 [c]	1.5 [c]
Thermal diffusivity rock, m^2 s^{-1}	1.2 x 10^{-6} [d]	1.2 x 10^{-6} [d]	1.2 x 10^{-6} [d]
Reference	Barelli et al. (1982)	Barelli et al. (1982)	Barelli et al. (1982)

[a] The absolute pipe roughness was not reported. Instead the absolute roughess of Ambastha and Gudmundsson (1986a) was used.
[b] The rock surface temperature was not reported explicitly. Barelli et al. (1982) used atmospheric temperature in their model.
[c,d] Thermal conductivity and thermal diffusivity were assumed from values in Eppelbaum et al. (2014).

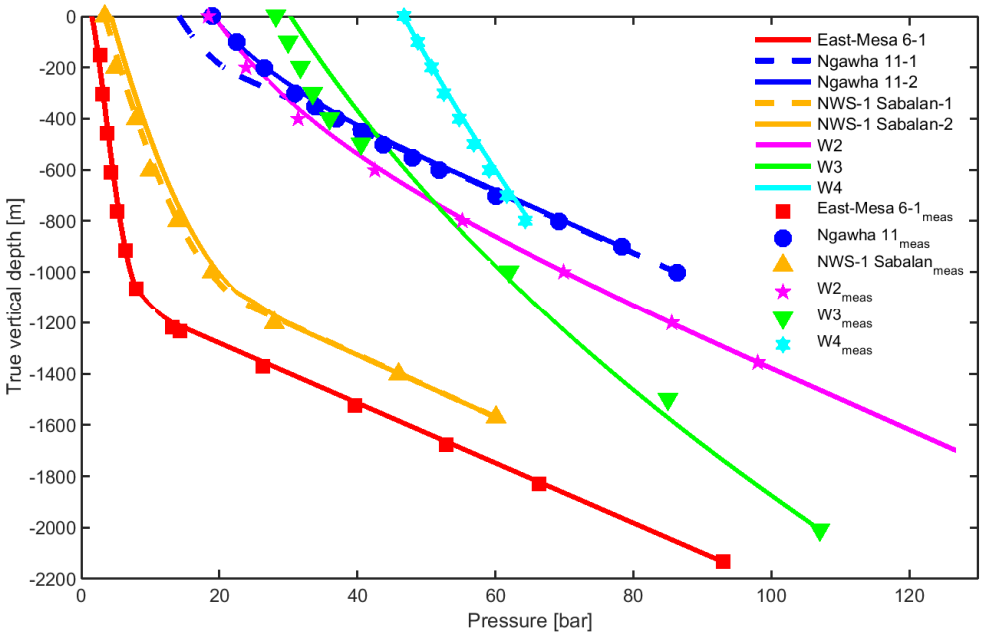


Figure 4.1: Pressure profiles of geothermal wells given in Table 4.1 and Table 4.2. The continuous and dashed lines represent the model calculations of the present model. The markers represent the measured values, which were obtained from literature.

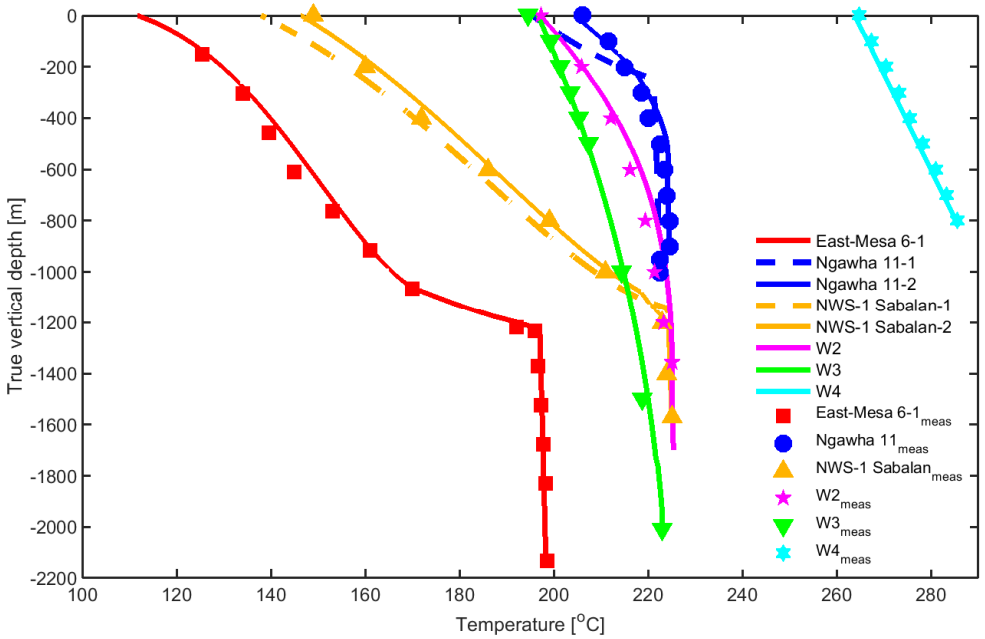


Figure 4.2: Temperature profiles of geothermal wells given in Table 4.1 and Table 4.2. The continuous and dashed lines represent the model calculations of the present model. The markers represent the measured values, which were obtained from literature.

Table 4.3: Comparison of measured and calculated, pressure and temperature profiles of geothermal wells given in Table 4.1.

Well	East-Mesa 6-1	Ngawha 11-1 (Ngawha 11-2)	NWS-1 Sabalan-1 (NWS-1 Sabalan-2)
Mean error P, bar	0.6428	1.9132 (0.9288)	0.4084 (1.0079)
Standard deviation mean error P, bar	0.6602	2.0398 (0.7022)	0.5647 (0.6704)
Mean error T, ˚C	1.2783	2.7212 (0.9802)	3.0686 (1.3112)
Standard deviation of mean error T, ˚C	1.2627	3.2104 (0.8177)	3.2678 (1.1283)
Best fitting drift-flux model	Hasan et al. (2010)	Rouhani and Axelsson (1970)	Rouhani and Axelsson (1970)

Table 4.4: Comparison of measured and calculated, pressure and temperature profiles of geothermal wells given in Table 4.2.

Well	W2	W3	W4
Mean error P, bar	0.7098	2.8677	0.4928
Standard deviation mean error P, bar	0.6268	1.5818	0.3276
Mean error T, ˚C	1.2701	0.2698	0.8828
Standard deviation of mean error T, ˚C	1.0758	0.4267	0.3626
Best fitting drift-flux model	Rouhani and Axelsson (1970)	Rouhani and Axelsson (1970)	Rouhani and Axelsson (1970)

4.1.3. Analysis and Conclusion

The present mathematical model has been validated with measured data from literature. According to literature, it can be concluded that it is difficult to accurately model wellbore flow for geothermal applications. Ambastha and Gudmundsson (1986a) and Akbar et al. (2016) did not succeed to match their models to the available measured data. One important cause is the consideration of salts and NCG. Another

cause can be the neglect of heat loss to or heat gain from the formation. Additionally, uncertainties in certain parameters can give rise to deviations. Exact information about geothermal rock thermal properties are often lacking, while it becomes increasingly important with a relatively short production time as in well W2, W3 and W4. Wellbore roughness is another difficult to measure parameter, but it can affect wellbore diameter or frictional pressure losses significantly. Finally, the accuracy of the measured values must be taken into account. In Barelli et al. (1982), errors of the order of 0.5% of the maximum pressure and 2 ˚C for temperature were given. Table 4.3 and Table 4.4 show that simulated pressure and temperature profiles by the present model were valid and reliable. Five of six simulations were performed using the drift-flux model of Rouhani and Axelsson (1970). The drift-flux model of Hasan et al. (2010) was applied to East-Mesa 6-1. A possible explanation can be the fact that the model of Hasan et al. (2010) predicts two-phase flow of pure water more accurate, while Rouhani and Axelsson (1970) is more accurate for geothermal fluids containing NCG and salts. The accuracy of the Rouhani and Axelsson (1970) drift-flux model in comparison with the other drift-flux models confirmed the conclusion of literature survey in Section 2.4.3.3. Each geothermal well from Section 4.1.1 is briefly discussed in response to the results presented in Section 4.1.2.

East-Mesa 6-1:

Ambastha and Gudmundsson (1986a) reported a mean error and mean percent error with respect to pressure of 11 bar and 59.5 %. This means that their model calculated significant different values than the measured data. They do not give a real legitimate cause for this deviation, except that the sensitivity of the enthalpy used in their model is significant and therefore the depth of flashing is underestimated. From the knowledge obtained in the present study, analysis of their pressure profile indicates that pressure drop in the two-phase flow region is overestimated. A cause could be a wrong drift-flux model, that would mean an underestimation of the void fraction. Then a larger density was calculated, which results in a larger pressure drop. It can be seen from the sharp bend in the temperature and pressure profile that the geothermal fluid did not contain NCG, where flashing happens in an instant. In the present study, all drift-flux models gave rise to an overestimation of pressure loss for the East-Mesa 6-1 well, except the drift-flux model of Hasan et al. (2010).

Ngawha 11:

Ambastha and Gudmundsson (1986a) reported a mean error and standard deviation of the mean error with respect to pressure of 10.8 bar and 5.1 bar, respectively They have mentioned that Ngawha 11 had 1.4 % of NCG in the total flow, which was not considered in their model. Also, heat loss from wellbore to geothermal rock has not been considered. Therefore, for the present study additional literature was consulted. In the second simulation NCG, salts and heat loss were taken into account as can be seen by the values between brackets in Table 4.1. Also, the two deepest data points were neglected in this simulation, because the cause of this sudden temperature rise has not been fully understood. Two possible explanations can be heat gain from the formation or influx of warmer geothermal fluid at a shallower depth ($\pm$ 900 m). Sheppard (1987) has shown that geothermal temperature was between 225 – 240 ˚C at depths from 500 – 1500 m. Nevertheless, taking composition and heat transfer into account the present model accurately predicts pressure and temperature profiles of the Ngawha 11 well. The drift-flux model of Rouhani and Axelsson (1970) have shown the most accurate match. The first simulation, where pure water was assumed, already gave a more accurate fit of the pressure profile than the model of Ambastha and Gudmundsson (1986a). The second simulation has shown an even better fit for the pressure profile. However, the most striking effect of taking NCG into account is the difference in temperature profile, which was omitted by Ambastha and Gudmundsson (1986a).

NWS-1 Sabalan:

In Akbar et al. (2016), the calculated and measured, pressure and temperature profiles have been graphically presented. They predicted the pressure profile quite accurately. On the other hand, the calculated wellhead temperature of 139 ˚C significantly deviated from the measured wellhead temperature of 150 ˚C. According to them, it was attributed to the changes in the surface temperature. In the present study, this has not been found evident. According to Moghaddam (2006), the geothermal field near NWS-1 contains NCG and salts. The geothermal fluid contained 0.4 wt% CO_2 and 0.5 wt% NaCl. This is a more logical explanation of the deviation from the measured values. The pressure profile in the second simulation (values between brackets) did not give a better fit than in the first simulation. The temperature profile however improved quite drastically. From the Ngawha 11 well, the importance of NCG and salts has been shown already. It seems that the NWS-1 Sabalan well confirms this observation. It must be said that the

temperature profile in the first simulation approached the calculated values of Akbar et al. (2016) quite accurately, in which both models assumed pure water as geothermal fluid.

W2, W3 and W4:

These wells were all described in Barelli et al. (1982), where besides the different characteristics given in Table 4.2, equivalent assumptions have been applied to all other relevant parameters. Barelli et al. (1982) used a different approach then the method in the present study. They varied the CO_2 concentration until their calculated pressure and temperature profiles matched the measured values. In the case of well W4, where no CO_2 was present, they varied enthalpy instead to find a match. In all three cases, two-phase flow was already present at the bottom hole of the well. For wells W2 and W3, the quality was obtained from the GFP MATLAB model described in Section 3.2. This value has not been explicitly given by Barelli et al. (1982). However, the effect of quality and void fraction, described by the GFP model and the drift-flux model respectively, has been understood to be significant on pressure loss in the wellbore. Therefore, it can be concluded that the present model accurately predicted the pressure and temperature profiles of W2 and W3. Well W4 had a quality of 17 % at the bottom hole according to Barelli et al. (1982). This value was applied as an additional boundary condition in the GFP MATLAB model in order to calculate the enthalpy at the bottom hole. Again, an excellent fit was obtained. All three wells were simulated with the drift-flux model of Rouhani and Axelsson (1970).

4.2. Geothermal Fluid Property Model Validation

In response to the validation of the pressure and temperature profiles in Section 4.1, it can be carefully concluded that the GFP MATLAB model works properly. In addition, the GFP MATLAB model (Section 3.2) has been validated by comparing the vapor quality as a function of pressure, temperature and composition with the vapor quality according to Wagner and Pruss (2002) and Duan and Sun (2003) for pure water and ternary solution, respectively.

Figure 4.3 shows the calculated vapor quality profiles (continuous lines) of the wells East-Mesa 6-1, Ngawha 11-2, NWS-1 Sabalan-2, W2, W3 and W4, and validated vapor quality data from literature (markers) (Wagner and Pruss, 2002; Duan and Sun 2003). Table 4.5 presents the mean error and standard deviation of the mean error with respect to quality between the present model and validated data from literature. It can be seen that the present GFP MATLAB model has been found valid and reliable.

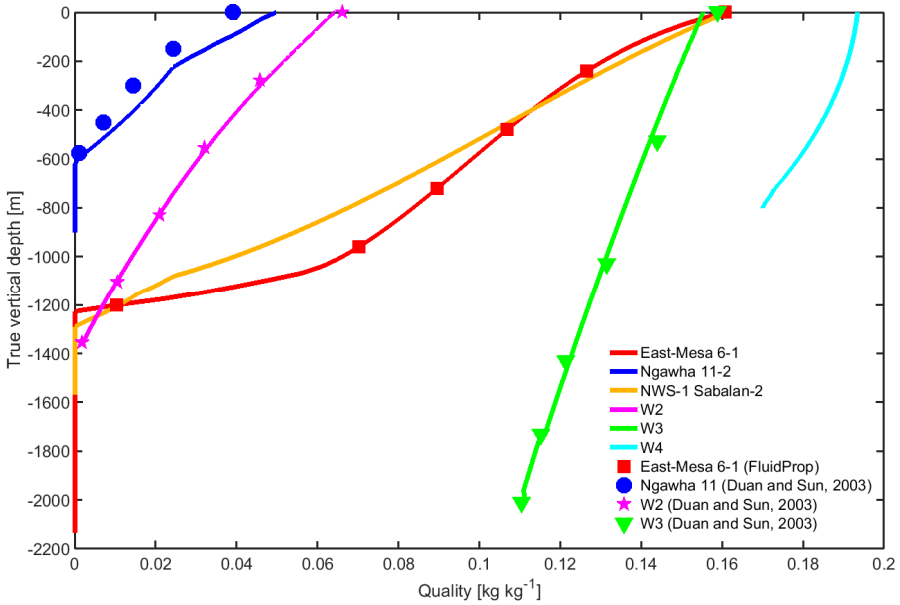


Figure 4.3: Vapor quality profiles of geothermal wells given in Table 4.1 and Table 4.2. The continuous lines represent the model calculations of the present model. The squares represent validated data from literature, which were obtained from Wagner and Pruss (2002) and Duan and Sun (2003).

The well Ngawha 11-2 deviates most from verified vapor quality data, with 0.62% and 0.76% for mean error and the standard deviation of the mean error with respect to vapor quality, respectively. This deviation

can be explained by the relatively low CO_2 concentration. The vapor quality data from Duan and Sun (2003) were obtained by entering pressure, temperature and composition. The less CO_2 a geothermal fluid contains, the more it behaves like a pure fluid. The vapor quality of pure fluids is rather a function of enthalpy accompanied by pressure or temperature than a function of pressure and temperature. The consequence of using pressure and temperature is that minor deviation in pressure and/or temperature can cause large deviations in quality close to the saturation pressure of pure water. For this exact same reason, this method failed for wells NWS-1 Sabalan-2 and W4, because the CO_2 concentration is 0.4 wt% and 0 wt%, respectively. Unfortunately, FluidProps does not have a library for salts. Therefore, Figure 4.3 and Table 4.5 do not show validated data and mean errors for vapor quality, respectively. Nevertheless, calculated vapor quality profiles of wells NWS-1 Sabalan-2 and W4 are presented to show that it was validated qualitatively by comparing the trend of these curves with the other four curves.

Table 4.5: Comparison of vapor quality profiles of various geothermal wells given in Table 4.1 and Table 4.2.

Well	East-Mesa 6-1	Ngawha 11-2	W2	W3
Mean error χ	5.70E-05	0.0062	7.71E-04	0.0014
Standard deviation of mean error χ	2.55E-05	0.0076	0.0011	0.0019

4.3. Drift-Flux Model Validation

In Section 4.1, it has been found obvious that the drift-flux model of Rouhani and Axelsson (1970) described slip between liquid and gas phase, and thereby two-phase flow, for ternary solutions most accurately. The drift-flux models, given in Table 2.6, are basically simple empirical correlations. The drift-flux model of Hasan et al. (2010) is more comprehensive and it has been found desirable to validate the flow patterns and flow pattern transitions. The East-Mesa 6-1 well was used to examine this drift-flux model, since the best fit has been found by applying Hasan et al. (2010). Figure 4.4 presents the flow pattern map according to the equations of the drift-flux model of Hasan et al. (2010). In order to validate the flow pattern map, a comparison with the validated flow pattern map of Taitel et al. (1980), given in Figure 4.5, was made. The continuous blue lines represent the boundaries between the flow patterns used in the present model. It must be explicitly said that this particular validation of flow pattern was performed with a water-air mixture at atmospheric conditions for vertical tubes of 0.05 m diameter. The continuous red line in Figure 4.4 represents the slug/annular transition according to Taitel et al. (1980). In Hasan et al. (2010), this transition is shifted leftwards, because of the additional condition that annular flow exists at void fractions above 0.7. At low liquid superficial velocities, this condition is met earlier than the minimum superficial gas velocity for transition from slug to annular flow u_{gc} given by eq. (2.51). The green line represents the transition from slug to churn flow according to Hasan et al. (2010) as well. Nevertheless, this transition and associated equations have not been adopted in the present model. Instead, the slug/churn transition from Taitel et al. (1980) has been applied, represented by the dashed blue lines. In this case, churn flow depends also on location in the wellbore, i.e. churn flow is a function of entrance length L_E and diameter D as well. Eq. (4.1) gives the minimum mixture velocity for slug/churn transition.

$$u_{mc} = \sqrt{gD}\left(\frac{L_E}{40.6D} - 0.22\right) \tag{4.1}$$

It basically means that closer to the entrance length at a certain superficial liquid and superficial gas velocity, churn flow is sooner expected. Hasan et al. (2010) assumed fully developed flow in their model, and they neglected entrance effects. In the present model, however, the gas lift valve disturbs the flow. Therefore, entrance length is calculated from the depth where gas is injected in the production well model – gas lift.

In comparison with the relatively simple drift-flux correlations from Table 2.6, the comprehensive drift-flux model of Hasan et al. (2010) applied different equations for different flow patterns. On one hand, the effect of different flow patterns on two-phase flow has been taken into account. While on the other hand, transitions between flow patterns can be sharp, although they are rarely abrupt. Therefore, Hasan et al. (2010) proposed equations, in which the distribution parameter C_0 has been calculated with an exponential-weighted average for smoothening the transitions between flow patterns. The existence of certain flow patterns in geothermal wells has not been directly validated. Chadha and Malin (1993) validated their two-phase model with the same measured data found in Ambastha and Gudmundsson (1986a). They

implemented different equations and used a different terminology for different flow patterns than Hasan et al. (2010). Nevertheless, the trend for the pressure-, density- and quality profile matched reasonably well. In their model, flow pattern transitions were even sharper than in the model of Hasan et al. (2010). They overestimated pressure loss slightly, from which it can be concluded that smoothening the flow pattern transition according to Hasan et al. (2010) paid off. Section E.2 presents the pressure-, density-, vapor quality- and void fraction profiles calculated with the present model using the drift-flux model of Hasan et al. (2010) and the pressure-, density- and vapor quality profiles of Chadha and Malin (1993).

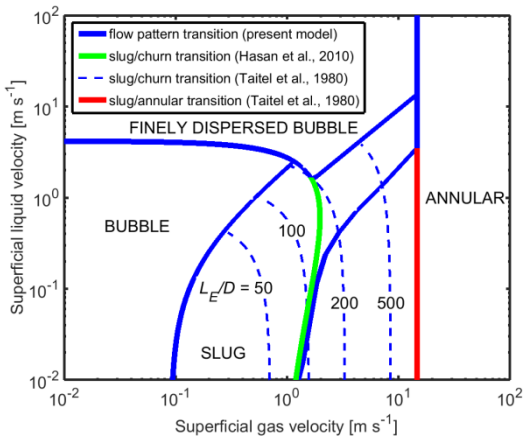


Figure 4.4: Flow pattern map according to drift-flux model of Hasan et al. (2010) for vertical tubes $D = 0.05$ m, $T = 25$ °C, $P = 1$ bar. Blue (continuous and dashed) have been implemented in the present model.

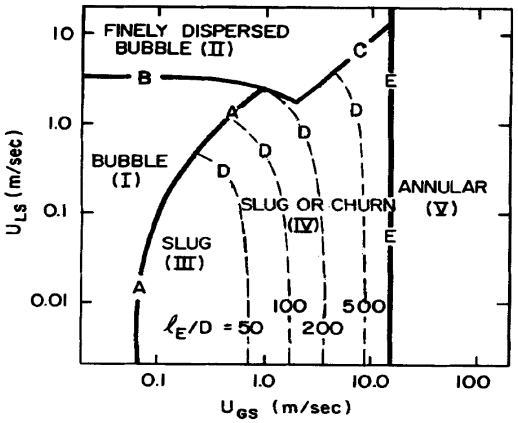


Figure 4.5: Flow pattern map from Taitel et al. (1980) for vertical tubes $D = 0.05$ m, $T = 25$ °C, $P = 1$ bar.

4.4. Production Well with Gas Lift Model Validation

This section was written in anticipation of Chapter 5, from which the model input parameters for this production well were adopted, given in Table F.1 and Table F.2. Furthermore, the mass fraction of NaCl was 0.05 kg kg^{-1} and the mass fraction of CO_2 was 0.015 kg kg^{-1}. The flashing depth and consequently depth of the gas lift valve was 1400 m. Three scenarios were compared: the self-flowing production well was compared to two gas lifted production wells with 0.5 and 1.0 kg s^{-1} of lift gas. Because of the novelty of this concept for geothermal applications with its characteristics, experimental data has not been found. Therefore, the production well with gas lift has been validated on the basis of trend lines of pressure, temperature, enthalpy, quality, void fraction and density.

Figure 4.6 shows six property profiles as a function of true vertical depth of these three production wells. It can be seen that as expected in (a), the pressure loss decreases in the production wells with gas lift. Consequently, temperature decrease in (b) is smaller, because flashing is associated to temperature decrease. The enthalpy losses in (c) are relatively similar. A small enthalpy step can be seen on the point of gas lift, because CO_2 is injected at that point, which affects the enthalpy. The quality (gas mass fraction) in (d) suddenly increases at the gas lift valve depth as a result of the injected CO_2. Logically, the quality for 1.0 kg s^{-1} injected gas mass flow rate is higher than for 0.5 kg s^{-1} injected gas. Finally, an interesting trend can be found in (e) and (f), which represent the void fraction and the density. At first, the density of the geothermal fluid decreases after injection of lift gas. The result of less hydrostatic pressure can be seen in (a). But consequently, the pressure near the top of the production well is higher for the production well with gas lift than for the self-flowing production well. At a certain moment, density in the gas lifted production wells becomes higher than of the self-flowing well. Since the void fraction is a function among others of density, the void fraction shows a similar trend.

4.5. Geothermal Power Plant Model Validation

In this section, the power output and thermal efficiencies of existing geothermal power plants have been compared to the output of the present mathematical model. In Section 4.5.1 and Section 4.5.2, the single-flash power plant and the binary cycle power plant are discussed, respectively. Geothermal power plant data

was obtained from Zarrouk and Moon (2014). They have published a worldwide review of efficiencies of geothermal power plants. Their work covers 94 geothermal power plants in total, of which 34 were single-flash power plants and 31 were binary cycle power plants. It has been shown that there is a correlation between thermal efficiency of geothermal power plants, reservoir enthalpy, mass flow rate of geothermal fluid and net power produced. They calculated thermal efficiency according to eq. (4.2).

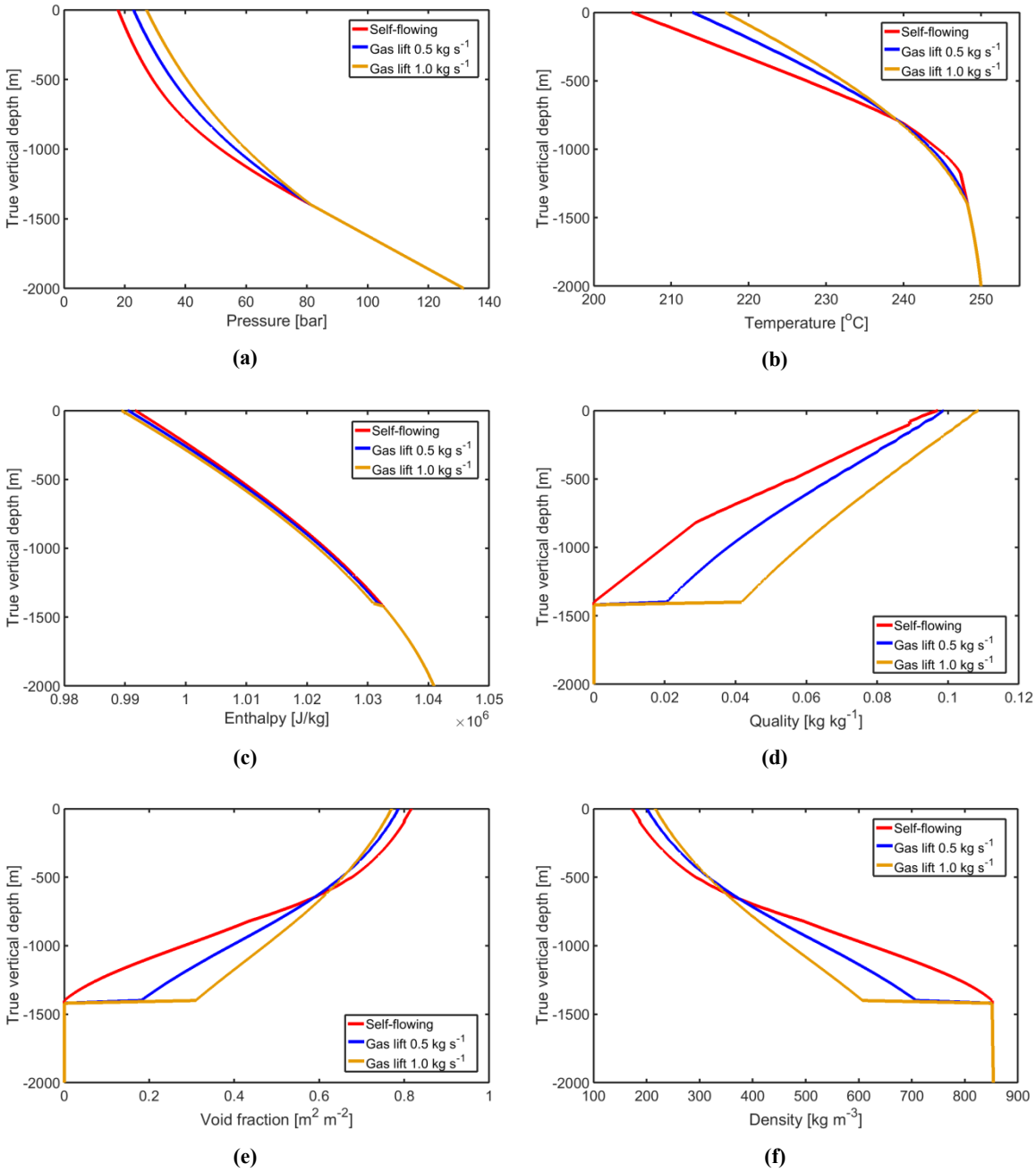


Figure 4.6: Property profiles as a function of true vertical depth of (a) pressure (b) temperature (c) enthalpy (d) quality (gas mass fraction) (e) void fraction (f) density.

$$\eta_{th} = \frac{\dot{W}_{net}}{\dot{m}_{gf} h_{res}} \tag{4.2}$$

Where h_{res} is based on pure water and $\dot{W}_{net}$is calculated taking NCG and auxiliary power supply into account. It must be noted that this equation is different then the calculation of the thermal efficiency in the present study, which is given in eq. (2.62). Also, it can be arbitrary to compare geothermal systems all over the world on the basis of a reservoir enthalpy value, since enthalpy is relative to its reference state.

Additionally, the enthalpy alone does not give sufficient information about the state of the reservoir. Therefore generally, the most appropriate method to compare geothermal power plants is according to the utilization efficiency, given in eq. (2.63). Nevertheless, the fact that the required reservoir data to calculate the utilization efficiency has not been available for all geothermal fields makes eq. (4.2) and the results from Zarrouk and Moon (2014) a valuable method to validate the geothermal power plant models.

4.5.1. Single-Flash Power Plant

Figure 4.7 shows the thermal efficiency according to eq. (4.2) as a function of reservoir enthalpy of the single-flash power plants published in Zarrouk and Moon (2014). Only reservoir enthalpies in the range of 700 – 1100 kJ kg^{-1} have been presented, because two-phase reservoirs have not been studied in the present work. The wellhead properties of five wells (East-Mesa 6-1, Ngawha-11, NWS-1 Sabalan, W2 and W3) presented in Table 4.1 and Table 4.2 were used as input for the single-flash power plant model simulation. Detailed overview of model input parameters and the technical specifications of the five single-flash power plants are given in Table E.1 and Table E.2. It can be seen that the model simulations fall into the range of the thermal efficiencies from literature. Small deviations can be related to e.g. NCG content, silica scaling in equipment, turbine efficiencies, heat loss from equipment, power plant parasitic load (fans, pumps and gas extraction systems). In the case of Zarrouk and Moon (2014), well characteristics were taken into account as well, because thermal efficiency was calculated with the reservoir enthalpy. Then, also e.g. heat loss from the well to the surrounding geothermal rock and calcite scaling in the well can affect thermal efficiency.

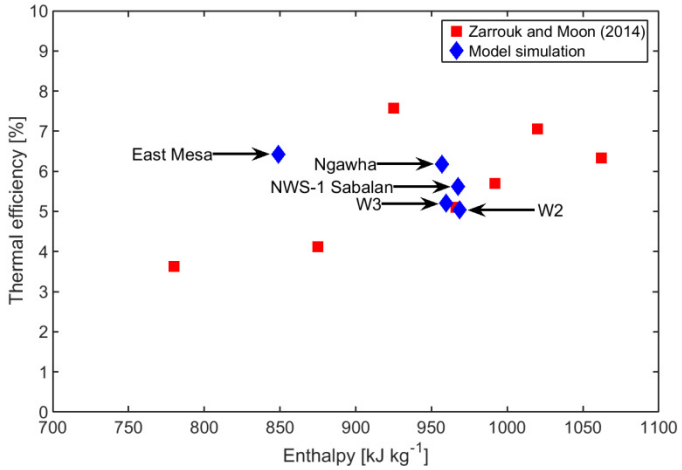


Figure 4.7: Thermal efficiency as a function of reservoir enthalpy. Model simulations of single-flash power plant show comparison with published thermal efficiencies according to Zarrouk and Moon (2014).

Additionally, a more specific validation of steam turbine and SE/C was performed. The mathematical model was compared to technical specification of the Cerro Prieto I (units 1 – 4) geothermal power plant (DiPippo, 2012). This power plant has much in common with the present model, because both power plants include a two stage steam ejector NCG system. Table E.3 presents the technical data of the power plants. The model input was adjusted to the known values of the Cerro Prieto plant. Unknown parameters (e.g. turbine and pump efficiencies) were set to the default model values. The outlet temperature of the condenser was varied to solve the energy balance for the condenser. The model simulation shows a great similarity to the Cerro Prieto power plant. Simulated net power is 37.78 MW compared to Cerro Prieto net power of 37.5 MW, simulated steam ejector motive fluid mass flow rate is 6.07 kg s^{-1} compared to 6.68 kg s^{-1} for the Cerro Prieto plant. The deviation can be due to assumed equipment efficiencies. Also, the assumed CO_2 content could cause this deviation. The NCG content (0.01308 wt% in 2005) in the Cerro Prieto I field was obtained from Ocampo-Diaz et al. (2005). The technical data however correspond to the operation of the Cerro Prieto power plant in 1980. A CO_2 content of 0.0161 wt% would exactly match the technical data from DiPippo (2012). It is not unusual that CO_2 content declines during the lifetime of a single-flash geothermal power plant, because most NCG are vented to the atmosphere.

It has to be said that it is not intended to give a false impression of the efficiencies of geothermal power plants. The calculation of thermal efficiency according to eq. (4.2) has only been done in this section, in order to compare and validate the model. Therefore, Table E.2 also presents the utilization efficiencies

based on pure water conditions, calculated with eq. (2.63). The utilization efficiencies of the model simulations are in the range of 24 – 33%. DiPippo (2012) has reported utilization efficiencies in the range of 28 – 40% for single-flash power plant. These are comparable taken into account that the efficiencies in DiPippo (2012) are based on large and optimized single-flash power plants (> 30 MW). Larger equipment is generally more efficient.

4.5.2. Binary Cycle Power Plant

The operation of the binary cycle power plant model in this work was validated with model calculations from Parada (2013) based on the "Berlin" binary cycle power plant located in El-Salvador. Parada (2013) have shown that isopentane is the most ideal working fluid for geothermal fluids of 180 ˚C and reinjection temperatures in the range of 60 – 140 ˚C. Geothermal fluids with temperatures in the range of 180 – 250 ˚C show even better performance. This is due to the relative high critical temperature (187.83 ˚C) of isopentane compared with other candidate working fluids. With the assumption of a subcritical working fluid, the exergy losses in the preheater and evaporator with isopentane are lower due to smaller temperature differences. Figure 4.8 presents the turbine power output as a function of reinjection temperature for a geothermal fluid entering the power plant at a temperature of 180 ˚C and mass flow rate of 221 kg s^{-1}. The condensing temperature of isopentane was 40 ˚C. The binary cycle power plant model in this study shows a good resemblance with the data from Parada (2013). In the model simulation the optimum turbine inlet pressure is calculated, based on an assumed pinch point temperature of 5 K and a saturated or superheated gas at the inlet of the turbine. All model input parameters obtained and adopted from Parada (2013) are presented in Table E.4.

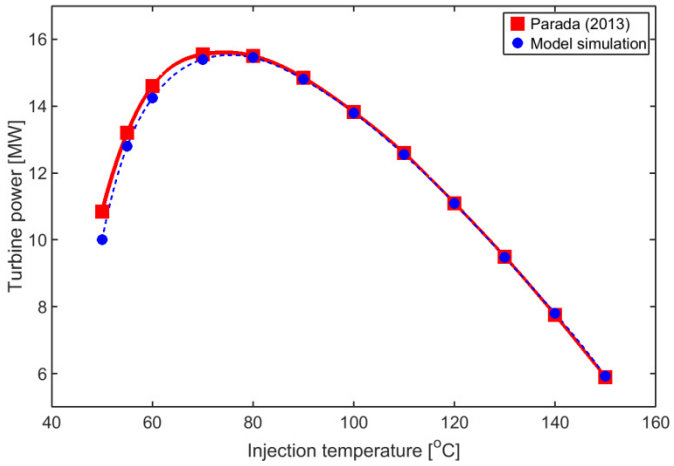


Figure 4.8: Turbine power output as a function of reinjection temperature for a geothermal fluid entering the power plant at a temperature of 180 ˚C and mass flow rate of 221 kg s^{-1}. The binary cycle working fluid was isopentane with a condensing temperature of 40 ˚C.

The binary cycle power plant has not been validated based on the thermal efficiencies, because the inlet temperatures for binary cycle power plants reported in literature are generally below 180 ˚C. For those temperatures isopentane is not necessarily the optimum working fluid. The present model does not contain a library for other working fluids. Additionally, the working fluids used at different binary cycle power plants is difficult to find. This would make the validation more arbitrary than the validation of the single-flash power plants.

4.6. Production Well Model Sensitivity Analysis

4.6.1. Sensitivity of Model Input Parameters and Phenomena

In this section a sensitivity analysis on the two production well models is performed. Multiple simulations were carried out to examine the sensitivity of certain model input parameters or certain phenomena associated with fluid flow in a geothermal production well, in which the present model is simplified with a single assumption in comparison to the full model. The full model refers to the results, given in Table 4.6,

of the simulations of the well NWS-1 Sabalan-2 given in Section 4.1.2 without any simplifications. The absolute differences between wellhead temperature and wellhead pressure of the simplified model simulation and the full model are listed in Table 4.7.

By far the most important parameter in wellbore simulation is the hydrostatic pressure loss term, which can be seen by the assumption of no hydrostatic pressure loss. Directly related to this term is the density of the geothermal fluid, which is among others a function of composition of the geothermal fluid and whether it is in two-phase or not. From the sensitivity analysis of the fluid composition it can be seen that good knowledge of the geothermal fluid composition is crucial in designing production wells. Especially the CO_2 mass fraction has a major influence on the wellhead temperature. Therefore, it is important to apply an accurate GFP model to the production well model.

The mass flow rate and pipe roughness increase also affected the wellhead temperature and pressure significantly. Especially the pipe roughness is difficult to establish. The wall roughness of the production well can be heavily eroded, corroded and/or deposited with calcite. Another model parameter which can be related to scaling is the inner diameter. Locally the inner diameter can be decreased affecting the increasing the pressure drop at that point.

The energy loss related parameters did not seem to have much effect on the pressure or temperature at the wellhead. It must be noted that the NWS-1 Sabalan-2 well flashes almost at the bottom hole depth. The fact that the wellhead temperature or pressure do not differ much from the full model, does not necessarily mean that the model parameters can be neglected. One important reason is the location of the flashing depth. If heat transfer to the surroundings is neglected, the flash depth increases. On the other hand, pressure losses are then decreasing. In the end, this can be balanced and no real deviation at the wellhead can be seen, while the pressure and temperature profiles in the well are different.

4.6.2. Sensitivity of Segment Length

Additionally, the sensitivity of the segment length on wellhead temperature and pressure was examined. Figure 4.9 and Figure 4.10 show the results for the East-Mesa 6-1 well and the W3 well, respectively. These wells were specifically chosen, because the geothermal fluid composition shows extremes. The geothermal fluid in the East-Mesa 6-1 well was modelled as pure water, while the geothermal fluid in the W3 well has been containing high CO_2 mass fractions. It can be seen in Figure 4.9 that the wellhead temperature is very dependent on segment length. In this case pure water was boiling according to its vapor pressure and temperature. Therefore, relatively small variations in pressure cause large variations in temperature at these low pressures. Additionally, the flashing depth is less accurate with large segment lengths, which causes an error on the pressure loss around those segments. Figure 4.10 shows much less dependence of wellhead pressure and wellhead temperature on the segment length. In this particular case a high CO_2 mass fraction was present in the geothermal fluid. This behavior is best explained by Figure 2.14. At the beginning of the flashing process, small enthalpy losses causes minor temperature variations, when large CO_2 mass fractions are present in the geothermal fluid. The order of magnitude of the pressure loss is approximately equal to the pure water case in Figure 4.9.

Although smaller segment lengths show a higher accuracy, it is outweighed by the additional computational time. Still, it is recommended to check the effect of segment length on the hydraulic and thermal behavior of the geothermal fluid in the production well for every particular case, when high accuracy is required. A default segment length of 20 m is recommended. This segment length was applied to the simulations executed in the results (Chapter 5).

4.7. Power Plant Model Sensitivity Analysis

In this section a sensitivity analysis on the geothermal power plant model is discussed. Multiple simulations were carried out to examine the sensitivity of certain model input parameters. The absolute differences between power output of the modified model simulations and the initial model simulation are listed in Table 4.8. The output of the simulations of production well NWS-1 Sabalan-2 functions as input for these simulations. A complete overview of the model input parameters for the full model simulation of the single-flash power plant and the binary cycle power plant has been given in Table E.5 and Table E.6, respectively.

Table 4.6: Wellhead pressure and wellhead temperature for the simulation of the full model for the production well without gas lift and the production well with gas lift.

Assumption	Self-flowing P_{wh}, bar	Self-flowing T_{wh}, ˚C	Gas lift P_{wh}, bar	Gas lift T_{wh}, ˚C
Full model	4.25	145.80	10.42	176.71

Table 4.7: Sensitivity of model input parameters and physical phenomena associated with fluid flow in a geothermal production well. The pressure- and temperature differences are absolute compared to the full model simulation from well NWS-1 Sabalan-2 given in Table 4.1.

Assumption	Changed parameter value	Self-flowing ΔP_{wh}, bar	Self-flowing ΔT_{wh}, ˚C	Gas lift ΔP_{wh}, bar	Gas lift ΔT_{wh}, ˚C
1. Full model		0	0	0	0
Geothermal fluid properties					
2. No CO_2 content	0.004 → 0, kg kg^{-1}	−0.87	−8.08	−0.54	−1.70
3. No NaCl content	0.005 → 0, kg kg^{-1}	0.04	0.17	0.11	0.30
4. No CO_2 and NaCl content	2 and 3, kg kg^{-1}	−0.91	−8.60	−0.44	−1.41
5. CO_2 content + 100%	0.004 → 0.008, kg kg^{-1}	1.82	13.43	−0.25	−1.74
6. NaCl content + 100%	0.005 → 0.01, kg kg^{-1}	−0.67	−5.09	−0.10	−0.20
7. CO_2 and NaCl content + 100%	See assumption 5 and 6	1.69	12.76	−0.50	−2.67
8. Mass flow rate + 10%	30 → 33, kg s^{-1}	−0.34	−3.00	−0.76	−2.46
Production well characteristics					
9. Pipe roughness increase	1.5×10^{-6} → 1.8×10^{-4}, m	−0.91	−8.55	−0.34	−1.3
10. Inner diameter decrease	0.2244 → 0.2200, m	0.60	−5.47	−0.12	−0.40
11. Production time decrease	100 → 10, h	0.01	0.05	−0.03	−0.30
Pipe flow characteristics					
12. No heat flow to surroundings	$d\dot{Q} = 0$	−0.01	−0.04	0.03	0.30
13. No potential energy loss	$dE_p = 0$	−0.13	−1.13	0.37	1.70
14. No frictional pressure loss	$dP_f = 0$	0.42	3.44	0.52	2.00
15. No kinetic pressure loss	$dP_k = 0$	0.04	0.35	0.06	0.15
16. No hydrostatic pressure loss	$dP_{hs} = 0$	55.72	73.04	N/A	N/A
Rock characteristics					
17. Thermal conductivity + 100%	1.5 → 3, W m^{-1}K^{-1}	0.00	0.04	0.00	0.10
18. Thermal diffusivity + 100%	1.2E-6 → 2.4E-6, W m^{-1}K^{-1}	0.00	-0.01	0.01	0.10
Drift-flux model					
19. Rouhani and Axelsson (1980)		0.00	0.00	0.00	0.00
20. Hasan et al. (2010)		3.85	25.14	4.52	18.30
21. Dix (1971)		−1.27	−12.50	−1.63	−8.90
22. Nicklin (1961) [1]		> -4.25	> -36.60	−6.32	−54.39
23. Toshiba		−0.58	−5.25	−0.71	−3.70

[1] The pressure in the wellbore was already below atmospheric before it reached the wellhead, because the pressure loss in the wellbore is too high.

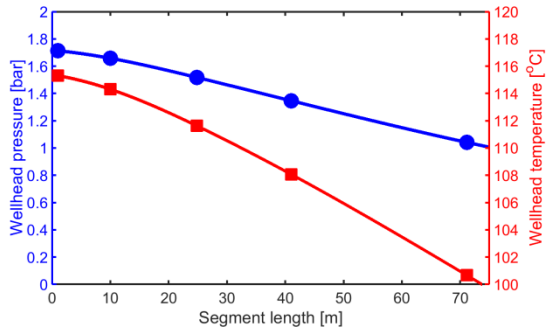


Figure 4.9: Sensitivity analysis on the segment length as a function of wellhead pressure (left) and wellhead temperature (right) for the East-Mesa 6-1 production well. The characteristics of this well have been described in Table 4.1.

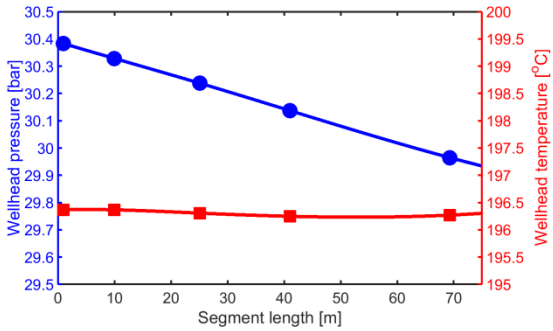


Figure 4.10: Sensitivity analysis on the segment length as a function of wellhead pressure (left) and wellhead temperature (right) for the W3 production well. The characteristics of this well have been described in Table 4.2.

Table 4.8: Sensitivity of model input parameters for the geothermal power plant. The net power output differences are absolute, compared to the initial model input parameters for the geothermal power plant model, described in Table E.5 and Table E.6.

Assumption	Changed parameter value	Single-flash power plant ΔW_{net}, MW	Binary cycle power plant ΔW_{net}, MW
1. Full model		0 (W_{net} = 1.57)	0 (W_{net} = 1.63)
Power plant characteristics			
2. Outlet pressure turbine increase	0.0738 → 0.1, bar	0.12	N/A [1]
3. No turbine losses		0.25	0.36
4. No pump losses		0.01	0.02
5. No generator losses		0.05	0.06
6. Outlet temperature condenser decrease	37 → 35, ˚C	0.11	0.03
Geothermal fluid properties			
7. No CO_2 content		0.65	−0.05
8. No NaCl content		0.06	0.01
9. No CO_2 and NaCl content		0.66	−0.03

[1] Outlet pressure turbine and outlet temperature condenser are functions of each other for the binary cycle power plant, because of the pure working fluid.

The sensitivity analysis on the geothermal power plants shows some interesting results. The single-flash plant net power increased with increasing outlet pressure of the turbine. Also, the power output increased by decreasing the condenser outlet temperature. Both of these results are most likely caused by the CO_2 content present in the stream and the demand for motive fluid to remove it from the condenser. This requires optimization for every single power plant. For binary cycle power plants, this is not the case and the power output is therefore less affected by these assumptions. The same trend can be seen with the assumption of no CO_2 content. The single-flash power plant power output increases significantly, because there is no need of a steam consuming SE/C.

Summarizing, the outcome of the simulation is affected by many parameters and assumptions. The model validation in Sections 4.1 and 4.5 shows that if enough details of the geothermal field and power plant are known the predictive accuracy is good. In the next chapter, more research on geothermal fluid composition and gas lift mass flow rate is discussed on the basis of a hypothetical case.

5

RESULTS AND DISCUSSION OF A HYPOTHETICAL CASE

This chapter discusses the results of a hypothetical geothermal system. It is partly derived from existing geothermal systems in order to stick to reality as much as possible. Section 5.1 gives an overview of the assumed constant model input parameters and the variable model input parameters. In Section 5.2 the results of the simulations are presented and discussed. Thereafter an optimization of a high potential scenario, derived from the results of the hypothetical case, is discussed in Section 5.3. Finally, this chapter is concluded with a comparison of a production well with an electrical submersible pump to a production well with gas lift in Section 5.4.

5.1. Model Input Parameters

The number of input parameters that can be varied are numerous. Therefore, a selection was made on the researched input parameters. The production well characteristics were expected to have minor influence on the difference between net power output of the single-flash power plant and binary cycle power plant. In this comparison production well dimensions were equivalent for both power plants, with the exception of the gas lift duct. The injection well parameters were equal to the self-flowing production well parameters for both power plants. The pressure and temperature of the geothermal fluid at reservoir conditions were 159 bar (hydrostatic pressure) and 250 ˚C (maximum temperature), respectively. The mass flow rate was 30 kg s^{-1}. The true vertical depth of the production well and injection well was 2000 m. Table F.1 and Table F.2 show a detailed overview of all the model input parameters of this hypothetical case.

Table 5.1 presents the examined model input parameters that were varied. The aim was to show the effect of geothermal fluid composition and mass flow rate of lift gas on power plant performance for the binary cycle power plant and compare this to a single-flash power plant for the exact same geothermal field. These varied model input parameters were applied to six power plant scenarios presented in Table 5.2 and Table 5.3 for a single-flash power plant and a binary cycle power plant, respectively. The two single-flash power plants were categorized on the gas extraction system outlined in Section 2.4.4.3. The four binary cycle power plants were categorized on the injection temperature and on the inlet conditions of the gas lift compressor. Two injection temperatures were assumed. One injection temperature was exactly equal to the injection temperature of the SF-1 power plant with corresponding reservoir conditions. The other injection temperature was assumed to be 70 ˚C, which has been based on the worldwide temperature range of injectates for hot water systems of 50 – 100 ˚C (Rivera Diaz et al., 2015). The two scenarios for the inlet conditions of the gas lift compressor have been discussed in Section 2.4.4.2. One scenario assumed an inlet pressure and temperature equal to the wellhead pressure and temperature. The other scenario assumed atmospheric conditions at the compressor inlet.

Finally, two other varied model input parameters should be mentioned, which were a function of a particular simulation. One varied parameter was the back pressure at the outlet of the single-flash power plant turbine, which was optimized for every simulation to obtain the optimum produced net power. Decreasing the back pressure of the turbine is associated with a smaller temperature difference between the cooling water and the gas mixture in the condenser, discussed in Section 2.4.4.3. For the SF-1 scenario, with a steam ejector/condenser, this would involve an increase in required motive fluid to extract the NCG from the condenser. For the SF-2 scenario, with a centrifugal compressor, this would involve a higher power output of the turbine accompanied by a higher required power input for the compressor. The other varied parameter was the depth of the gas lift valve. For every binary scenario (with a production well with

gas lift) the depth of the gas lift valve was assumed to be equal to the depth of the flashing horizon of the single-flash scenario (with a self-flowing production well) for the corresponding reservoir conditions.

Table 5.1: Varied model input parameters to examine the effect on power plant performance.

Input parameter	Value
Mass fraction NaCl in reservoir, kg kg^{-1}	0.025, 0.05
Mass fraction CO_2 in reservoir, kg kg^{-1}	0, 0.005, 0.01, 0.015, 0.02, 0.025, 0.034
Mass flow rate injected gas, kg s^{-1}	0.5, 1.0

Table 5.2: Single-flash power plant scenarios.

Scenario	Type of power plant	NCG extraction system
SF-1	Single-flash	SE/C
SF-2	Single-flash	Centrifugal compressor

Table 5.3: Binary cycle power plant scenarios.

Scenario	Type of power plant	Injection temperature	Inlet conditions compressor
BC-1	Binary cycle	$T_{inj,BC-1} = T_{inj,SF-1}$	$(P,T)_{c1,BC-1} = (P,T)_{A,BC-1}$
BC-2	Binary cycle	$T_{inj,BC-2} = T_{inj,SF-1}$	$(P,T)_{c1,BC-1} = (P,T)_{atm}$
BC-3	Binary cycle	$T_{inj,BC-3} = 70$ ˚C	$(P,T)_{c1,BC-1} = (P,T)_{A,BC-3}$
BC-4	Binary cycle	$T_{inj,BC-4} = 70$ ˚C	$(P,T)_{c1,BC-1} = (P,T)_{atm}$

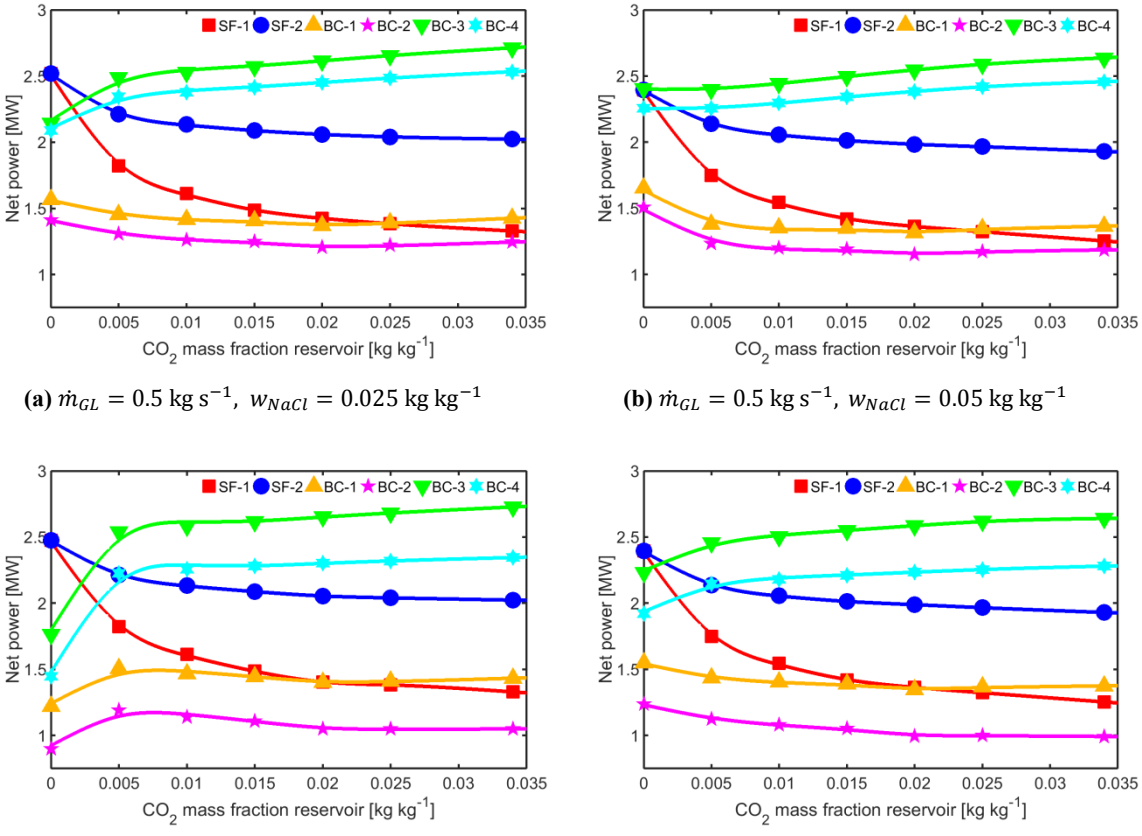
5.2. Results & Discussion

Figure 5.1, Figure 5.2 and Figure 5.3 show the net power, utilization efficiency and CO_2 mass emitted per MWh produced as a function of CO_2 mass fraction present in the geothermal fluid in the reservoir for every scenario discussed in Section 5.1.

5.2.1. Net Power

As can be seen in Figure 5.1, the net power generally shows a recurring trend in the plots (a – d). The net power for the single-flash power plant scenarios SF-1 and SF-2 decreases with increasing CO_2 mass fraction for all combination of mass flow rate of lift gas and NaCl mass fraction. For SF-1 this is caused by the increasing demand of motive fluid by the SE/C necessary to extract increasing amounts of NCG from the condenser. For SF-2 the degradation of net power is caused by the increasing auxiliary power demand to drive the centrifugal compressor to extract increasing amounts of NCG from the condenser. It can be seen that the auxiliary power demand of the centrifugal compressor for SF-2 is smaller than the equivalent net power reduction caused by the drained motive fluid for SF-1. Equivalent has been used in this context, because the SE/C itself does not consume power. The difference in net power between SF-1 and SF-2 increases with increasing CO_2 mass fraction. It is important to mention that a centrifugal compressor is more expensive to purchase and maintain than a SE/C. The difference between BC-1 and BC-2 and the difference between BC-3 and BC-4 is in both cases caused by the assumption of two scenarios for the CO_2 conditions at the inlet of the gas lift compressor. Logically, compression from atmospheric conditions requires more power. Therefore, the net power of BC-1 is higher than that of BC-2 and the net power of BC-3 is higher than that of BC-4. The difference between BC-1/BC-2 on one hand and BC-3/BC-4 on the other hand is caused by the assumption of two scenarios for the injection temperature. A lower injection temperature results in more heat transfer from the geothermal fluid to the working medium in the binary cycle. Consequently, a higher net power output is obtained with a lower injection temperature in this hypothetical case.

The difference in trends of the BC-1/BC-2 curves on one hand and the BC-3/BC-4 curves on the other hand are mainly caused by two varied model input parameters: the gas lift valve depth and the injection temperature. BC-3 and BC-4 show an increase in net power with increasing CO_2 mass fraction and constant injection temperature (70 ˚C). An increasing CO_2 mass fraction requires deeper installation of the gas lift valve, because flashing of geothermal fluid is induced at higher pressures. Generally, deeper in the production well the pressure and temperature is higher. Consequently, deeper installation of the gas lift valve results particularly in smaller hydrostatic pressure losses as a result of the decrease in density caused by the injected lift gas. Therefore pressure, temperature and liquid mass fraction at the production well wellhead is higher with deeper installation of the gas lift valve. Consequently, the net power increases for BC-3 and BC-4 with increasing CO_2 mass fraction. BC-1 and BC-2 show a rather constant net power for the CO_2 mass fraction range examined, with the exception of a CO_2 free geothermal fluid. The explanation related to the installation depth of the gas lift valve also applies to these two scenarios. Additionally, in these cases it has been assumed that the injection temperature varied with increasing CO_2 mass fraction in accordance with the injection temperature of SF-1. The injection temperature of SF-1 increases with increasing CO_2 mass fraction. This phenomenon is explained by the increasing production temperature of the self-flowing well associated with increasing CO_2 mass fraction. The liquid part of the produced fluid is sent to the injection well after being separated from the gas part in the cyclone separator and therefore the injection temperature for the single-flash power plant increases with increasing CO_2 mass fraction. Summarizing for BC-1 and BC-2, the increasing injection temperature decreases net power and the increasing gas lift valve depth increases net power, resulting in a rather constant net power as a function of CO_2 mass fraction. Also in these cases, it is important to mention that deeper installation of the gas lift valve induces most likely higher investment costs.



(a) $\dot{m}_{GL} = 0.5\ \text{kg s}^{-1},\ w_{NaCl} = 0.025\ \text{kg kg}^{-1}$ **(b)** $\dot{m}_{GL} = 0.5\ \text{kg s}^{-1},\ w_{NaCl} = 0.05\ \text{kg kg}^{-1}$

(c) $\dot{m}_{GL} = 1.0\ \text{kg s}^{-1},\ w_{NaCl} = 0.025\ \text{kg kg}^{-1}$ **(d)** $\dot{m}_{GL} = 1.0\ \text{kg s}^{-1},\ w_{NaCl} = 0.05\ \text{kg kg}^{-1}$

Figure 5.1: Net power as a function of CO_2 mass fraction present in the geothermal fluid at reservoir conditions. Four different situations were examined, in which the mass flow rate of lift gas and NaCl mass fraction at reservoir conditions were varied. The different plots represent the single-flash power plant and binary cycle power plant scenarios described in Table 5.2 and Table 5.3.

Generally, the net power produced by BC-3 and BC-4 becomes higher than that of SF-2 for a CO_2 mass fraction in the range of 0 – 0.005 kg kg^{-1}. If BC-3 or BC-4 is compared to SF-1, net power becomes higher

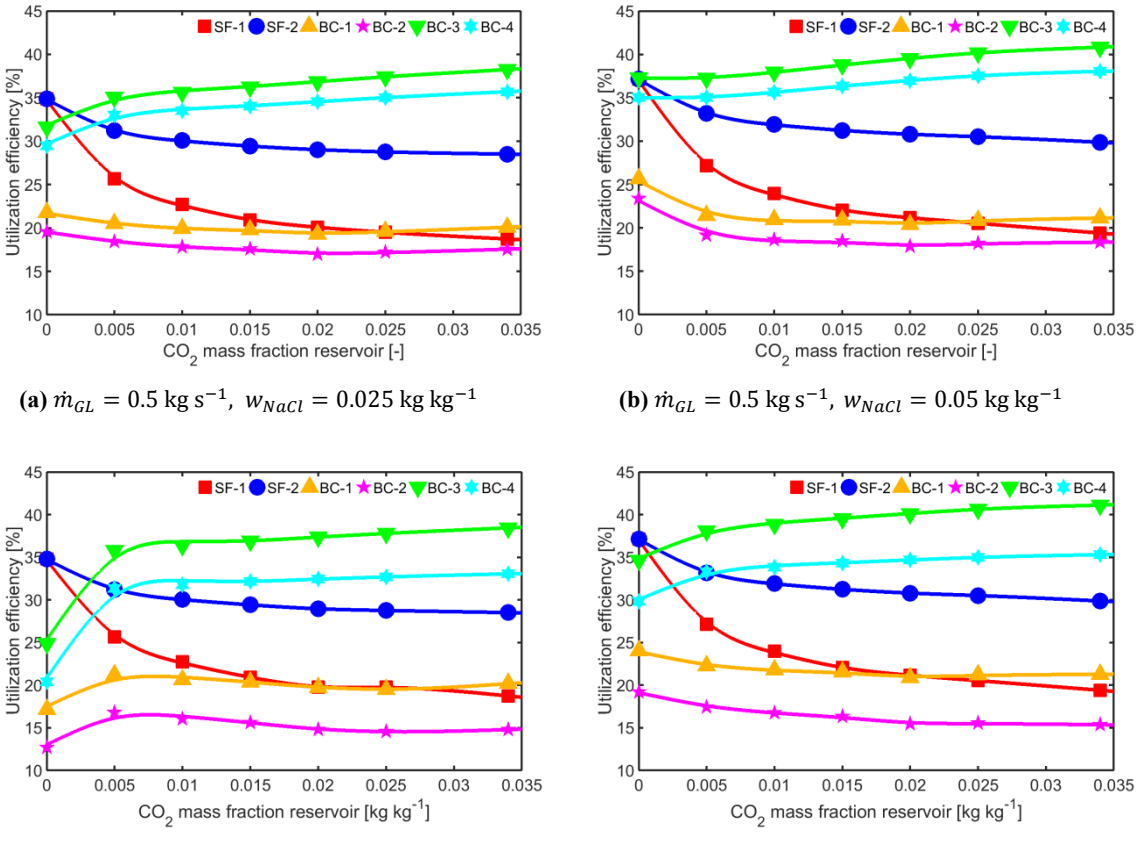
for a CO_2 mass fraction in the range of 0 – 0.003 kg kg^{-1}. It can be seen that the injection temperature of the binary cycle has a major influence on the net power. If the injection temperature is equal to SF-1, as in BC-1 and BC-2, a binary cycle is most likely unable to compete with a single-flash power plant. Also, it has to be kept in mind that the investment costs for binary cycles are generally higher than for single-flash plants.

Finally, if (a – d) of Figure 5.1 are compared to each other it is striking that the conditions examined in (c) results in the highest net power. It seems that a higher NaCl mass fraction decreases the net power. This is most likely caused by the increase in density associated with increasing NaCl mass fraction. Consequently, hydrostatic pressure loss in the production well increases and the potential extracted work by the turbine decreases. Additionally, for this hypothetical case a gas lift mass flow rate of 1.0 kg s^{-1} is in favour of a mass flow rate of 0.5 kg s^{-1}.

5.2.2. Utilization Efficiency

The utilization efficiencies for the different scenarios, presented in Figure 5.2, show a similar trend as the net power. This is rather straightforward, because utilization efficiency is among others a function of net power as can be seen in eq. (2.63). Since the reservoir conditions are almost constant, the difference in net power trends and utilization efficiency trends are negligible. The discussion in Section 5.2.1 can also be applied to the utilization efficiency.

Generally, the utilization efficiency of BC-3 and BC-4 becomes higher than that of SF-2 for a CO_2 mass fraction in the range of 0 – 0.005 kg kg^{-1}. If BC-3 or BC-4 is compared to SF-1, utilization efficiency becomes higher for a CO_2 mass fraction in the range of 0 – 0.003 kg kg^{-1}. Utilization efficiencies for BC-3 and BC-4 are approximately in the range of 30 – 45 % for the entire CO_2 range examined, while for SF-1 and SF-2 the utilization efficiency decreases for increasing CO_2 mass fraction.



(a) $\dot{m}_{GL} = 0.5$ kg s^{-1}, $w_{NaCl} = 0.025$ kg kg^{-1}

(b) $\dot{m}_{GL} = 0.5$ kg s^{-1}, $w_{NaCl} = 0.05$ kg kg^{-1}

(c) $\dot{m}_{GL} = 1.0$ kg s^{-1}, $w_{NaCl} = 0.025$ kg kg^{-1}

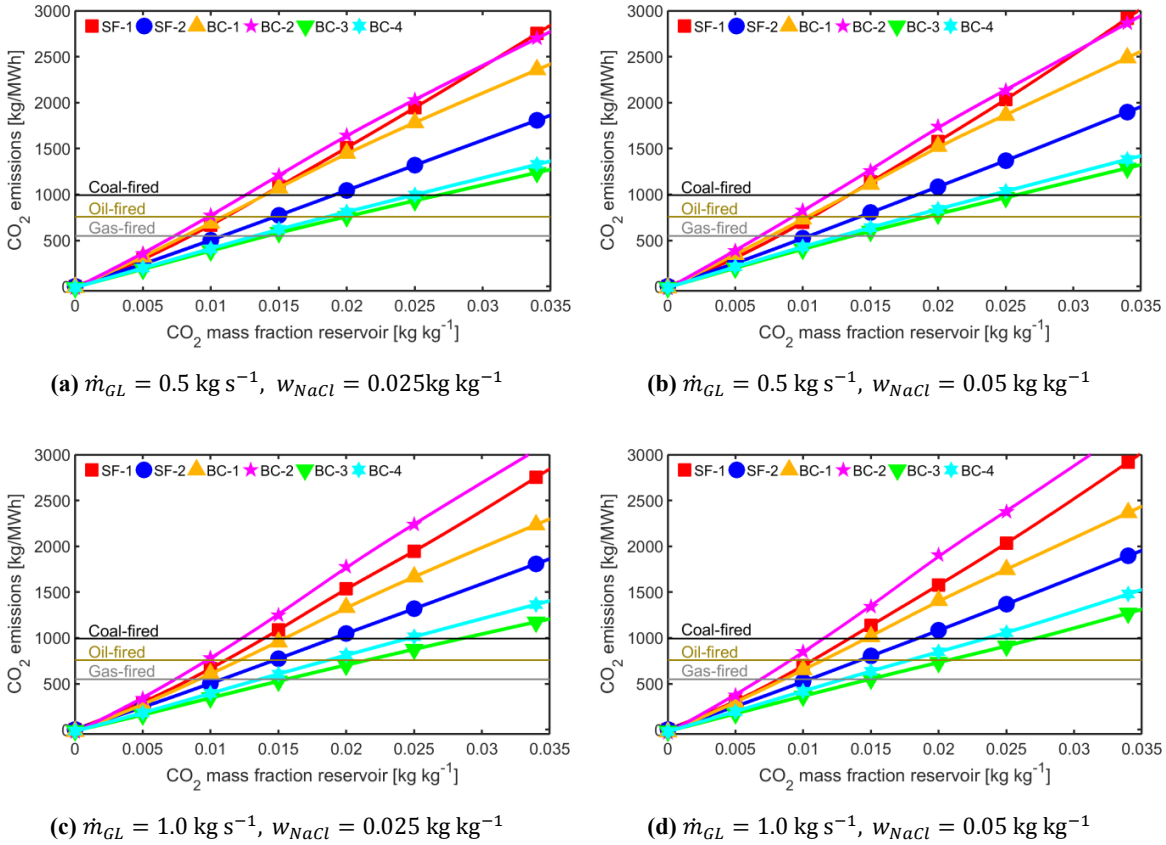
(d) $\dot{m}_{GL} = 1.0$ kg s^{-1}, $w_{NaCl} = 0.05$ kg kg^{-1}

Figure 5.2: Utilization efficiency as a function of CO_2 mass fraction present in the geothermal fluid at reservoir conditions. Four different situations were examined, in which the mass flow rate of lift gas and NaCl mass fraction at reservoir conditions were varied. The different plots represent the single-flash power plant and binary cycle power plant scenarios described in Table 5.2 and Table 5.3.

5.2.3. CO_2 Emissions

Figure 5.3 shows the mass of CO_2 emitted per MWh produced for every scenario assumed in Section 5.1. It can be seen that with increasing CO_2 mass fraction in the reservoir, the CO_2 emissions increase. In accordance with the net power and the utilization efficiency of BC-1 and BC-2 in comparison to the single-flash power plants, BC-1 and BC-2 do not perform significantly better than SF-1 taking the CO_2 emissions into consideration. SF-2 even outperforms BC-1 and BC-2. On the other hand, if there is a possibility of decreasing the injection temperature in the binary cycle power plant significantly, this scenario gains in interest. BC-3 and BC-4 approximately decrease the CO_2 emissions per MWh by 40 – 50% compared to SF-1, and approximately 25% compared to SF-2. If there is a possibility to lower to injection temperature from 70 ˚C, the binary cycle power plant with a production well with gas lift becomes even more favourable compared to a single-flash power plant with a self-flowing production well.

Again it can be observed that with lower NaCl mass fraction in the reservoir power plant performance related to CO_2 emissions increases in all scenarios for the exact same conditions. This can be seen by comparing (a) to (b) and (c) to (d). Furthermore, a gas lift mass flow rate of 1.0 kg s^{-1} induces less CO_2 emissions than a gas lift mass flow rate of 0.5 kg s^{-1} for the exact same conditions in case of BC-3. This can be seen by comparing (a) to (c) and (b) to (d).



(a) $\dot{m}_{GL} = 0.5\ \text{kg s}^{-1},\ w_{NaCl} = 0.025\text{kg kg}^{-1}$ **(b)** $\dot{m}_{GL} = 0.5\ \text{kg s}^{-1},\ w_{NaCl} = 0.05\ \text{kg kg}^{-1}$

(c) $\dot{m}_{GL} = 1.0\ \text{kg s}^{-1},\ w_{NaCl} = 0.025\ \text{kg kg}^{-1}$ **(d)** $\dot{m}_{GL} = 1.0\ \text{kg s}^{-1},\ w_{NaCl} = 0.05\ \text{kg kg}^{-1}$

Figure 5.3: Mass flow rate of CO_2 emitted to the atmosphere as a function of CO_2 mass fraction present in the geothermal fluid at reservoir conditions. Four different situations were examined, in which the mass flow rate of lift gas and NaCl mass fraction at reservoir conditions were varied. The different plots represent the single-flash power plant and binary cycle power plant scenarios described in Table 5.2 and Table 5.3.

Finally, the comparison has been made with three conventional power plants, namely the coal-fired power plant, the oil-fired power plant and the gas-fired power plant. The average values presented in Figure 5.3 have been adopted from DiPippo (2012). It can be seen that SF-2 emits more CO_2 per MWh than a gas-fired power plant for geothermal fluids with a CO_2 mass fraction > 0.011 kg kg^{-1}. For SF-1, BC-1 and BC-2 this is even the case for lower CO_2 mass fraction. BC-3 and BC-4 outperforms every power plant evaluated in this study for a CO_2 mass fraction < 0.015 kg kg^{-1}. It is interesting to observe that with higher CO_2 mass

fraction in the reservoir, geothermal power plants require somehow a CO_2 reinjection system in order to compete with conventional power plants.

5.3. Optimization of the Hypothetical Case

One particular high potential scenario with specific characteristics was selected to be optimized according to mass flow rate of lift gas and binary cycle injection temperature. This was performed for a CO_2 mass fraction of 0.01 kg kg^{-1}. This particular value has been chosen for a number of reasons. Studies have shown that total costs related to centrifugal compressors become lower than setups with a SE/C for a CO_2 mass fraction > 0.1 kg kg^{-1} of the gas fraction of the gas stream after the CS (Geremew, 2012). In this hypothetical case, this is equivalent to a CO_2 mass fraction of 0.02 kg kg^{-1} in the reservoir. Additionally, in Figure 5.3 it has been shown that a CO_2 mass fraction < 0.015 kg kg^{-1} is preferred when it comes to CO_2 emissions. Finally, it is aimed for to find the maximum performance difference between the single-flash system and the binary system. This difference is higher for a CO_2 mass fraction of 0.01 kg kg^{-1}, than for CO_2 mass fraction of 0.005 kg kg^{-1}. Table 5.4 shows the differences in net power, utilization efficiency and CO_2 emissions between BC-3 and SF-1. The maximum differences are highlighted in green. In this hypothetical case a mass flow rate of lift gas of 1.0 kg $^{-1}$ and a NaCl mass fraction of 0.05 kg kg^{-1} shows the highest potential. The hypothetical case with these model input parameters were optimized based on binary cycle injection temperature and mass flow rate of lift gas.

Figure 5.4 shows net power, utilization efficiency and CO_2 emissions as a function of binary cycle injection temperature in the range of 43 – 150 ˚C. In order to avoid confusion, Table 5.5 presents the two binary cycle scenarios, which now only differ from each other by the gas lift compressor inlet conditions. Therefore, BC-1 and BC-3 on one hand and BC-2 and BC-4 on the other hand have been joined. The comparison with SF-1 and SF-2 is made for the exact same model input parameters. The minimum theoretical injection temperature is 43 ˚C. This is based on the minimum pinch point temperature of 5 ˚C in the preheater and evaporator of the binary cycle. The temperature of the working medium after the condenser pump and at the inlet of the preheater is approximately 38 ˚C. It can be seen in (a) that the net power increases with decreasing injection temperature. The net power and utilization efficiency of BC-1/BC-3 are higher than the net power of SF-1 and SF-2 for injection temperatures < 121 ˚C and < 94 ˚C, respectively. For BC-2/BC-4, the net power and utilization efficiency of SF-1 and SF-2 become higher for injection temperatures < 104 ˚C and < 77 ˚C, respectively. It can be seen in (a) and (b) of Figure 5.4 that with an injection temperature of 43 ˚C, the net power and utilization efficiency of BC-1/BC-3 increase approximately by 95% and 45% compared to SF-1 and SF-2, respectively. And even for BC-2/BC-4, the net power and utilization efficiency increase approximately by 75% and 30% compared to SF-1 and SF-2, respectively.

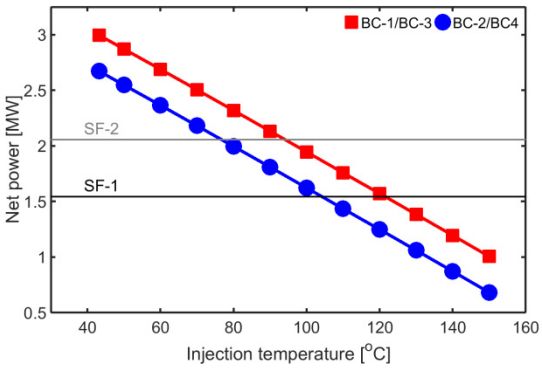
Table 5.4: Differences in net power, utilization efficiency and CO_2 emissions between BC-3 and SF-1.

Model input parameters	ΔW_{net}, MW	$\Delta \eta_u$, %	$\Delta \dot{m}_{CO2 \rightarrow atm}$, kg MWh^{-1}
$\dot{m}_{GL} = 0.5$ kg s^{-1}, $w_{NaCl} = 0.025$ kg kg^{-1}	0.92	12.9	− 281
$\dot{m}_{GL} = 0.5$ kg s^{-1}, $w_{NaCl} = 0.05$ kg kg^{-1}	0.90	14.0	− 291
$\dot{m}_{GL} = 1.0$ kg s^{-1}, $w_{NaCl} = 0.025$ kg kg^{-1}	0.97	13.7	− 322
$\dot{m}_{GL} = 1.0$ kg s^{-1}, $w_{NaCl} = 0.05$ kg kg^{-1}	0.96	14.9	− 331

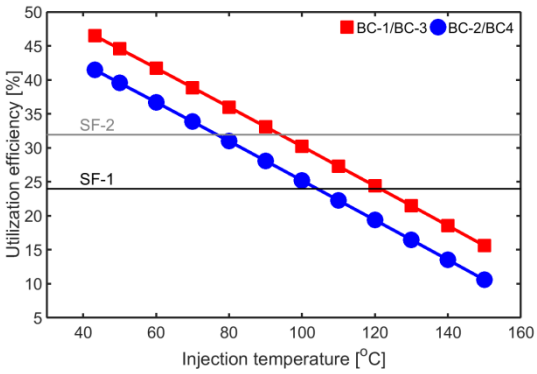
The CO_2 emissions of BC-1/BC-3 are lower than the CO_2 emissions of SF-1 and SF-2 for injection temperatures < 133 ˚C and < 110 ˚C, respectively. For BC-2/BC-4 the CO_2 emissions are lower than that of SF-1 and SF-2 for injection temperatures < 118 ˚C and < 92 ˚C, respectively. The CO_2 emission for BC-1/BC-3 and BC-2/BC-4 are more than twice as low as that for SF-1 and approximately 1.5 times as low as that for SF-2.

Table 5.5: Binary cycle power plant scenarios examined for binary cycle injection temperature optimization (see Figure 5.4).

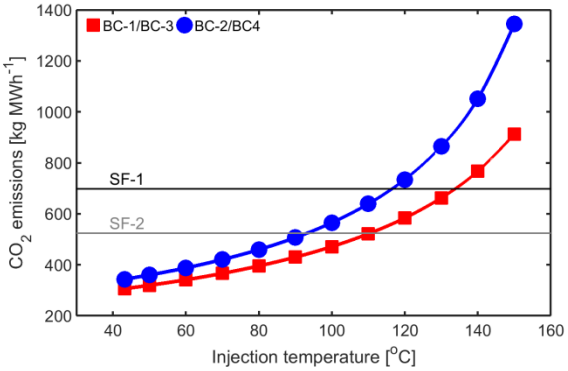
Scenario	Type of power plant	Injection temperature	Inlet conditions compressor
BC-1/BC-3	Binary cycle	Variable	$(P,T)_{c1,BC-1} = (P,T)_{A,BC-1}$
BC-2/BC-4	Binary cycle	Variable	$(P,T)_{c1,BC-1} = (P,T)_{atm}$



(a) Net power as a function of the injection temperature at the wellhead of the binary cycle injection well.



(b) Utilization efficiency as a function of the injection temperature at the wellhead of the binary cycle injection well.



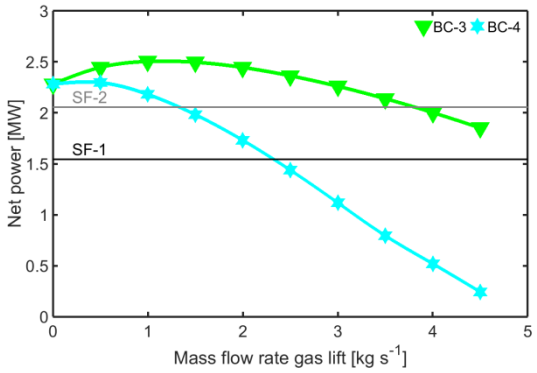
(c) CO_2 emmissions per produced MWh as a function of the injection temperature at the wellhead of the binary cycle injection well.

Figure 5.4: These plots show the (a) net power (b) utilization efficiency (c) CO_2 emissions of the binary cycle power plant as a function of the injection temperature for $\dot{m}_{GL} = 1.0$ kg s^{-1}, $w_{NaCl} = 0.05$ kg kg^{-1}, $w_{CO2} = 0.01$ kg kg^{-1}. Additionally, the optimized SF-1 and SF-2 for these conditions are shown. The difference between BC-1/BC-3 and BC-2/BC-4 is in this case only the conditions of the lift gas at the compressor inlet (see Table 5.5).

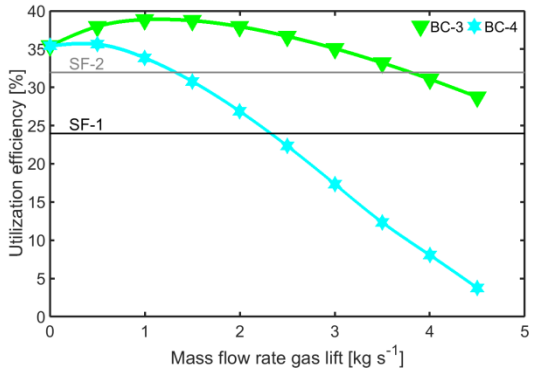
Figure 5.5 shows net power, utilization efficiency and CO_2 emissions as a function of mass flow rate of lift gas in the range of 0 – 4.5 kg s^{-1} for scenarios BC-3 and BC-4 (see Table 5.3). The optimization was performed for an injection temperature of 70 ˚C. The comparison with SF-1 and SF-2 was made for the exact same model input parameters. It can be seen in (a) and (b) that the optimum mass flow rate of lift gas related to net power and utilization efficiency for BC-3 lies between 1 – 1.5 kg s^{-1}. For BC-4, the optimum lies between 0 – 0.5 kg s^{-1} for the net power and utilization efficiency. The difference between BC-3 and BC-4 can be explained by the gas lift compressor inlet conditions. In case of BC-3, the wellhead pressure increases with increasing mass flow rate of lift gas, because the hydrostatic pressure loss in the production well decreases. Therefore, according to the assumption of compressor inlet conditions equal to wellhead conditions, the compression ratio decreases. This is due to the fact that the injection pressure at the gas lift valve does not change. Additional changes are related to the mass flow rate through the compressor, which increases with increasing mass flow rates of lift gas. Finally, also pressure losses due to friction in the gas lift duct are a function of mass flow rate of lift gas, because the dimensions of the duct in this study has not

been optimized and has been assumed constant in all simulations. These properties cause an optimum for BC-3 as it is. In case of BC-4, this optimum lies closer to 0 kg s^{-1}. This is mainly caused by the assumption of the atmospheric inlet conditions of the wellhead. Since, the compression ratio is in this case equal for all examined mass flow rates of lift gas; the required gas lift compressor power is mainly caused by the mass flow rate through the compressor.

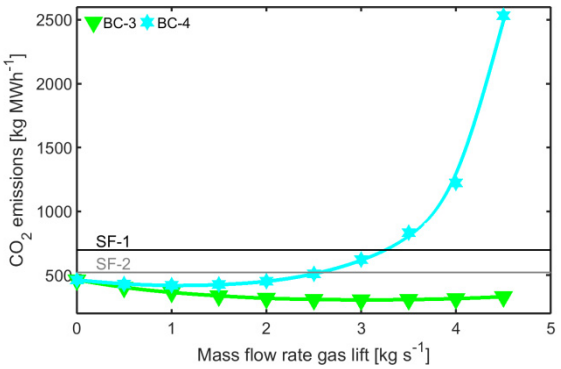
The optimum mass flow rate of lift gas related to CO_2 emissions is approximately 3.0 kg s^{-1} and 1.0 kg s^{-1} for BC-3 and BC-4, respectively. The reason that the optimum shifts to a higher mass flow rate of lift gas in comparison to the net power and the utilization efficiency can be found in the increasing wellhead pressures resulting from higher mass flow rates of lift gas. The dissolution of CO_2 is among others a function of pressure, with increasing pressures at the wellhead more CO_2 stays in the liquid solution that is sent to the evaporator and preheater. There the CO_2 stays dissolved and it is eventually reinjected in the reservoir. Since the CO_2 emissions are also a function of net power, BC-4 in (c) shows a large increase for higher mass flow rates of lift gas, because the net power reduces significantly as can be seen in (a).



(a) Net power as a function of the mass flow rate of lift gas (CO_2).



(b) Utilization efficiency as a function of the mass flow rate of lift gas (CO_2).



(c) CO_2 emmissions per produced MWh as a function of the mass flow rate of lift gas (CO_2).

Figure 5.5: These plots show the (a) net power (b) utilization efficiency (c) CO_2 emissions of the binary cycle power plant (BC-3 and BC-4) as a function of the mass flow rate of lift gas for $T_{inj} = 70$ ˚C, $w_{NaCl} = 0.05$ kg kg^{-1}, $w_{CO2} = 0.01$ kg kg^{-1}. Additionally, the optimized SF-1 and SF-2 for these conditions are shown. The difference between BC-3 and BC-4 is in this case only the conditions of the lift gas at the compressor inlet.

5.4. Electrical Submersible Pump Versus Gas Lift

From the literature survey on artificial lift in wells, discussed in Section 2.2, it has been concluded that gas lift is the most suitable lifting technique for temperatures in the range of 200 < T < 250 ˚C. Nevertheless, it is interesting to compare gas lift in a geothermal well to the use of an electrical submersible pump (ESP) to lift the geothermal fluids. Therefore at first, the hydraulic and thermal behavior of the self-flowing production well and the production well with gas lift was compared to a production well that is pressurized with an ESP. The relevant model input parameters for a production well with an ESP were exactly equal to

the other two production wells for the optimized case in Section 5.3, which means a CO_2 mass fraction of 0.01 kg kg^{-1} and a NaCl mass fraction of 0.05 kg kg^{-1}. The mass flow rate of gas lift was 1.0 kg s^{-1} and the gas lift valve depth was 1220 m. The ESP was installed at the same depth. In case of an ESP, two scenarios were distinguished: ESP-1 and ESP-2. In scenario ESP-1, the geothermal fluid is pressurized to such an extent that the wellhead pressure of ESP-1 is equal to the wellhead pressure of the production well with gas lift. In scenario ESP-2, the geothermal fluid is pressurized to such an extent that the wellhead pressure is just above the degassing pressure. This means that in ESP-2, the geothermal fluid does not flash. In case of ESP-1 the geothermal fluid was pumped from 66.4 bar to 106 bar. While in case of ESP-2 the fluid was pumped from 66.4 bar to 165 bar. The required pumping power was calculated by eq. (2.60), from which the change in enthalpy was calculated. The isentropic efficiency of the ESP was assumed to be 65% (Table 2.3). The hydraulic and thermal behavior is presented in Figure 5.6 and Figure 5.7, respectively. It can be seen that the temperature in case of ESP-2 degrades minimally, because flashing does not occur in the well. It is also observed that pumping the geothermal fluid, causes a small temperature increase. In case of ESP-1 flashing still occurs, but at a depth of 580 m.

Additionally, the behavior of the production well with gas lift is qualitatively validated by the trend of the plots in Figure 5.6 and Figure 5.7. At 1220 m, 1.0 kg s^{-1} CO_2 was injection. From that point on, the gas mass fraction and void fraction increases. Consequently, the associated density and therefore the hydrostatic pressure loss is smaller than in case of the self-flowing production well. Eventually the wellhead pressure is higher in the gas lifted well. Also, temperature degradation is smaller, because pressure loss is smaller. Temperature degradation can be a function of pressure loss during flashing of the geothermal fluid.

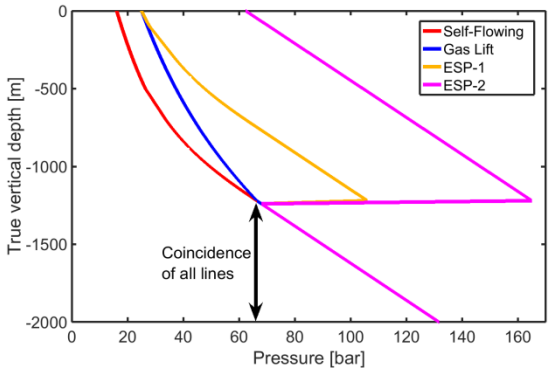


Figure 5.6: Pressure profiles in the production well as a function of true vertical depth for a self-flowing well, gas lifted well and two pumped wells.

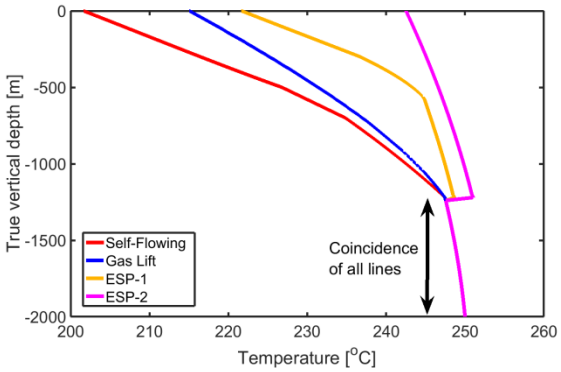


Figure 5.7: Temperature profiles in the production well as a function of true vertical depth for a self-flowing well, gas lifted well and two pumped wells.

Next, the performance of a binary cycle power plant connected to the production wells with an ESP was computed for the high potential scenario found in Section 5.2. The relevant input and output data is presented in Table 5.6.

It can be seen that the net power in case of ESP-1 and ESP-2 is significantly higher than the net power of BC-3 and BC-4. One cause is the larger mass flow rate of geothermal fluid from which heat is transferred in the binary cycle power plant. This can be seen by the gas mass fraction of 0.11 kg kg^{-1} for the gas lifted production well, while the gas mass fraction at the production well wellhead for ESP-1 and ESP-2 is 0.06 kg kg^{-1} and 0 kg kg^{-1}, respectively. In Figure 5.7, it can be seen that the temperature of ESP-1 and ESP-2 is higher than the gas lifted production well. The CO_2 emission in case of ESP-1 does not show a significant difference with BC-3 or BC-4. But in case of ESP-2, the geothermal fluid does not flash and all CO_2 remains dissolved in the geothermal fluid.

From these computed results, the ESP-2 scenario seems a really interesting case. At the current moment, ESP's that can deliver these powers (530 kW) at such depths (1220 m) and temperatures (250 ˚C) have not been in production yet. Another problem of ESP's is the short life expectancy for these harsh conditions.

Table 5.6: Model input parameters (green) and performance parameters (red) of six geothermal power plant scenarios for the model input parameters given in Table F.1 and Table F.2 for a reservoir system with w_{NaCl} = 0.05 kg kg^{-1}, w_{CO2} = 0.01 kg kg^{-1}.

Scenario	SF-1	SF-2	BC-3	BC-4	ESP-1	ESP-2
Type of production well	Self-flowing	Self-flowing	Gas lift	Gas lift	Pump (ESP)	Pump (ESP)
Mass flow rate of lift gas, kg s^{-1}	N/A	N/A	1.0	1.0	N/A	N/A
Depth gas lift valve or pump, m	N/A	N/A	1220	1220	1220	1220
Gas mass fraction wellhead, kg kg^{-1}	0.10	0.10	0.11	0.11	0.06	0
Type of power plant	Single-flash	Single-flash	Binary cycle	Binary cycle	Binary cycle	Binary cycle
Pressure turbine outlet, bar	0.134	0.092	1.373	1.373	1.373	1.373
NCG extraction system	SE/C	Compressor	N/A	N/A	N/A	N/A
Injection temperature wellhead, ˚C	129	118.6	70	70	70	70
Net power, MW	1.54	2.06	2.50	2.18	2.65	2.92
Utilization efficiency, %	24.0	31.9	38.9	33.9	41.1	45.3
CO_2 emission, kg MWh^{-1}	697	523	366	420	381	0
Power ESP, MW	N/A	N/A	N/A	N/A	0.19	0.53
Power gas lift compressor, MW	N/A	N/A	0.12	0.44	N/A	N/A

6

CONCLUSIONS AND RECOMMENDATIONS

6.1. Conclusions

According to the present study, it can be concluded that binary cycle power plants with artificial lift (gas lift) in geothermal wells are technical and thermodynamic feasible for reservoir temperatures up to 250 ˚C when it is compared to single-flash power plants with self-flowing geothermal wells for the hypothetical case proposed in this work.

In order to compare these two geothermal power plant systems it is vital to precisely model the thermal and hydraulic behavior of the geothermal fluid in the production wells. One of the components affecting the behavior of fluid in the artificially lifted geothermal well is the type of lift system, which has been studied in this work. The literature on artificial lift in geothermal wells and petroleum wells have shown that based on certain criteria the only possible lift method is gas lift. The two most important criteria are the maximum allowed operating temperature and the highest possible volumetric flow rates. Furthermore, the behavior of the fluid is greatly affected by its thermodynamic and transport properties. The fluid properties on its turn depend on the thermodynamic state and composition of the geothermal fluid. According to this study, it is allowed to assume that the geothermal fluid is a ternary solution containing H_2O, NaCl and CO_2. Generally, the upward flowing geothermal fluid in the wellbore exhibits hydrostatic pressure loss, frictional pressure loss, kinetic pressure loss, heat loss to the surroundings, frictional energy loss and kinetic energy loss. Additionally, phase change and flow pattern change arise from these losses along the wellbore and this affects thermal and hydraulic behavior of the fluid significantly. The one-dimensional, steady state numerical models of the production wells developed in this work consider all these losses and phenomena to solve the energy and momentum balances. Consequently, the fluid compositions and flow patterns along the wellbore are simulated by this comprehensive mathematical model in order to calculate the pressure and temperature profiles as a function of true vertical depth. The injection well has been modeled in accordance with the production well taking into account the relevant phenomena associated with reinjection of liquids.

Another crucial part of the geothermal system is the geothermal power plants. The amount of non-condensable gases (NCG) heavily influences the generated net power of single-flash power plants, because the NCG need to be extracted from the condenser for optimum performance. This goes at the expense of steam (steam ejector/condenser) or power (centrifugal compressor), depending on the type of extraction system. The performance of the basic binary cycle is among others a function of the type of working fluid. In literature, it has been shown that isopentane is the most suitable working medium for high temperature geothermal sources (> 200 ˚C). In order to make a fair comparison between a single-flash power plant and a binary cycle power plant, it is important to include all equipment demanding or generating power or equipment inducing phase change and/or fluid separation. The modeled equipment for the single-flash power plant are the cyclone separator, steam turbine, generator, condenser, condenser pump, steam ejector/condenser or centrifugal compressor, cooling water pump, make-up pump and injection pump. The binary cycle power plant model comprises the compressor, cyclone separator, evaporator, preheater, turbine, generator, condenser, condenser pump, make-up pump, cooling water pump and injection pump.

The crucial components of the mathematical model have been explicitly validated, qualitatively and quantitatively. It has been shown that the production well model is capable of predicting the thermal and hydraulic behavior of the production well accurately for the following property ranges: bottom hole pressures, 64 – 106 bar, bottom hole temperatures, 199 – 286 ˚C, mass flow rates, 7 – 50 kg s^{-1}, NaCl mass fraction, 0 – 10 wt%, CO_2 mass fractions, 0 – 12 wt%. These property ranges correspond to the field data

on which the production well model – self-flowing has been tested. From the sensitivity analysis of the production well, it can be concluded that properties like geothermal fluid composition, mass flow rate, pipe roughness, inner diameter, hydrostatic pressures loss and the choice of the drift-flux model can affect hydraulic and thermal behavior significantly. It can be concluded that the drift-flux model of Rouhani and Axelsson (1970) is the most accurate of drift-flux models based on the tested field data. The crucial part of the single-flash power plant model is the operation and interaction of the steam turbine, condenser and steam ejector/condenser. The computational results of the model have been validated by field data from a single-flash power plant with a similar setup. Also the binary cycle power plant model operating with isopentane as working medium has been validated with literature data of a binary cycle power plant with a similar setup.

The computational results of the hypothetical case proposed in this study show that for certain model input parameters the binary cycle power plant with gas lift in the geothermal well outperforms the single-flash plant with a self-flowing geothermal when it comes to net power, utilization efficiency and CO_2 emissions. The hypothetical case included a production well with among others a true vertical depth of 2000 m and an inner diameter of 0.245 m. The geothermal fluid had a mass flow rate of 30 kg s^{-1}, bottom hole pressure of 159 bar and bottom hole temperature of 250 ˚C. The NaCl mass fraction, CO_2 mass fraction and mass flow rate of lift gas were varied. Additionally, different scenarios were simulated in which the gas extraction system varied in the single-flash power plant model, and the injection temperature and inlet conditions of the gas lift compressor varied in the binary cycle power plant model. From the results it can be concluded that it is difficult for the binary cycle power plant with gas lift in the geothermal and an injection temperature of 70 ˚C to compete with optimized single-flash power plants with self-flowing wells for a CO_2 mass fraction < 0.005 kg kg^{-1} in the reservoir. Nevertheless, for a CO_2 mass fraction > 0.005 kg kg^{-1} the binary cycle power plant net power and utilization efficiency rises above those of the single-flash power plant. Furthermore, the CO_2 emissions of the binary cycle power plant are generally lower than the CO_2 emissions of the single-flash power plant. Finally, a high potential scenario in favor of the binary cycle power plant was optimized related to injection temperature and mass flow rate of lift gas for a CO_2 mass fraction of 0.01 kg kg^{-1} and NaCl mass fraction of 0.05 kg kg^{-1}. It can be concluded that for this hypothetical case an injection temperature of 43 ˚C and a mass flow rate of lift gas of 1.1 kg s^{-1} results in maximum net power and utilization efficiencies. In this optimized case the binary cycle power plant scenarios have shown approximately a 75 – 95% higher net power and utilization efficiency than those of the single-flash power plant with a steam ejector/condenser extraction system and a 30 – 45% higher net power and utilization efficiency than the single-flash power plant with a centrifugal compressor extraction system. Finally, it can be concluded that the binary cycle power plant is more environmental friendly when it comes to CO_2 emissions. For the optimized case, the CO_2 emission is approximately 100% and 33% lower than those of the single-flash power plant with a steam ejector/condenser and with a centrifugal compressor, respectively.

At the end of this study, the thermodynamic performance of a production well equipped with an ESP was evaluated. The results have shown that net power, utilization efficiency and CO_2 were all in favor of a binary cycle power plant with an ESP in the production well compared to the binary cycle with a gas lift. Although, ESP's for these conditions have not been in production yet, it can be concluded that it is important to develop such pumps.

6.2. Recommendations

This study has been initiated to explore the use of artificial lift in geothermal production wells. Although this work has been extensive already there is still much to investigate, because of the novelty of gas lift in a geothermal well. Future work should include a more comprehensive optimization of the complete geothermal power plant. Besides a thermodynamic optimization, an economic optimization will be necessary to examine the viability of gas lift in a geothermal well. In this section, critical commentary and recommendations have been given in order to bring this technique to the next level if possible.

In this work, it has been assumed that pure CO_2 is injected in the production well. CO_2 is recycled from the gas flow at the wellhead of the production well. This gas flow is a mixture of H_2O and CO_2. The consumed energy to separate CO_2 from H_2O has not been taken into account. Furthermore, the dimensions of the gas lift system have not been taken into account. Since it turned out according to the simulations that pressure and enthalpy in the gas lift duct is influenced minimally with a hydraulic diameter above 0.05 m. On the other hand, with an economic feasibility study dimensions of the production well/gas lift system and

additional equipment necessary above earth's surface will become increasingly important. An increase in drilling diameter entails an increase in investment costs.

Also the single-flash power plant should be economically optimized. As it has been discussed already in Section 2.4.4.3, NCG in the steam does not only influence the thermodynamic performance. A gas extraction system is needed to remove the NCG from the condenser. The degree of condensation depends among other things on the assumed outlet temperature of the condenser. Optimization of this process involves numerous parameters e.g. the amount of NCG, but also the available cooling medium, which is a site specific parameter. Therefore, it is recommended to conduct this optimization for a case study of a certain geothermal field where all details of the field are known. This optimization felt outside the scope of the present work, this work has only been a first set-up of the feasibility of artificial lift in geothermal wells.

The properties of the geothermal fluid are obtained from an MS Excel model. This model needed some adjustments to calculate two-phase flow properties. The error induced with geothermal systems with low CO_2 concentration was higher, because small deviations in temperature or pressure near the saturation pressure of H_2O can cause iterative problems. It has been shown in the model validation that this error was relatively small with CO_2 concentrations of 0.004 wt%. In future research with lower CO_2 concentrations it is recommended to evaluate the accuracy of the present model. Solutions to this potential flaw should be sought in an accurate equation of state describing the properties of a geothermal fluid or the use of commercial simulators.

In this work only one organic working medium for the binary cycle power plant has been evaluated. It may be of value to examine other working fluids. According to DiPippo (2012), propane, i-butane, n-butane, n-pentane and ammonia are other candidate working fluids for binary plants. There is the possibility that the optimum working medium depends on the thermodynamic state of the geothermal fluid and thereby is again a site specific optimization parameter. Also, environmental, safety and health properties are important to consider in the choice of a working fluid. Lastly, supercritical cycles can be considered, because it allows a better match between the cooling curve of the geothermal fluid and the working medium. This increases the exergy efficiency of the heat exchanger.

The mathematical model has been partly validated quantitatively with experimental data in literature. It has been aimed for to validate the model with random experimental data having significant different properties. This has been achieved in the present study, even though detailed and complete field data of the power plant were hard to find. Nevertheless, it is advised to collect more current experimental data in literature or from running geothermal power plants.

This work has particularly been a feasibility study of the thermodynamics of the geothermal power plant. In future work, the effect of injecting gas via a gas lift valve should be investigated. Calcite scaling can be a serious problem in well casings and it is directly related to the degassing of CO_2. The question after this study remains related to the chemical consequences at the gas lift valve when CO_2 is injected. Additionally, alternatives gases for CO_2 can be considered. Air is abundant, but it is potentially hazardous when methane is present in the geothermal fluid. Nitrogen can be assumed inert, but it is not available in advance, making it a more expensive alternative.

Finally, a recommendation towards the industry is made related to the development of electrical submersible pumps for high temperature and high pressure conditions.

A

SUPPLEMENTARY THEORIES

A.1. Types of Geothermal Power Plants

A.1.1. Double-Flash Steam Power Plant

The double-flash steam power plant can be seen as an upgrade of the previously described single-flash plant. Generally, it increases the power output by 15-25% for the same geothermal reservoir in comparison to a single-flash plant. However, the extra power output is at the expense of complexity, cost and maintenance. Figure 2.1 shows that double-flash plants have not been utilized for liquid-dominated high-enthalpy and vapor-dominated systems above 1850 kJ/kg. This is due to the fact that at sites with high enthalpy systems single-flash plants often serve the electricity demand of the area and there is no need for a double-flash plant. While on the contrary at sites with low- and medium-enthalpy systems single-flash plants cannot always meet the electricity demand of the area, while double-flash plants can. Typical utilization efficiencies are in the range of 35 – 45% (DiPippo, 2012; Zarrouk and Moon, 2014).

Figure A.1 shows a simplified double-flash steam power plant schematic. The double-flash plant lay-out resembles the single-flash power plant at some point. The main difference is that the liquid geothermal fluid coming from the cyclone separator is flashed for a second time to generate additional steam after a throttling valve (TV), but with a lower pressure than the steam delivered by the cyclone separator. The flasher (F) is a flash vessel, which separates the low-pressure steam from the liquid. The high-pressure steam (primary steam) is fed to a high-pressure turbine. Additionally, the low-pressure steam joins the primary steam before it expands in a low-pressure section of the turbine. In this way, more energy is extracted from the produced geothermal fluid.

As with a single-flash steam power plant, the risk of scaling in the production well, cyclone separator and moisture remover is present. The second flash reduces temperature and pressure even more, increasing the scaling potential in the flasher, water piping (WP) and injection well (IW).

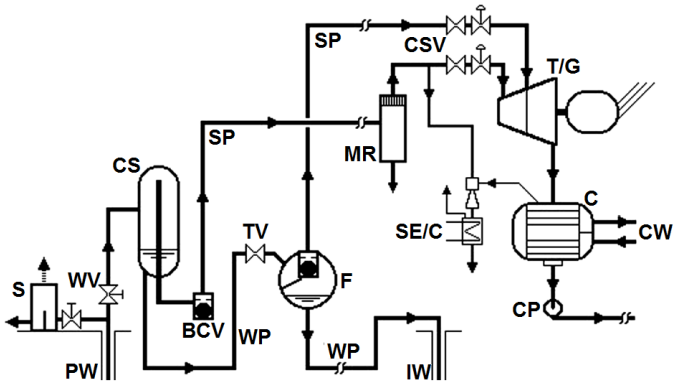


Figure A.1: Simplified double-flash power plant schematic (DiPippo, 2012).

A.1.2. Dry-Steam Power Plant

Dry-steam power plants were the first commercial geothermal power plants in operation. Although the number of dry-steam power plants accounts for only 10% of the total number of geothermal power plants (Figure 1.2), caused by the scarcity of high-enthalpy vapor dominated fields. There are only two major dry-

steam fields in the world, Lardarello (Italy) and The Geysers (U.S.). Nevertheless, the contribution to the total installed capacity is 23%. The average capacity rating of a dry-steam geothermal plant in the world was 45 MW_e in 2016 (Bertani, 2016). Typical utilization efficiencies are in the range of 50 – 65% (DiPippo, 2012).

A dry-steam power plant has many similarities with a single-flash unit (Figure A.2). They are essentially identical from the point where the steam enters the moisture remover (MR) to the point where the steam is reinjected into the injection well (IW). The difference is the use of a particulate remover (PR) in case of the dry-steam power plant in place of the cyclone separator in a single-flash plant. Dry steam at the wellhead valve (WV) is a prerequisite for this power plant. Dry-steam plants are not necessarily built at sites with dry-steam fields. Under the right conditions, the flashing process in the production well can ensure the delivery of dry steam to the wellhead valve. This depends on pressure loss due to friction, gravity and acceleration of the two-phase fluid. Geothermal reservoirs at relatively low depth and high temperature/high enthalpy are often suitable for dry-steam power plants. In Indonesia 34% of electric power is generated by dry-steam power plants, resulting from the volcanic environment (Bertani, 2016).

Dry-steam power plants are less complex and cheaper than the single-flash variant, since there is no remaining liquid geothermal fluid to cope with. Additionally, there is no mineral-laden fluid to dispose of which avoids the chance of fouling in equipment. The flashing in the production well can cause scaling. Another negative aspect of a dry-steam plant is that the NCG are released in its entirety to the atmosphere (DiPippo, 2012).

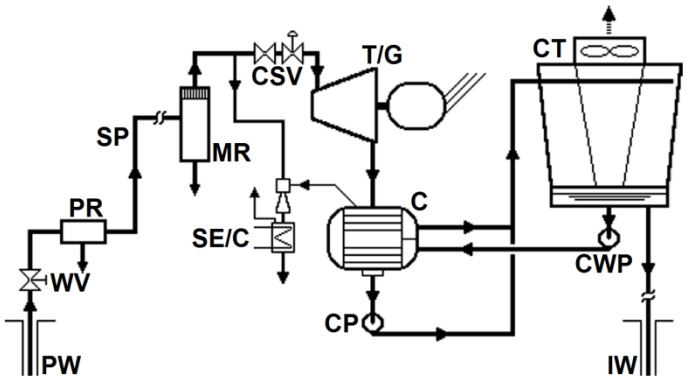


Figure A.2: Simplified dry-steam power plant schematic (DiPippo, 2012).

A.1.3. Hybrid Flash-Binary Cycle Power Plant

Flash-binary cycle power plants are an extension of the single-flash power plant and a variant to the double-flash plant. Instead of flashing the low pressure geothermal fluid again, the remaining liquid is used to heat a working fluid in a binary cycle. Generally, the binary cycle power plant is added to a single-flash power plant after a few years, if the demand for electricity increases and the geothermal reservoir has proven its consistency. In this way, the power output and conversion efficiency are increased. In Figure 1.2 flash-binary power plants have not been included, because these are subdivided in single-flash and binary cycle plants. There were 47 flash-binary units in operation in 2012, which was approximately 4% of the total installed capacity worldwide (DiPippo, 2012).

Figure A.3 gives a simplified schematic of a combined flash-binary power plant. It shows that the left side of the figure agrees with a single-flash plant (Figure 2.2). The right side agrees with a basic binary cycle power plant (Figure 2.3). The two-phase fluid entering the cyclone separator (CS) flashes partially. The steam is sent to the steam turbine, whereas the liquid geothermal fluid is transported to the binary cycle. The final temperature and pressure after flashing can be optimized to achieve the highest power output or efficiency.

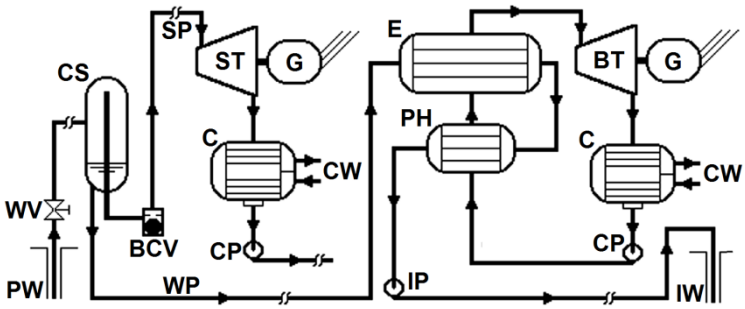


Figure A.3: Simplified combined flash-binary power plant schematic (DiPippo, 2012).

A positive aspect of flash-binary plants is the large operational experience of both single plants: single-flash and binary cycle. Also, the risk reduces by investments in stages. Additionally, there is no need of extra production or injection wells, which are a considerable part of the investment costs. On the other hand, investment costs are higher compared to other power plants. Negative aspects of the single-flash power plant and binary cycle power plant add up: scaling in the flash cycle and safety measures for the binary cycle plant is an issue (Van der Hoorn et al., 2012).

A.2. Binary System H_2O – NaCl

A.2.1. Saturation Pressure

In order to model the two-phase region the properties at saturated liquid condition as a function of P, T and w_{NaCl} are necessary. Dittman (1977) calculated brine saturation pressures as a percentage of the pure water saturation pressure at the same temperature with eq. (A.1) and incorporated it into various numerical codes. Table A.1 shows the corresponding coefficients valid in the range of 0-25 wt%.

$$P_{sat,b}(T) = a_1 \times P_{sat}(T) \tag{A.1}$$

Where $P_{sat.b}$ is the liquid saturation pressure of brine [bar], P_{sat} is the saturation pressure of pure water [bar] and T is the temperature [˚C].

Table A.1: Brine saturated pressure coefficients as a function of NaCl mass fraction (Dittman, 1977)

w_{NaCl}	a_1
5	0.969
10	0.934
15	0.894
20	0.847
25	0.794

It is assumed that the salt content in the vapor is negligible (Dittman, 1977). The saturated water vapor curve is provided by the IAPWS-IF97 equation of state from Wagner et al. (2002) available in FluidProp.

A.2.2. Density

Adams and Bachu (2002) reviewed seven different algorithms to calculate brine density. The Batzle and Wang (1992) algorithm has been found the most versatile and more accurate over a wider range of conditions compared to the others. It is valid for pressures up to100 MPa, temperatures in the range of 20-350 ˚C and salinities up to 320 g l^{-1}. Firstly, the freshwater density at different temperature and pressure conditions is calculated according to eq. (A.2).

$$\rho_w = [1 + 1 \times 10^{-6}(-80\,T - 3.3\,T^2 + 0.00175\,T^3 + 489\,P - 2\,TP + 0.016\,T^2P - 1.3 \times 10^{-5}\,T^3P - 0.333\,P^2 - 0.002\,TP^2)] \times 10^3 \quad \text{(A.2)}$$

And then the fresh water density is used in eq. (A.3) to calculate the brine density.

$$\rho_b = \rho_w + w_{NaCl}\{0.668 + 0.44\,w_{NaCl} + 1 \times 10^{-6}[300\,P - 2400\,Pw_{NaCl} + T(80 + 3\,T - 3300\,w_{NaCl} - 13\,P + 47\,Pw_{NaCl})]\} \quad \text{(A.3)}$$

In eq. (A.2) and (A.3) ρ_b and ρ_w are the brine and water density [kg m^{-3}], w_{NaCl} is the mass fraction [kg kg^{-1}], P is the pressure [MPa] and T is the temperature [˚C]. These correlations are solely applicable to liquid brine. In order to calculate the mixture density the volumetric-weighted average of the two phases is applied in eq. (A.4) (Hasan et al., 2010).

$$\rho_m = \rho_g \varepsilon_g + \rho_l(1 - \varepsilon_g) \quad \text{(A.4)}$$

Where ρ_m is the density in the two-phase region [kg m^{-3}], ρ_g is the density of the vapor phase [kg m^{-3}], ε_g is the cross-sectional void fraction [m^2 m^{-2}] and ρ_l is the density of the liquid phase [kg m^{-3}].

A.2.3. Viscosity

Adams and Bachu (2002) reviewed six different viscosity algorithms, from which the Kestin et al. (1981) has been found the most versatile. It is unfortunately only valid up to 150 ˚C. Philips (1981) and Batzle and Wang (1992) developed correlations without pressure terms. According to experimental results it has been concluded that even at 500 bar, viscosity only increased a few percent. Brine viscosity decreases rapidly with increasing temperature, but it is little affected by pressure. With increasing salinity, the viscosity increases. Palliser and McKibbin (1998b) used the algorithm from Philips (1981) for their extrapolation for higher temperatures up to 800 ˚C. The correlation to calculate the viscosity is a function of pressure, temperature and mass fraction. For temperatures up to 200 ˚C the Batzle and Wang (1992) algorithm deviates most from other algorithms. Philips (1981), which is identical to Palliser and McKibbin (1998b) for temperatures up to 350 ˚C, claimed that their correlation reproduces data to an average better than ± 2% for pressures in the range of 0.1 – 50 MPa, temperatures in the range of 10 – 350 ˚C and molalities in the range of 0 – 5 mol kg^{-1}. Therefore, the correlation from Philips (1981) is adopted in the present work (see eq. (A.5)).

$$\mu_b = \mu_w \times 10^{-3}[1 + 0.0816\,m - 0.0122\,m^2 + 0.000128\,m^3 + 0.000629\,T(1 - e^{-0.7\,m})] \quad \text{(A.5)}$$

In eq. (A.5) μ_b and μ_w are the dynamic viscosities for brine and water [Pa s], m is the molality [mol kg^{-1}] and T is the temperature [˚C]. This expression has been used to correct for salinity effects in various models that calculate flow in geothermal reservoirs (Adams and Bachu, 2002). Eq. (A.5) is only applicable to the liquid brine. The viscosity in the two-phase region is calculated by the mass-weighted average of the two phases (Hasan et al., 2010).

$$\mu_m = \mu_g \chi + \mu_l(1 - \chi) \quad \text{(A.6)}$$

Where μ_m is the dynamic viscosity in the two-phase region [Pa s], μ_g is the dynamic viscosity of the gas phase in Pa s, χ is the quality [kg kg^{-1}] and μ_l is the dynamic viscosity of the liquid phase [Pa s].

A.2.4. Specific Enthalpy

The enthalpy correlations have not been studied as extensively as density and viscosity. Dittman (1977) used a power curve fitted to experimental data to calculate the brine saturated liquid enthalpy as a function of temperatures up to 204 ˚C and mass fractions up to 25 wt% for liquid saturated conditions. The enthalpies for saturated liquid temperatures up to 316 ˚C were extrapolated using the power curve. Brine vapor enthalpies were obtained from pure water equations. Philips (1981) collected worldwide published experimental and calculated data for NaCl(aq). He published saturated liquid enthalpy values for

temperatures in the range of 0 – 300 ˚C and NaCl mass fraction in the range of 0 – 25 wt%. Pitzer et al. (1984) used experimental measurements of the enthalpy to derive a semi-empirical equation of the NaCl(aq) at constant pressures. The equation is valid for pressures in the range of P_{sat} – 1000 bar, temperatures in the range of 0 – 300 ˚C and molalities in the range of 0 – 6 mol kg^{-1}. Enthalpy values within this range were tabulated and published. Uncertainty estimation at 300 ˚C and 1000 bar is ± 20%, because the developed equations did not perfectly fit the experimental data at high temperatures and high pressures. At 200 ˚C, the uncertainty estimation is only ± 4% at its maximum. Palliser and McKibbin (1998b) proposed correlations for the specific enthalpy of brine as a function of temperature, pressure and mass fraction of NaCl based on various data sets. Their correlations cover the entire $T - P - x$ state-space and were specially designed for subroutines in numerical simulation programs (Palliser and McKibbin, 1998a). Correlations were developed at the boundaries of certain regions and linear interpolation was used to calculate the enthalpy between the boundaries. They attempted to derive correlations based on the tabulated values from Pitzer et al. (1984) in the subcritical liquid region. But since their reference state is significantly different compared to the other evaluated data sets, the data remained inconsistent with those other data sets. Therefore, in the region of temperatures of $T < T_c$ and pressures of $P > P_{sat}$, Palliser and McKibbin (1998b) decided that the enthalpies were independent of pressure and equal to the enthalpy at saturated liquid conditions $h_{l,sat}$ (eq. (A.7)).

$$h_{l,sat}(T,P) = h_{w,l,sat}(T) + \left[h_{l,SAT}(T) - h_{w,l,sat}(T)\right]\left[\frac{P_{w,sat}(T) - P}{P_{w,sat}(T) - P_{SAT}(T)}\right]^{1/1.4} \tag{A.7}$$

Where $h_{w,l,sat}(T)$ is the saturated liquid enthalpy of water [kJ kg^{-1}], $h_{l,SAT}(T)$ is the halite-saturated liquid enthalpy on the three-phase surface [kJ kg^{-1}] (eq. (A.8)), $P_{w,sat}(T)$ is the saturated liquid pressure [bar], $P_{SAT}(T)$ is the saturated pressure on the three-phase surface [bar] (eq. (A.9)).

$$h_{l,SAT}(T) = m_0 + m_1 T + m_2 T^2 + m_3 T^3 \tag{A.8}$$

$$P_{SAT}(T) = a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{A.9}$$

Table A.2 gives the coefficients m and a, T is the temperature [˚C] and t is a coefficient for temperature given by eq. (A.10).

$$t = \left({}^{T}/_{800}\right)^2 \tag{A.10}$$

Table A.2: Coefficient values for the correlations in eq. (A.8) and eq. (A.9) (Palliser and McKibbin, 1998b)

i	0	1	2	3	4	5
m	0.00000e0	3.57384e0	-3.79475e-3	1.59816e-6		
a		1.32729e1	3.18909e3	-7.24296e2	-8.15640e3	5.67834e3

Driesner (2007) developed a set of correlations for enthalpies of phases in the system H_2O – NaCl as a function of temperatures in the range of 0 – 1000 ˚C, pressures in the range of 1-5000 bar and compositions in the range of 0 – 1 mol mol^{-1}. The enthalpies agreed within 1 – 3% to other studies. He has shown that the correlations for enthalpy of Palliser and McKibbin (1998b) were substantially too low, because of the lack of pressure-dependence. The correlation of Driesner (2007) is given by eq. (A.11).

$$h_{sol}(T,P,x_{\mathrm{NaCl}}) = h_{\mathrm{H_2O}}(T_h^*,P) \tag{A.11}$$

Where h_{sol} is the enthalpy of the solution [J kg^{-1}], T is the temperature [˚C], P is the pressure [bar] and x_{NaCl} is the mole fraction [mol mol^{-1}]. The scaled temperature T_h^* [˚C] is given by eq. (A.12).

$$T_h^* = q_1 + q_2 T \tag{A.12}$$

Where q_1 and q_2 are coefficients given by eq. (A.13) and eq. (A.14), respectively.

$$q_1 = q_{10} + q_{11}(1 - x_{\mathrm{NaCl}}) + q_{12}(1 - x_{\mathrm{NaCl}})^2 \tag{A.13}$$

$$q_2 = q_{20} + q_{21}\sqrt{x_{\mathrm{NaCl}} + q_{22}} + q_{23}x_{\mathrm{NaCl}} \tag{A.14}$$

Where q_{10} and q_{20} are eliminated by the conditions $q_1 = 0$ and $q_2 = 1$ at $x_{\mathrm{NaCl}} = 0$, while q_{12} and q_{23} are eliminated by the values of q_1 (eq. (A.15)) and q_2 (eq. (A.16)) at $x_{\mathrm{NaCl}} = 1$.

$$q_{1,X_{\mathrm{NaCl}}=1} = 47.9048 - 9.36994 \times 10^{-3}\, P + 6.51059 \times 10^{-6}\, P^2 \tag{A.15}$$

$$q_{2,X_{\mathrm{NaCl}}=1} = 0.241022 + 3.45087 \times 10^{-5}\, P - 4.28356 \times 10^{-9}\, P^2 \tag{A.16}$$

The parameters q_{11}, q_{21} and q_{22} are given in Table A.3 as a function of P [bar].

Table A.3: Coefficients for eq. (A.13) and eq. (A.14) with $\boldsymbol{P}$ in bar.

q_{11}	$-32.1724 + 0.0621255\, P$
q_{21}	$-1.69513 - 4.52781 \times 10^{-4}\, P - 6.04279 \times 10^{-8}\, P^2$
q_{22}	$0.0612567 + 1.88082 \times 10^{-5}\, P$

The specific enthalpy for the two-phase region is calculated as the mass-weighted average of the two phases by eq. (A.17).

$$h_m = h_g\chi + h_l(1 - \chi) \tag{A.17}$$

Where h_m is the specific enthalpy in the two-phase region [J kg^{-1}], h_g is the specific enthalpy of the gas phase [J kg^{-1}], χ is the quality [kg kg^{-1}] and h_l is the specific enthalpy of the liquid phase [J kg^{-1}].

A.2.5. Specific Entropy

Publications of specific entropy values for NaCl(aq) are scarce. Dittman (1977) developed a correlation for entropy change and tabulated specific entropy values as a function temperature up to 316 ˚C and mass fraction up to 25 wt% for liquid saturated conditions. The difference between Dittman (1977) relationship and the experimental data was less than 6%. The correlation has been found valid only for small temperature changes and it is independent of pressure. Pitzer et al. (1984) tabulated entropy values for pressures in the range of P_{sat} – 1000 bar, temperatures in the range of 0 – 300 ˚C and molalities in the range of 0 – 6 mol kg^{-1}, as a function of $T - P - m$. The basis of these specific entropy values were experimental measurements of the osmotic and activity coefficient, the enthalpy and heat capacity. The tables from Pitzer et al. (1984) are more comprehensive than the tables from Dittman (1977). Therefore, Dittman (1977) is preferred

A.2.6. Isobaric Heat Capacity

Philips (1981) tabulated heat capacity values as a function of temperatures in the range of 0 – 300 ˚C and molalities in the range of 0 – 4.28 mol kg^{-1}. The maximum error in comparison to the fitted experimental data is only 0.003 kJ kg^{-1} K^{-1} at 300 ˚C and 4.28 mol kg^{-1}, showing an excellent fit. Heat capacity increases with temperature and decreases with molality. Driesner (2007) developed a set of correlations for the heat capacity as a function of temperatures in the range of 0 – 1000 ˚C, pressures in the range of 1 – 5000 bar and mole fractions in the range of 0 – 1 mol mol^{-1}. Good agreement to experimental data from Gates et al. (1987) has been obtained. The experimental data falls within the range for molalities up to 3 mol kg^{-1} and pressures up to 180 bar. However, at high temperatures (> 300 ˚C) and relatively high molalities (> 3 mol kg^{-1}) disagreement up to 10% is encountered. Eq. (A.18) gives the correlation from Driesner (2007).

$$c_p(T, P, x_{\mathrm{NaCl}}) = q_2 c_{p,\mathrm{H_2O}}(T_h^*, P) \tag{A.18}$$

Where c_p is the heat capacity [J kg^{-1} K^{-1}], T is the temperature [˚C], P is the pressure [bar], x_{NaCl} is the mole fraction [mol mol^{-1}] and q_2 is a coefficient given by eq. (A.14). The scaled temperature T_h^* [˚C] is given by eq. (A.12). Furthermore, the identical calculation scheme as for the specific enthalpy is used. The heat capacity in the two-phase region is calculated as the mass-weighted average of the two phases by eq. (A.19).

$$c_{p,m} = c_{p,g}\chi + c_{p,l}(1 - \chi) \tag{A.19}$$

Where $c_{p,m}$ is the heat capacity in the two-phase region [J kg^{-1} K^{-1}], $c_{p,g}$ is the heat capacity of the gas phase [J kg^{-1} K^{-1}], χ is the quality [kg kg^{-1}] and $c_{p,l}$ is the heat capacity of the liquid phase [J kg^{-1} K^{-1}].

A.2.7. Thermal Conductivity

Data for NaCl(aq) for high temperatures have not been published extensively. Philips (1981) used experimental data from Yusufova et al. (1975) to derive a correlation for thermal conductivity as a function of temperatures in the range of 0 – 330 ˚C and molalities in the range of 0 – 4 mol kg^{-1} given in eq. (A.20).

$$\begin{aligned} {}^{k}/_{k_w} = {} & 1 - w_{NaCl}[2.3434 \times 10^{-1} - T\,(7.924 \times 10^{-4}) + T^2\,(3.924 \times 10^{-6})] \\ & + w_{NaCl}^2\,[1.06 \times 10^{-1} - T\,(2 \times 10^{-4}) - T^2(1.2 \times 10^{-6})] \end{aligned} \tag{A.20}$$

Where k is the thermal conductivity of the NaCl(aq) [W m^{-1} K^{-1}], k_w is the thermal conductivity of water [W m^{-1} K^{-1}] and calculated by eq. (A.21), T is the temperature [˚C] and w_{NaCl} is the mass fraction NaCl given in eq. (A.22).

$$\begin{aligned} k_w = {} & -0.92247 + 2.8395\left(\frac{T + 273.15}{273.15}\right) - 1.8007\left(\frac{T + 273.15}{273.15}\right)^2 \\ & + 0.52577\left(\frac{T + 273.15}{273.15}\right)^3 - 0.07344\left(\frac{T + 273.15}{273.15}\right)^4 \end{aligned} \tag{A.21}$$

$$w_{NaCl} = \frac{58.443\,m}{1000 + 58.443\,m} \tag{A.22}$$

Where m is the molality [mol kg^{-1}].

A.2.8. Solubility

The solubility of NaCl(aq) depends mainly on temperature and pH, where solubility increases with increasing temperature and solubility decreases with increasing pH. Many publications refer to Potter et al. (1977), who measured the solubility of NaCl(aq) for temperatures in the range of 0 – 400 ˚C (Philips, 1981; Chou, 1987; Battistelli et al, 1997). Chou (1987) has shown that published NaCl solubility data below 400 ˚C agreed reasonably well. Philips (1981) tabulated solubility values as a function of temperatures in the range of 0-350 ˚C with a deviation of less than 1% from various experimental data. Battistelli et al. (1997) published solubility data as a function of temperatures in the range of 0 – 382 ˚C and compared it to data from Bischoff and Pitzer (1989). The deviation between the published solubility data is relatively small and the different data are all suitable for implementation in a numerical model.

A.3. Gas Flow in GL Duct – Overall Heat Transfer Coefficient

In this section the heat transfer correlations are presented used to calculate heat transfer between the production well and the gas lift duct (eq. (A.23)).

$$\frac{1}{UA} = \frac{1}{h_{c,Wo} 2\pi r_{Wo} L} + \frac{ln(r_{Wo}/r_{Wi})}{2\pi k_{Wc} L} + \frac{1}{h_{c,Wi} 2\pi r_{Wi} L} \tag{A.23}$$

The convective heat transfer coefficient at the outer well is calculated by eq. (A.24).

$$h_{c,Wo} = \frac{\mathrm{Nu}_{GL}\, k_{GL}}{D_h} \tag{A.24}$$

The Nusselt number for outer well convective heat transfer for fully developed turbulent flow in annular ducts according to Gnielinski (2009), given in VDI Heat Atlas (VDI, 2010), is calculated by eq. (A.25). Where eqs. (A.26) – (A.31) give the annular friction factor, modified Reynolds number, correlation constant for annular ducts, hydraulic diameter, correlation factor for annular ducts with heat transfer from both sides, and diameter ratio, respectively.

$$\mathrm{Nu}_{GL} = \frac{(f_a/8)\mathrm{RePr}}{k_1 + 12.7\sqrt{f_a/8}\,(\mathrm{Pr}^{2/3} - 1)} \left[1 + \left(\frac{D_h}{L_E}\right)^{2/3}\right] F_a \tag{A.25}$$

$$f_a = (1.8\log_{10}(\mathrm{Re}^*) - 1.5)^{-2} \tag{A.26}$$

$$\mathrm{Re}^* = \mathrm{Re}\frac{[1 + a^2]\ln a + [1 - a^2]}{[1 - a]^2 \ln a} \tag{A.27}$$

$$k_1 = 1.07 + \frac{900}{\mathrm{Re}} - \frac{0.63}{(1 + 10\mathrm{Pr})} \tag{A.28}$$

$$D_h = D_{ao} - D_{Wo} \tag{A.29}$$

$$F_a = \frac{0.75a^{-0.17} + (0.9 - 0.15a^{0.6})}{1 + a} \tag{A.30}$$

$$a = D_{Wo}/D_{ao} \tag{A.31}$$

Inside the production well, the mechanism of heat transfer in convective boiling is present. For the convective heat transfer coefficient at the inner side of the well, the method proposed by Chen (1966), given in Chemical Engineering Design (Sinnott and Towler, 2009), is adopted. The convective heat transfer coefficient at the inner side of the well, given in eq. (A.32), is considered to be made up of convective and nucleate boiling terms.

$$h_{c,Wi} = h'_{fc} + h'_{nb} \tag{A.32}$$

The forced-convective heat transfer coefficient can be estimated with the single-phase forced-convective heat transfer coefficient modified by an enhancement factor (two-phase correction factor) as in eq. (A.33). This enhancement factor is obtained from Chen (1966).

$$h'_{fc} = h_{fc} F_c \tag{A.33}$$

The single-phase forced-convective heat transfer coefficient is calculated by Gnielinski (1976), given in VDI Heat Atlas (VDI, 2010) (eq. (A.34)). Where the Nusselt number of the geothermal fluid is given by eq. (A.35), and the friction factor is given by eq. (A.36).

$$h_{fc} = \frac{\mathrm{Nu}_{gf}\, k_{gf}}{D_h} \tag{A.34}$$

$$\mathrm{Nu}_{gf} = \frac{(f_a/8)\mathrm{RePr}}{1 + 12.7\sqrt{f_a/8}\,(\mathrm{Pr}^{2/3} - 1)}\left[1 + \left(\frac{D_h}{L_E}\right)^{2/3}\right] \tag{A.35}$$

$$f_a = (1.8\log_{10}\mathrm{Re} - 1.5)^{-2} \tag{A.36}$$

The enhancement factor is obtained empirically from experimental data by Chen (1966) and it is a function of the Lockhart-Martinelli two-phase flow parameter (eq. (A.37)) with turbulent flow in both phases.

$$\frac{1}{X_{tt}} = \left[\frac{\chi}{1-\chi}\right]^{0.9}\left[\frac{\rho_l}{\rho_g}\right]^{0.5}\left[\frac{\mu_g}{\mu_l}\right]^{0.1} \tag{A.37}$$

The nucleate boiling heat transfer coefficient for convective boiling (eq. (A.38)) is modified by a suppression factor to take into account that nucleate boiling is more difficult in a flowing fluid. The suppression factor has been determined empirically by Chen (1966). It is a function of $\mathrm{Re}_l F_c^{1.25}$. Re_l evaluates the Reynolds number if only the liquid phase would flow in the pipes. It is given by eq. (A.39).

$$h'_{nb} = h_{nb} F_s \tag{A.38}$$

$$\mathrm{Re}_l = \frac{(1-\chi)GD_h}{\mu_l} \tag{A.39}$$

The nucleate pool boiling heat transfer coefficient in eq. (A.40) has been proposed by Forster and Zuber (1955).

$$h_{nb} = 0.00122\left[\frac{k_l^{0.79} c_{p,l}^{0.45} \rho_l^{0.49}}{\sigma^{0.5}\mu_l^{0.29}\lambda^{0.24}\rho_g^{0.24}}\right](T_w - T_s)^{0.24}(P_w - P_s)^{0.75} \tag{A.40}$$

Eq. (A.40) is only valid for boiling single-component fluids or close boiling range mixtures (< 5 ˚C). In mixtures the nucleate pool boiling heat transfer coefficient will generally be lower according to Sinnott and Towler (2009). In the present work, CO_2 and H_2O have a wide boiling range. Additionally, it is observed that $h'_{nb} << h'_{fc}$. Therefore, nucleate boiling is neglected and eq. (A.32) evolves to eq. (A.41).

$$h_{c,Wi} = h'_{fc} \tag{A.41}$$

A.4. Thermodynamics Other Geothermal Power Plants

A.4.1. Double-Flash Steam Power Plant

The double-flash steam power plant is almost similar to the single-flash power plant except for the second flashing process to increase maximum power output. The temperature-entropy diagram for double-flash power plants is shown in Figure A.4. The governing equations for the first flashing and separation process are given in eqs. (2.53) and (2.54). The second flashing and separation process is defined by eqs. (A.42) and (A.43).

$$h_3 = h_6 \tag{A.42}$$

$$\chi_6 = \frac{h_3 - h_7}{h_8 - h_7} \tag{A.43}$$

The power produced by the high-pressure (hp) turbine $\dot{W}_{hp,t}$ is given by eq. (2.55) and the power produced by the low-pressure (lp) turbine is given by eq. (A.44).

$$\dot{W}_{lp,t} = \left((1 - \chi_2)\chi_6 + \chi_2\right) \dot{m}_2 (h_9 - h_{10}) \tag{A.44}$$

Where the enthalpy at state 9 is a mixture of the low-pressure saturated vapor at state 8 and the expanded steam from the hp turbine. It is calculated by eq. (A.45).

$$h_9 = \frac{\chi_2 h_5 + (1 - \chi_2)\chi_6 h_8}{\chi_2 + (1 - \chi_2)\chi_6} \tag{A.45}$$

And the enthalpy at stage 10 is given by eq. (A.46).

$$h_{10} = h_9 - \eta_{lp,t}(h_9 - h_{10s}) \tag{A.46}$$

Where according to the Baumann rule the isentropic efficiency of the lp turbine is given by eq. (A.47).

$$\eta_{lp,t} = \eta_{lp,tw} = \eta_{lp,td}\left[\frac{\chi_9 - \chi_{10}}{2}\right] \tag{A.47}$$

The condensation process is then expressed by eq. (A.48).

$$\dot{m}_{cw} c_{p,w} \Delta T_{cw} = (1 - \chi_2)\chi_6 \, \dot{m}_2 (h_{10} - h_{11}) \tag{A.48}$$

The electrical power is now calculated by eq. (A.49).

$$\dot{W}_e = \eta_g \left(\dot{W}_{hp,t} + \dot{W}_{lp,t}\right) \tag{A.49}$$

The equation for consumed power by the pumps is similar to that of the single-flash plant eq. (2.60), as well as the equations for net power, thermal and utilization efficiency given by eq. (2.61), (2.62) and (2.63), respectively.

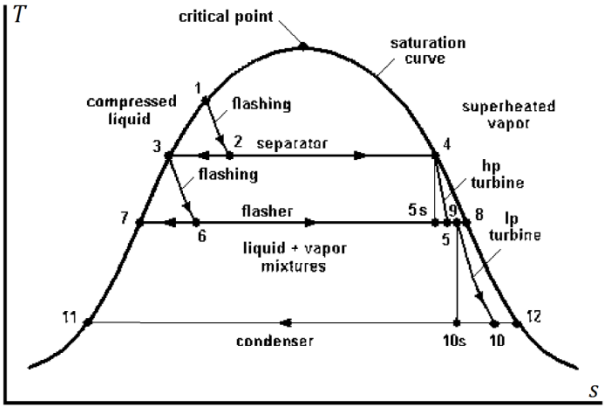


Figure A.4: Temperature-entropy diagram for double-flash plants (DiPippo, 2012).

A.4.2. Dry-Steam Power Plant

The vapor entering the turbine can either be saturated as in Figure A.5 or superheated as in Figure A.6. In case of saturated steam the governing equations, with rearrangement of the numbers, are similar to the single-flash power plant equations for the expansion and condensation process. In case of superheated

steam entering the turbine, the expansion process is imaginary divided in two parts. In the first part the superheated steam is expanded until saturated steam. Whereas in the second part the moisture is involved in the expansion process. The governing equations are again similar to single-flash technology, except for the first part of expansion, in which the Baumann rule is not used to correct the isentropic efficiency.

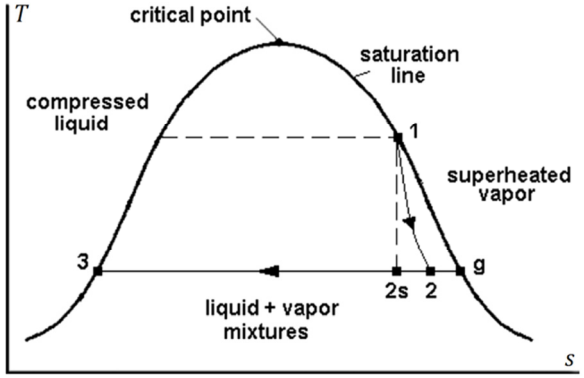


Figure A.5: Temperature-entropy diagram for a dry-steam plant with saturated steam at turbine inlet (DiPippo, 2012).

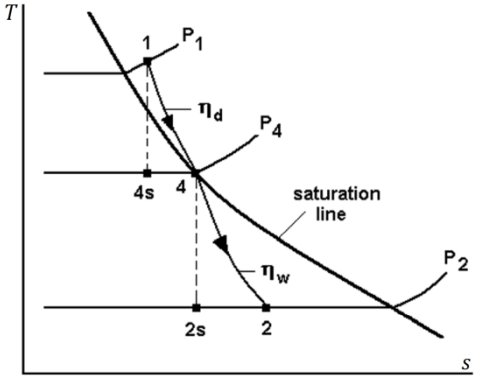


Figure A.6: Dry and wet turbine expansion processes for superheated steam at turbine inlet (DiPippo, 2012).

A.5. Steam Ejector/Condenser

A.5.1. Operation Principle

Figure A.7 presents the schematic of a steam jet ejector. Motive steam enters the ejector at point p. The velocity is subsonic. The velocity increases and the pressure decreases in the converging part of the nozzle. At the throat of the nozzle, point 1, sonic velocity (Mach 1) is reached. The trend of the temperature and pressure in the divergent section of the nozzle continues due to supersonic conditions. At the outlet of the nozzle, point 2, the motive steam pressure is lower than the pressure of the entrained NCG stream, referred to as the suction load. The suction load enters the ejector at point e. In the convergent section of the suction chamber, the velocity increases and the pressure decreases until the suction load mixes with the motive flow. Mixing occurs upward of point 2. In the constant cross sectional throat of the diffuser the mixture experiences a shock wave at point 4. The pressure increases and the velocity reduces to a value below sonic velocity. The shock is induced by the back pressure resistance of the condenser. In the divergent part of the diffuser the pressure of the subsonic mixture is increased by converting the kinetic energy into pressure. The emerging pressure at point c is higher than the suction pressure of the condenser (El-Dessoucky et al., 2002).

A.5.2. Calculation Method

In this section the calculation method is described to calculate the mass flow rate of motive steam required to remove the NCG from the condenser. The numbers in the following equations correspond to Figure 2.24 in Section 2.4.4.3. The mass flow rate of the suction load is given by eq. (A.50).

$$\dot{m}_{11} = \dot{m}_{CO2,11} + \dot{m}_{H2O,11} \tag{A.50}$$

The ratio of the mass flow rate of H_2O and CO_2 is calculated by eq. (A.51). The partial pressures can be calculated by eqs. (2.79) and (2.80).

$$\dot{m}_{H2O,11} = \dot{m}_{CO2,11} \frac{M_{H2O}}{M_{CO2}} \frac{P_{H2O,11}}{P_{CO2,11}} \tag{A.51}$$

The Heat Exchange Institute (HEI) empirically developed a standard to determine the required motive steam graphically. In this study the HEI graphs were adopted and prepared for MATLAB (IPS, 1998; Geremew, 2012). The method is based on determining a dry air equivalent mass flow rate for the NCG mass flow rate. The temperature correction factor (TCF) is determined from Figure A.8. The molecular

weight entrainment ratio (WER) for H_2O and CO_2 is determined from Figure A.9. Then the dry air equivalent (DAE) to steam and DAE to CO_2 is calculated by eq. (A.52) and eq. (A.53), respectively.

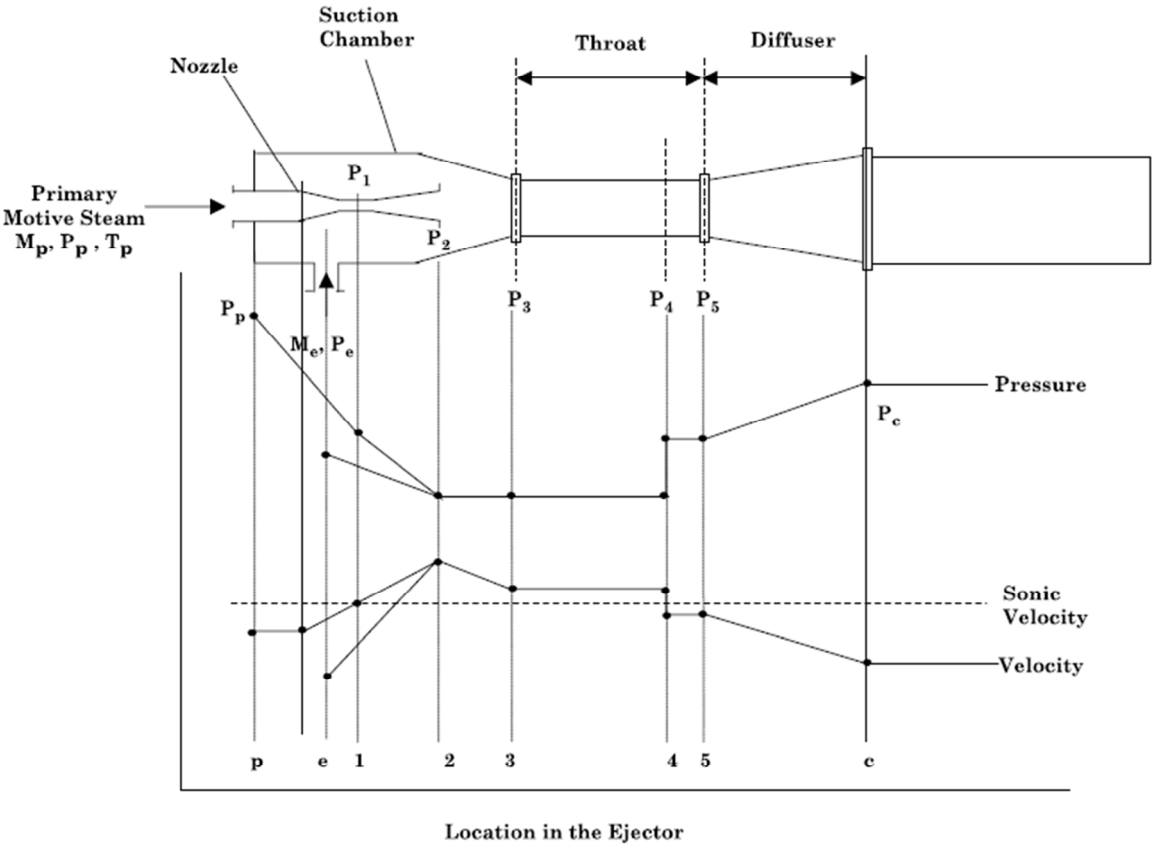


Figure A.7: Schematic of a steam jet ejector with the pressure profile and velocity profile of the flows as a function of the location in the ejector (El-Dessoucky et al., 2002).

$$DAE_{H2O} = \frac{\dot{m}_{H2O}}{TCF_{H2O} WER_{H2O}} \tag{A.52}$$

$$DAE_{CO2} = \frac{\dot{m}_{CO2}}{TCF_{CO2} WER_{CO2}} \tag{A.53}$$

The total DEA is the sum of H_2O to air equivalent and the CO_2 to air equivalent.

$$DAE = DAE_{CO2} + DAE_{H2O} \tag{A.54}$$

In order to calculate the total steam consumption (motive flow), the air to steam ratio (ASR) has to be determined. The ASR is obtained from Figure A.10 as a function of the compression ratio (CR) and the expansion ratio (ER). CR and ER are calculated by eq. (A.55) and eq. (A.56), respectively. The subscripts correspond to Figure A.7.

$$CR = \frac{P_c}{P_e} \tag{A.55}$$

$$ER = \frac{P_p}{P_e} \tag{A.56}$$

Finally, the steam consumption (SC) is calculated by eq. (A.57). The ASR can be obtained from Figure A.10. These graphs have been implemented in MATLAB as data tables.

$$SC = \frac{DAE}{ASR} \tag{A.57}$$

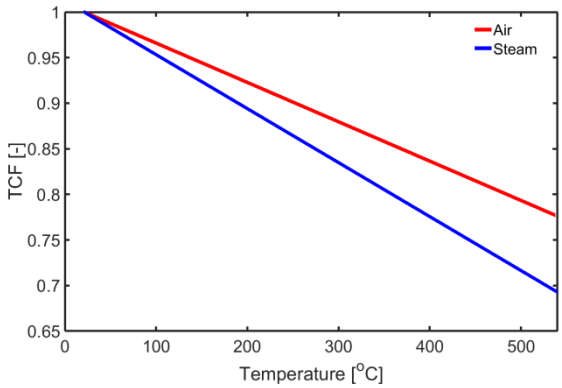


Figure A.8: Temperature correction factor for air and steam.

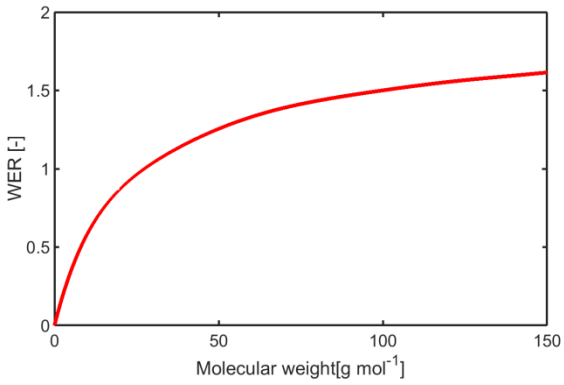


Figure A.9: Dry and wet turbine expansion processes for superheated steam at turbine inlet (DiPippo, 2012).

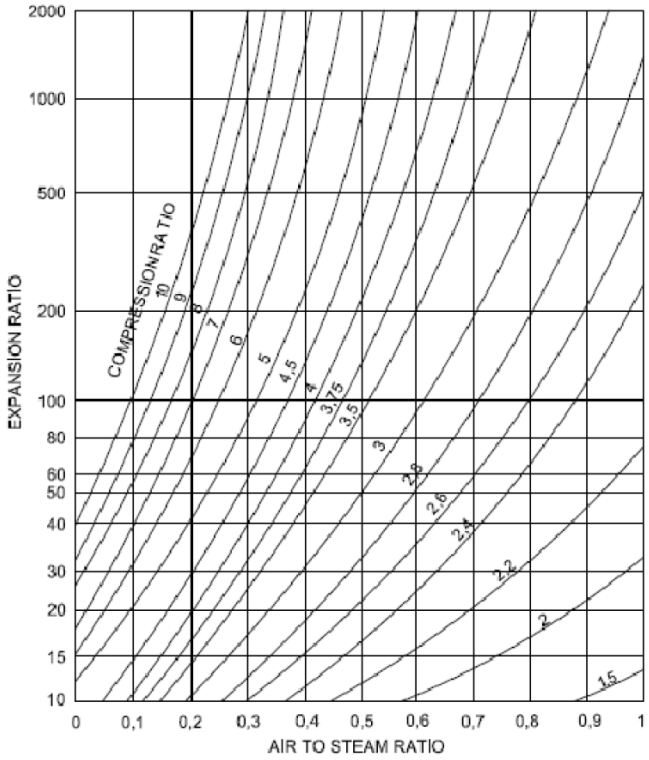


Figure A.10: Air to steam ratio as a function of expansion ratio and compression ratio (Geremew, 2012)

B

MATLAB CODE

B.1. Contents

Main script:

Sub model functions (invoked by sub model):

Remaining sub models and functions (invoked by sub model):

Geothermal fluid property model:

Drift-flux model:

B.2. Code

Continued on next page.

```
%% Start

% Frank Niewold
% Master of Science Thesis
% Commissioned by IF Technology
% Released version 1.0, February 2017
% Documented in "Artificial Lift in Geothermal Wells - A Study to Binary Cycle Geothermal Power Plants
% with Gas Lift in the Production Well"

% This is the main script to perform a simulation of a geothermal power
% plant system containing a reservoir, production well, power plant and
% injecion well.

% Before running the script Start.m
% Add to path selected folders and subfolders in the folder 'Matlab model'
% Define operating conditions and geometries in 'Model Input.xlsx'

% close all
% clear all

tic

%% Declaration of relevant data tables, constants and auxiliary model parameters
[status, input, data, stat] = fSettings();
if (stat == status.SUCCES); disp('Model settings succes'); end;
if (stat == status.FAILURE); return; end;

%% Excel initialization
[stat] = fInitialize_Excel(status);
if (stat == status.SUCCES); disp('Initialize Excel succes'); end;
if (stat == status.FAILURE); return; end;

%% Read and structure the model input
[input, stat] = fModel_Input(input, data, status);
if (stat == status.SUCCES); disp('Model Input succes'); end;
if (stat == status.FAILURE); return; end;

%% Vapor-liquid equilibrium calculation
[output, stat] = fCalc_VLE(input, status);
if (stat == status.SUCCES); disp('VLE calculation succes'); end;
if (stat == status.FAILURE); return; end;

%% Reservoir simulation
[input, output, stat, geofprops] = fCalc_reservoir(input, output, data, status);
if (stat == status.SUCCES); disp('Reservoir calculation succes'); end;
if (stat == status.FAILURE); return; end;

%% Production well single-flash power plant simulation
[input, output, stat, geofprops] = fCalc_prodwell(input, output, data, status);
if (stat == status.SUCCES); disp('Production well calculation succes'); end;
if (stat == status.FAILURE); return; end;

%% Production well with gas lift system simulation
[input, output, stat, geofprops] = fCalc_prodwell_GL(input, output, data, status);
if (stat == status.SUCCES); disp('Production well with gas lift calculation succes'); end;
if (stat == status.FAILURE); return; end;

%% Single-flash power plant simulation
[input, output, stat] = fCalc_SF(input, output, status, data, 1);
if (stat == status.SUCCES); disp('Single-flash power plant calculation succes'); end;
if (stat == status.FAILURE); return; end;

%% Binary cycle power plant simulation
[input, output, stat] = fCalc_BC(input, output, status, data, 1);
if (stat == status.SUCCES); disp('Binary cycle power plant calculation succes'); end;
if (stat == status.FAILURE); return; end;

%% Injection well simulation single-flash power plant
[input, output, stat, geofprops] = fCalc_injwell(input, output, data, status);
if (stat == status.SUCCES); disp('Injection well calculation succes'); end;
if (stat == status.FAILURE); return; end;

%% Injection well simulation binary cycle (BC) power plant
[input, output, stat, geofprops] = fCalc_injwell_BC(input, output, data, status);
if (stat == status.SUCCES); disp('Binary cycle injection well calculation succes'); end;
if (stat == status.FAILURE); return; end;

%% Create figures
fCreate_figures(input, output, status);
if (stat == status.SUCCES); disp('Creating figures succes'); end;
if (stat == status.FAILURE); return; end;

fClose_Excel();

toc
```

```
%% fSettings

% Declaration of relevant data tables, constants and auxiliary model parameters
% Frank Niewold
% Released version 1.0, February 2017

function [status, input, data, stat] = fSettings()

    % List of constants
    input.general.g         = 9.81;     % gravitational constant [m/s2]
    input.general.M_CO2     = 44.01;    % molar mass CO2 [g/mol]
    input.general.M_H2O     = 18.01528; % molar mass H2O [g/mol]
    input.general.M_NaCl    = 58.4428;  % molar mass NaCl [g/mol]
    input.general.gamma     = 1.781072; % Euler's constant e^0.577215

    %% List of model parameters
    % Extensive explanation @ end of this function

    %% fSettings
    input.settings.dT_H2O_sat   = 0.1;      % Safety margin T_sat H2O (fSettings)

    %% fCalc_VLE
    input.settings.T_VLE_range  = 100:1:260;% Necessary temperature range production well (fCalc_VLE)
    input.settings.dP_VLE_sat_v = 0.001;    % Safety margin P_sat_v for VLE gas phase pr...(fCalc_VLE)

    %% fCalc_prodwell
    input.settings.nr_it_dp     = 1:10;     % Maximum number of iterations to recalculate production
                                            % well between degassing pressures (fCalc_prodwell)
    input.settings.dP_abs_pw    = 0.01;     % Minimum required absolute pressure [bar] difference
                                            % between two subsequent iterations for convergence
                                            % at degassing pressure (fCalc_prodwell)
    %% fCalc_geofprops3
    input.settings.error_h_gp3  = 1;        % Error between calculated enthalpies [J/kg]
                                            % (fCalc_geofprops3)
    input.settings.n_it_gp3     = 10;       % Maximum number of iterations before iteration switches
                                            % to fixed step iterations (fCalc_geofprops3)
    input.settings.n_dT_gp3     = 3;        % Maximum number of iterations where dT_old < dT_new,
                                            % which means that T diverges (fCalc_geofprops3)
    input.settings.dT_gp3       = 0.1;      % Fixed temperature step [K] to converge to solution
                                            % (fCalc_geofprops3)

    %% fCalc_geofprops2
    input.settings.error_h_gp2  = 10;       % Error between calculated enthalpies [J/kg]
                                            % (fCalc_geofprops2)
    input.settings.n_it_gp2     = 10;       % Maximum number of iterations before iteration switches
                                            % to fixed step iterations (fCalc_geofprops2)
    input.settings.n_dT_gp2     = 1;        % Maximum number of iterations where dT_old < dT_new,
                                            % which means that T diverges (fCalc_geofprops2)
    input.settings.dT_gp2       = 0.1;      % Fixed temperature step [K] to converge to solution
                                            % (fCalc_geofprops2)
    input.settings.dT_VLE_sat_v = 0.09;     % Safety margin on T_sat_v for temperature check region
                                            % 2 (fCalc_geofprops2)

    %% fCalc_SF
    input.settings.dP_step_SF      = 0.5;   % Stepsize pressure [bar] to find maximum power
                                            % single-flash power plant (fCalc_SF)
    input.settings.chi_2_min       = 0.1;   % Minimum quantity for initial flash calculation if
                                            % quality is 0 in production well (fCalc_SF)
    input.settings.error_eta_t_SF  = 0.0001;% Error between calculated new and old turbine efficiency
                                            % taking into account wet turbine efficiency (fCalc_SF)
    input.settings.T0_12           = 150;   % Initial temperature for iteration of temperature @
                                            % state 12 (fCalc_SF)
    input.settings.error_T_9_10    = 1;     % Error between T_10 and T_9 [K] (fCalc_SF)

    %% fCalc_BC
    input.settings.dT_evap  = 0.5;          % Temperature step calculation maximum power output BC

    %% Load data tables with relevant thermophysical properties for interpolation
    load H2O_sat;
    load m_CO2_degas; load m_NaCl_degas; load P_degas;   load T_degas;
    load m_SC.mat;    load T_SC.mat;     load SC.mat;
```

```
    load P_CO2;       load T_CO2;       load h_CO2;     load s_CO2;
    load H2O_sat_props;
    load h_H2O_SH;    load s_H2O_SH;    load T_H2O_SH;  load P_H2O_SH;   load cp_H2O_SH;
    load T_H2O_SC;    load P_H2O_SC;    load cp_H2O_SC; load rho_H2O_SC; load s_H2O_SC
    load C5H12_sat_props;

    % Degassing tables Duan and Sun (2003)
    data.m_CO2_degas    =   m_CO2_degas;    % molality CO2 0 - 1.5 [mol/kg]
    data.m_NaCl_degas   =   m_NaCl_degas;   % molality NaCl 0 - 3 [mol/kg]
    data.T_degas        =   T_degas;        % temperature 423 - 523 [K]
    data.P_degas        =   P_degas;        % pressure [bar]

    % Separation coefficient tables
    data.m_SC           =   m_SC;           % molality NaCl 0 - 3 [mol/kg]
    data.T_SC           =   T_SC;           % temperature SC 100 - 260 [C]
    data.SC             =   SC;             % separation coefficient

    % H2O properties
    data.H2O_sat_props  =   H2O_sat_props;  % 7 columns (P_sat[bar],T_sat[C],h_sat_l[kJ/kg],
                                            % h_sat_v[kJ/kg],s_sat_l[kJ/kg/K],s_sat_v[kJ/kg/K],rho_l)
    data.P_H2O_SC       =   P_H2O_SC;       % pressure subcooled H2O 0.01 - 15 [bar]
    data.T_H2O_SC       =   T_H2O_SC;       % temperature subcooled H2O 6.9 - 198.3 [C]
    data.cp_H2O_SC      =   cp_H2O_SC;      % specific heat capacity subcooled H2O [J/kg/K]
    data.rho_H2O_SC     =   rho_H2O_SC;     % density subcooled H2O [kg/m3]
    data.s_H2O_SC       =   s_H2O_SC;       % entropy subcooled H2O [kg/m3]
    data.P_H2O_SH       =   P_H2O_SH;       % pressure superheated H2O 0.8 - 1.3 [bar]
    data.T_H2O_SH       =   T_H2O_SH;       % temperature superheated H2O 100 - 373 [C]
    data.cp_H2O_SH      =   cp_H2O_SH;      % specific heat capacity superheated H2O [J/kg/K]

    % CO2 properties
    data.P_CO2          =   P_CO2;          % pressure 0.005 - 40 [bar]
    data.T_CO2          =   T_CO2;          % temperature 20 - 400 [C]
    data.h_CO2          =   h_CO2;          % enthalpy [kJ/kg]
    data.s_CO2          =   s_CO2;          % entropy [kJ/kg/K]

    data.C5H12_sat_props = C5H12_sat_props; % 6 columns (T_sat[C],P_sat[bar],h_sat_l[kJ/kg],
                                            % h_sat_v[kJ/kg],s_sat_l[kJ/kg/K],s_sat_v[kJ/kg/K])

    % Adjustment of data tables
    data.H2O_sat(:,2) = H2O_sat(:,2) + input.settings.dT_H2O_sat;
    % Input saturated vapour temperature/pressure water
    data.H2O_sat = H2O_sat;

    %% Auxiliary parameters
    status.SUCCES  = 1;
    status.FAILURE = 0;
    status.YES     = 1;
    status.NO      = 0;

    % Succesfull simulation
    stat = status.SUCCES;

end

% List of model parameters
% Extensive explanation with default values

% input.settings.dT_H2O_sat = 0.1;
% Safety margin T_sat H2O. The saturated temperature is interpolated from the data table H2O_sat.
% The safety margin prevents that the interpolated saturated temperature is below the saturated
% temperature from the Francke Model.

% input.settings.T_VLE_range = 100:1:250;
% Necessary temperature range production well. This is the temperature range for which the VLE
% properties are obtained.

% input.settings.dP_VLE_sat_v = 0.001;
% Safety margin P_sat_v for VLE gas phase properties. The P_sat_v is decreased with DP_VLE_sat_v
% to make sure that gas phase properties are obtained from Francke Model.

%input.settings.DF_model = 2;
% Drift-flux model choice. 1 == Hasan & Kabir (2010), 2 == Rouhani & Axelsson (1970),
% 3 == Dix (1971), 4 == Nicklin (1961), 5 == Toshiba (1989).

% input.settings.nr_it_dp = 1:5;
% Number of iterations to recalculate production well between degassing pressures of Duan(2003)
% Francke (2014).

% input.settings.error_h_gp3 = 1;
```



```
% Error between calculated enthalpies [J/kg] (fCalc_geofprops3). The calculated enthalpies are
% from the energy balance and from the Francke Model iteration.

% input.settings.n_it_gp3 = 10;
% Maximum number of iterations before iteration switches to fixed step iterations. At first
% iteration is performed by a manual programmed code with variable stepsizes. If iteration does
% not succeed after user-defined number of iterations, fixed step iteration is performed based
% on the trend of the temperature and enthalpy.

% input.settings.n_dT_gp3 = 2;
% Maximum number of iterations where dT_old < dT_new, which means that T diverges. The iteration
% procedure is based on a mutable T in order to solve the equations for P and h. If the dT between
% T_new and T_old increases in a subsequent iteration, the calculation diverges and no solution
% is found.

% input.settings.dT_gp3 = 0.5;
% Fixed temperature step [K] to converge to solution (fCalc_geofprops3). Lower dT decreases the
% calculation error, but increases the computational time.

% input.settings.dP_abs_pw    = 0.1;
% Absolute pressure difference [bar] between two subsequent iterations taken into account P_degas
% from Duan and Sun (2003). This is applied on the first segment before degassing start according
% to Francke (2014).

% input.settings.error_h_gp2 = 10;
% Error between calculated enthalpies [J/kg] (fCalc_geofprops2). The calculated enthalpies are
% from the energy balance and from the Francke Model iteration.

% input.settings.n_it_gp2 = 10;
% Maximum number of iterations before iteration switches to fixed step iterations. At first
% iteration is performed by a manual programmed code with variable stepsizes. If iteration does
% not succeed after user-defined number of iterations, fixed step iteration is performed based
% on the trend of the temperature and enthalpy.

% input.settings.n_dT_gp2 = 3;
% Maximum number of iterations where dT_old < dT_new, which means that T diverges. The iteration
% procedure is based on a mutable T in order to solve the equations for P and h. If the dT between
% T_new and T_old increases in a subsequent iteration, the calculation diverges and no solution
% is found.

% input.settings.dT_gp2 = 0.1;
% Fixed temperature step [K] to converge to solution (fCalc_geofprops2). Lower dT decreases the
% calculation error, but increases the computational time.

% input.settings.dT_VLE_sat_v = 0.05;
% Safety margin on T_sat_v for temperature check region 2 (fCalc_geofprops2). (P < 40) &&
% (T < T_sat_v - dT_VLE_sat_v) applies to region 2.

% input.settings.dP_step_SF = 0.5;
% Stepsize pressure [bar] to find maximum power single-flash power plant (fCalc_SF). If the
% geothermal fluid have not flashed yet or enough in the production well, the fluid is flashed
% in the cyclone separator. dP_step_SF is the fixed stepsize to find maximum power.

% input.settings.chi_2_min = 0.1;
% Minimum quantity for initial flash calculation if quality is 0 in production well (fCalc_SF).

% input.settings.error_eta_t_SF  = 0.0001;
% Error between calculated new and old turbine efficiency taking into account wet turbine
% efficiency. Wet turbine efficiency is a function of the quantity, therefore it needs an extra
% iteration (fCalc_SF)

% input.settings.T0_12 = 150;
% Initial temperature for iteration of temperature @ state 12 (fCalc_SF). T0_12 is the iteration
% variable in order to find the solution for isentropic compression s_mix_11 = s_mix_12s. This is
% carefully chosen to converge to a solution.

% input.settings.error_T_9_10 = 1;
% Error between T_10 from injection well calculation and T_9 from single-flash power plant calculation
% in [K].
```

```
%% f_Initialize_Excel

% Excel initialization
% Frank Niewold
% Released version 1.0, February 2017
```

```
function [stat] = fInitialize_Excel(status)

    % Check if Excel workbook(s) are open
    try
        % Check if an Excel server is running
        Excel = actxGetRunningServer('Excel.Application');
    catch
        % No instance of Excel is currently running.
    end

    % If Excel workbook(s) are open, open dialogue box
    if exist('Excel','var')
        formatSpec = 'All active Excel files will be saved and closed.\n\nIs this OK?';
        str        = sprintf(formatSpec);
        choice     = questdlg(str,'Warning','YES','NO','NO');
        switch choice
            % case 'YES'
        case 'NO'
            msgbox('Save or close Excel files before running Start.m');
            stat = status.FAILURE;
            return
        end

        wbs = Excel.Workbooks; % Get the names of all open Excel files

        % List the entire path of all Excel workbooks that are currently open
        for i = 1:wbs.Count
            wbs.Item(i).FullName;
        end
        for i = 1:wbs.Count
            [~,name,~] = fileparts(wbs.Item(i).FullName); % [pathstr,name,ext]
            wbs.Item(name).Save;
        end
        Excel.Quit
    end

    % Create an Excel server and open brine_prop.xlsm
    Excel = actxserver ('Excel.Application');
    set (Excel, 'Visible', 1);
    Excel.Workbooks.Open([pwd '\brine_prop.xlsm']);

    % Set right Excel sheet
    Excel  = actxGetRunningServer('Excel.Application');

    % Activate sheet
    Sheets = Excel.ActiveWorkBook.Sheets;
    sheet2 = get(Sheets, 'Item', 1);
    invoke(sheet2, 'Activate');
    sheet  = Excel.Activesheet;

    % KCl, CaCl, N2, CH4 mass fraction are 0
    sheet.set('Range', 'C9', 0);
    sheet.set('Range', 'C10', 0);
    sheet.set('Range', 'C12', 0);
    sheet.set('Range', 'C13', 0);

    % Succesfull simulation
    stat = status.SUCCES;

end
```

```
%% fModel_Input

% Read and structure the model input
% Frank Niewold
% Released version 1.0, February

function [input, stat] = fModel_Input(input, data, status)

    % Succesfull simulation
    stat = status.SUCCES;

    % Read user-defined model input from Excel file 'Model Input'
    [num,txt,~]    = xlsread('Model Input','Input general');
    input_general = num;

    %% Input parameters reservoir properties
    input.general.m_gf     = input_general(1,1);    % mass flow geothermal fluid [kg/s]
    input.general.P_res    = input_general(2,1);    % pressure reservoir [bar]
    input.general.T_res    = input_general(3,1);    % temperature reservoir [C]
    input.general.PI       = input_general(4,1);    % productivity index
    input.general.II       = input_general(5,1);    % injectivity index
    input.general.w_NaCl   = input_general(6,1);    % mass fraction NaCl
    input.general.w_CO2    = input_general(7,1);    % mass fraction CO2

    % Check if reservoir is in liquid state single-flash power plant
    [geofprops] = fCalc_geofprops1(input.general.P_res, input.general.T_res, input.general.w_NaCl, ...
                  input.general.w_CO2);
    if geofprops(1,1) > (input.general.P_res)
        disp('ERROR: Reservoir pressure below degassing pressure. ACTION: Increase reservoir pressure');
        msgbox('Reservoir pressure below degassing pressure. ACTION: Increase reservoir pressure',
'Error','error');
        stat = status.FAILURE;
    end

    %% Environmental properties
    input.general.P_atm    = input_general(1,9);    % atmospheric pressure [bar]
    input.general.T_surf_r = input_general(2,9);    % temperature earth's surface rock [C]
    input.general.T_surf_w = input_general(3,9);    % temperature surface water [C]

    %% Input parameters production well
    years           = input_general(1,5); if isnan(years)   == 1; years   = 0; end;
    days            = input_general(2,5); if isnan(days)    == 1; days    = 0; end;
    hours           = input_general(3,5); if isnan(hours)   == 1; hours   = 0; end;
    seconds         = input_general(4,5); if isnan(seconds) == 1; seconds = 0; end;
    input.general.t = (years * 365 * 24 * 3600) + (days * 24 * 3600) + (hours * 3600) + seconds; % [s]

    % Input drift-flux model
    s1 = {'Homogeneous','Rouhani & Axelsson','Hasan & Kabir','Dix','Nicklin','Toshiba'};
    s2 = txt(15,6);
    tf = strcmp(s1,s2);
    input.prodwell.DF_model    = find(tf);
    input.prodwell_GL.DF_model = find(tf);

    %% Input parameters single-flash power plant
    input.SF.P_out_t    = input_general(15,1);    % pressure outlet steam turbine
    input.SF.eta_t      = input_general(16,1);    % turbine efficiency
    input.SF.eta_td     = input_general(17,1);    % dry steam turbine efficiency
    input.SF.eta_p      = input_general(18,1);    % pump efficiency
    input.SF.eta_g      = input_general(19,1);    % generator efficiency
    input.SF.eta_SEC    = input_general(20,1);    % efficiency centrifugal compressor
    input.SF.T_out_cd   = input_general(21,1);    % temperature outlet condenser
    input.SF.T_pinch_cd = input_general(22,1);    % pinchpoint temperature condenser
    input.SF.dP_cwp     = input_general(23,1);    % pressure build-up cooling water pump

%     % MATLAB simulation
%     s1 = {'YES'};
%     s2 = txt(34,2);
%     sim.SF = strcmp(s1,s2);                  % 1 = YES, 0 = NO

    %% Input parameters binary cycle power plant
    input.B.T_inj      = input_general(15,5);    % Reinjection temperature geothermal fluid
    input.B.T_pinch_ev = input_general(16,5);    % Pinchpoint temperature preheater/evaporator
    input.B.eta_td     = input_general(17,5);    % dry steam turbine efficiency
    input.B.eta_p      = input_general(18,5);    % pump efficiency
    input.B.eta_g      = input_general(19,5);    % generator efficiency
    input.B.eta_com    = input_general(20,5);    % gas lift compressor efficiency
    input.B.T_out_cd   = input_general(21,5);    % temperature outlet condenser
    input.B.T_pinch_cd = input_general(22,5);    % Pinchpoint temperature condenser
    input.B.dP_cwp     = input_general(23,5);    % pressure build-up cooling water pump

%     % MATLAB simulation
%     s1 = {'YES'};
%     s2 = txt(34,6);
%     sim.B = strcmp(s1,s2);         % 1 = YES, 0 = NO

    %% Input production well dimensions single-flash power plant
    dim_prodwell                = xlsread('Model Input','dim_prodwell');
    [row,~]                     = size(dim_prodwell);
    dim_prodwell                = dim_prodwell(2:row,:);
    dim_prodwell(isnan(dim_prodwell)) = 0;
    input.prodwell.segment(:,1) = 1:sum(dim_prodwell(:,11)); % create array of segment numbers
    input.prodwell.dl           = [];                        % create array of segment lengths
    input.prodwell.D_i          = [];                        % create array of segment diameters
    input.prodwell.dz           = [];                        % create array of segment heights
    input.prodwell.tvd          = [];    % create array of segment true vertical depths
    tvd_pre                     = 0;     % initial true vertical depth
```

```
input.prodwell.eps_pipe     = [];    % create array of segment absolute pipe roughness
input.prodwell.grad_T_g     = [];    % create array of segment geothermal temperature gradients
input.prodwell.k_r          = [];    % create array of segment rock thermal conductivities
input.prodwell.alfa_r       = [];    % create array of segment rock thermal diffusivities

% fill arrays with model input
for i = 1:size(dim_prodwell,1)
    dl       = dim_prodwell(i,3)/dim_prodwell(i,11);
    D_i      = dim_prodwell(i,9);
    dz       = dim_prodwell(i,5)/dim_prodwell(i,11);
    dz_tvd   = dim_prodwell(i,5)/dim_prodwell(i,11);
    eps_pipe = dim_prodwell(i,12);
    grad_T_g = dim_prodwell(i,13);
    k_r      = dim_prodwell(i,14);
    alfa_r   = dim_prodwell(i,15);
    for j = 1:dim_prodwell(i,11)
        input.prodwell.dl(end+1,1)        = dl;
        input.prodwell.D_i(end+1,1)       = D_i;
        input.prodwell.dz(end+1,1)        = dz;
        input.prodwell.tvd(end+1,1)       = dz_tvd + tvd_pre;
        tvd_pre                           = input.prodwell.tvd(end,1);
        input.prodwell.eps_pipe(end+1,1)  = eps_pipe;
        input.prodwell.grad_T_g(end+1,1)  = grad_T_g;
        input.prodwell.k_r(end+1,1)       = k_r;
        input.prodwell.alfa_r(end+1,1)    = alfa_r;
    end
end

% Flip arrays. Bottom production well = top array
input.prodwell.dl       = flipud(input.prodwell.dl);
input.prodwell.dz       = flipud(input.prodwell.dz);
input.prodwell.tvd      = flipud(input.prodwell.tvd);
input.prodwell.D_i      = flipud(input.prodwell.D_i);
input.prodwell.eps_pipe = flipud(input.prodwell.eps_pipe);
input.prodwell.grad_T_g = flipud(input.prodwell.grad_T_g);
input.prodwell.k_r      = flipud(input.prodwell.k_r);
input.prodwell.alfa_r   = flipud(input.prodwell.alfa_r);

%% Input injection well dimensions
dim_injwell                = xlsread('Model Input','dim_injwell');
[row,~]                    = size(dim_injwell);
dim_injwell                = dim_injwell(2:row,:);
input.injwell.segment(:,1) = 1:sum(dim_injwell(:,11));
input.injwell.dl           = []; % create array of segment lengths
input.injwell.D_i          = []; % create array of segment diameters
input.injwell.dz           = []; % create array of segment heights
input.injwell.tvd          = []; % create array of segment true vertical depths
tvd_pre                    = 0;
input.injwell.eps_pipe     = []; % create array of segment absolute pipe roughness
input.injwell.grad_T_g     = []; % create array of segment geothermal temperature gradients
input.injwell.k_r          = []; % create array of segment rock thermal conductivities
input.injwell.alfa_r       = []; % create array of segment rock thermal diffusivities

% fill arrays with model input
for i = 1:size(dim_injwell,1)
    dl       = dim_injwell(i,3)/dim_injwell(i,11);
    D_i      = dim_injwell(i,9);
    dz       = dim_injwell(i,5)/dim_injwell(i,11);
    dz_tvd   = dim_injwell(i,5)/dim_injwell(i,11);
    eps_pipe = dim_injwell(i,12);
    grad_T_g = dim_injwell(i,13);
    k_r      = dim_injwell(i,14);
    alfa_r   = dim_injwell(i,15);
    for j = 1:dim_injwell(i,11)
        input.injwell.dl(end+1,1)        = dl;
        input.injwell.D_i(end+1,1)       = D_i;
        input.injwell.dz(end+1,1)        = dz;
        input.injwell.tvd(end+1,1)       = dz_tvd + tvd_pre;
        tvd_pre                          = input.injwell.tvd(end,1);
        input.injwell.eps_pipe(end+1,1)  = eps_pipe;
        input.injwell.grad_T_g(end+1,1)  = grad_T_g;
        input.injwell.k_r(end+1,1)       = k_r;
        input.injwell.alfa_r(end+1,1)    = alfa_r;
    end
end

% Flip arrays. Bottom injection well = top array
input.injwell.dl       = flipud(input.injwell.dl);
input.injwell.dz       = flipud(input.injwell.dz);
input.injwell.tvd      = flipud(input.injwell.tvd);
input.injwell.D_i      = flipud(input.injwell.D_i);
input.injwell.eps_pipe = flipud(input.injwell.eps_pipe);
input.injwell.grad_T_g = flipud(input.injwell.grad_T_g);
input.injwell.k_r      = flipud(input.injwell.k_r);
input.injwell.alfa_r   = flipud(input.injwell.alfa_r);

%% Input production well dimensions with gas lift system
dim_prodwell                = xlsread('Model Input','dim_prodwell');
[row,~]                     = size(dim_prodwell);
dim_prodwell                = dim_prodwell(2:row,:);
dim_prodwell(isnan(dim_prodwell)) = 0;
input.prodwell_GL.segment(:,1) = 1:sum(dim_prodwell(:,11)); % create array of segment numbers
input.prodwell_GL.dl           = [];                        % create array of segment lengths
input.prodwell_GL.D_i          = [];                        % create array of segment diameters
input.prodwell_GL.dz           = [];                        % create array of segment heights
input.prodwell_GL.tvd          = [];    % create array of segment true vertical depths
tvd_pre                        = 0;     % initial true vertical depth
input.prodwell_GL.eps_pipe     = [];    % create array of segment absolute pipe roughness
input.prodwell_GL.grad_T_g     = [];    % create array of segment geothermal temperature gradients
input.prodwell_GL.k_r          = [];    % create array of segment rock thermal conductivities
input.prodwell_GL.alfa_r       = [];    % create array of segment rock thermal diffusivities

input.prodwell_GL.z_GL                 = dim_prodwell(5,19); % depth gas lift valve
input.prodwell_GL.m_GL                 = dim_prodwell(6,19); % initial mass flow rate gas lift
[row_sn,~]                             = find(dim_prodwell(:,2) == input.prodwell_GL.z_GL);
input.prodwell_GL.segnr_GL             = sum(dim_prodwell(row_sn+1:row-1,11)) + 1;
                                                    % segment number of gas lift valve

% fill arrays with model input
for i = 1:size(dim_prodwell,1)
    dl       = dim_prodwell(i,3)/dim_prodwell(i,11);
    D_i      = dim_prodwell(i,9);
    dz       = dim_prodwell(i,5)/dim_prodwell(i,11);
    dz_tvd   = dim_prodwell(i,5)/dim_prodwell(i,11);
    eps_pipe = dim_prodwell(i,12);
    grad_T_g = dim_prodwell(i,13);
    k_r      = dim_prodwell(i,14);
    alfa_r   = dim_prodwell(i,15);
    for j = 1:dim_prodwell(i,11)
        input.prodwell_GL.dl(end+1,1)        = dl;
        input.prodwell_GL.D_i(end+1,1)       = D_i;
        input.prodwell_GL.dz(end+1,1)        = dz;
        input.prodwell_GL.tvd(end+1,1)       = dz_tvd + tvd_pre;
        tvd_pre                              = input.prodwell_GL.tvd(end,1);
        input.prodwell_GL.eps_pipe(end+1,1)  = eps_pipe;
        input.prodwell_GL.grad_T_g(end+1,1)  = grad_T_g;
        input.prodwell_GL.k_r(end+1,1)       = k_r;
        input.prodwell_GL.alfa_r(end+1,1)    = alfa_r;
    end
end

% Flip arrays. Bottom production well = top array
input.prodwell_GL.dl       = flipud(input.prodwell_GL.dl);
input.prodwell_GL.dz       = flipud(input.prodwell_GL.dz);
input.prodwell_GL.tvd      = flipud(input.prodwell_GL.tvd);
input.prodwell_GL.D_i      = flipud(input.prodwell_GL.D_i);
input.prodwell_GL.eps_pipe = flipud(input.prodwell_GL.eps_pipe);
input.prodwell_GL.grad_T_g = flipud(input.prodwell_GL.grad_T_g);
input.prodwell_GL.k_r      = flipud(input.prodwell_GL.k_r);
input.prodwell_GL.alfa_r   = flipud(input.prodwell_GL.alfa_r);

%% Obtain molalities from GFP Excel model
Excel  = actxGetRunningServer('Excel.Application');
Sheets = Excel.ActiveWorkBook.Sheets;
sheet2 = get(Sheets, 'Item', 1);
invoke(sheet2, 'Activate');
sheet  = Excel.Activesheet;
sheet.set('Range', 'C8', input.general.w_NaCl);
sheet.set('Range', 'C11', input.general.w_CO2);
range  = sheet.get('Range', 'D8:D13');
range.Value;
data   = range.Value;
input.general.m_NaCl = cell2mat(data(1,1));
input.general.m_CO2  = cell2mat(data(4,1));

end
```

```
%% fCalc_VLE

% Calculation of vapor-liquid equilibrium
% Frank Niewold
% Released version 1.0, February 2017

function [output, stat] = fCalc_VLE(input, status)

    %input.general.w_NaCl = 0;
    %input.general.w_CO2  = 0;
    T = input.settings.T_VLE_range;

    Excel = actxGetRunningServer('Excel.Application');

    h = waitbar(0,'VLE calculation. Please wait...');

    % Obtain saturated vapor properties for user-defined temperatures
    for i = 1:length(T)
        P = 0; % for obtaining P_sat_v from Francke Model
        waitbar(i/length(T))
        Sheets = Excel.ActiveWorkBook.Sheets;
        sheet2 = get(Sheets, 'Item', 1); % two-phase
        invoke(sheet2, 'Activate');
        sheet  = Excel.Activesheet;

        sheet.set('Range', 'C3', P);
        sheet.set('Range', 'C4', T(i));
        sheet.set('Range', 'C8', 0);
        sheet.set('Range', 'C9', 0);
        sheet.set('Range', 'C10', 0);
        sheet.set('Range', 'C11', 0);
        %sheet.set('Range', 'C11', input.general.w_CO2);
        sheet.set('Range', 'C12', 0);
        sheet.set('Range', 'C13', 0);
        range = sheet.get('Range', 'I4:I19');
        range.Value;
        data  = range.Value;
        B     = regexp(strrep(char(data(16,1)), ',', '.'), '\d+', 'match'); % replace , with .
        output.VLE.P_sat_v(i,1) = str2num([char(B(1,4)) '.' char(B(1,5))]);
        output.VLE.T_sat_v(i,1) = T(i);

        % Obtain density, heat capacity and enthalpy for saturated vapor conditions
        Sheets = Excel.ActiveWorkBook.Sheets;
        sheet2 = get(Sheets, 'Item', 2); % gas phase
        invoke(sheet2, 'Activate');
        sheet  = Excel.Activesheet;
        sheet.set('Range', 'C2', output.VLE.P_sat_v(i,1) - input.settings.dP_VLE_sat_v);
        sheet.set('Range', 'C3', T(i));
        sheet.set('Range', 'C5', input.general.w_CO2);
        sheet.set('Range', 'C6', 0);
        sheet.set('Range', 'C7', 0);
        range  = sheet.get('Range', 'C8');
        range.Value;
        w_H2O  = range.Value;
        if w_H2O == 1
            range   = sheet.get('Range', 'H3:H5');
            range.Value;
            datagas = cell2mat(range.Value(1:3,1));
            output.VLE.rho_sat_v(i,1) = datagas(1,1);
            output.VLE.cp_sat_v(i,1)  = datagas(2,1);
            output.VLE.h_sat_v(i,1)   = datagas(3,1);
        else
            range  = sheet.get('Range', 'G3:G5');
            range.Value;
            datagas = cell2mat(range.Value(1:3,1));
            output.VLE.rho_sat_v(i,1) = datagas(1,1);
            output.VLE.cp_sat_v(i,1)  = datagas(2,1);
            output.VLE.h_sat_v(i,1)   = datagas(3,1);
        end

        % Obtain density, heat capacity and enthalpy for saturated liquid conditions
        Sheets = Excel.ActiveWorkBook.Sheets;
        sheet2 = get(Sheets, 'Item', 3); % liquid phase
        invoke(sheet2, 'Activate');
        sheet  = Excel.Activesheet;
        sheet.set('Range', 'C3', output.VLE.P_sat_v(i,1) + input.settings.dP_VLE_sat_v);
        sheet.set('Range', 'C4', T(i));
        sheet.set('Range', 'C6', input.general.w_NaCl);
        sheet.set('Range', 'C7', 0);
        sheet.set('Range', 'C8', 0);

        range   = sheet.get('Range', 'G4:G6');
        range.Value;
        dataliq = cell2mat(range.Value(1:3,1));
        output.VLE.rho_sat_l(i,1) = dataliq(1,1);
        output.VLE.cp_sat_l(i,1)  = dataliq(2,1);
        output.VLE.h_sat_l(i,1)   = dataliq(3,1);

    end
    close(h)
    stat = status.SUCCES;

end
```

```
%% fCalc_reservoir

% Simulation of reservoir
% Frank Niewold
% Released version 1.0, February

function [input, output, stat, geofprops] = fCalc_reservoir(input, output, data, status)

    % Succesfull simulation
    stat = status.SUCCES;

    output.reservoir.geofprops = zeros(3,31); % Create matrix with zeros
    % Row 1 = inlet reservoir, row 2 = far field reservoir, row 3 is outlet reservoir

    %% Outlet injection well and inlet reservoir
    output.reservoir.geofprops(1,1) = input.general.P_res + (input.general.m_gf / input.general.II);
                                                                                % pressure [bar]
    %% far-field reservoir
    output.reservoir.geofprops(2,1)   = input.general.P_res;   % Pressure [bar] in reservoir
    output.reservoir.geofprops(2,2)   = input.general.T_res;   % Temperature [C] in reservoir
    output.reservoir.geofprops(2,3:5) = [input.general.w_NaCl input.general.w_CO2 (1 - ...
                                         input.general.w_NaCl - input.general.w_CO2)];
                                                               % Overall composition mass fractions
    [geofprops] = fCalc_geofprops1 (output.reservoir.geofprops(2,1), input.general.T_res, ...
                  input.general.w_NaCl, input.general.w_CO2, output);
                                                               % Calculate geothermal fluid properties
    output.reservoir.geofprops(2,6:31) = geofprops(1,1:26);

    %% Inlet production well and outlet reservoir
    output.reservoir.geofprops(3,1)   = output.reservoir.geofprops(2,1) - (input.general.m_gf/...
                                        input.general.PI);                  % pressure [bar]
    output.reservoir.geofprops(3,2)   = input.general.T_res;                % temperature [C]
    output.reservoir.geofprops(3,3:5) = [input.general.w_NaCl input.general.w_CO2 (1 - ...
                                         input.general.w_NaCl - input.general.w_CO2)];% composition
    [geofprops] = fCalc_geofprops1 (output.reservoir.geofprops(3,1), input.general.T_res, ...
                  input.general.w_NaCl, input.general.w_CO2, output);    % geothermal fluid properties
    output.reservoir.geofprops(3,6:31) = geofprops(1,1:26);              % geothermal fluid properties

    % Check if geothermal fluid is liquid with degassing pressure Duan and Sun (2003)
    output.reservoir.P_degas_in = interp3(data.m_NaCl_degas, data.T_degas, data.m_CO2_degas, ...
                                  data.P_degas, input.general.m_NaCl, ...
                                  output.reservoir.geofprops(3,2) + 273.15, input.general.m_CO2);

    if output.reservoir.geofprops(3,1) < output.reservoir.P_degas_in
        disp('ERROR: Pressure at inlet production well below degassing pressure. ACTION: Decrease mass flow')
        msgbox('Pressure at inlet production well below degassing pressure. ACTION: Decrease mass flow',
'Error','error');
        stat = status.FAILURE;
    end
end
```

```
%% fCalc_prodwell

% Simulation of a production well - self flowing for a single-flash power plant
% Frank Niewold
% Released version 1.0, February 2017

function [input, output, stat, geofprops] = fCalc_prodwell(input, output, data, status)

    % Succesfull simulation
    stat = status.SUCCES;
```

```
% Create output from input production well dimensions
for i = 1:max(input.prodwell.segment);
    output.prodwell.segnr(i,1)     = input.prodwell.segment(i,1);  % segment nr.
    output.prodwell.D_i(i,1)       = input.prodwell.D_i(i,1);      % inner diameter wellbore [m]
    output.prodwell.dl(i,1)        = input.prodwell.dl(i,1);       % length [m]
    output.prodwell.dz(i,1)        = input.prodwell.dz(i,1);       % dz [m]
    output.prodwell.tvd(i,1)       = input.prodwell.tvd(i,1);      % true vertical depth tvd [m]
    output.prodwell.grad_T_g(i,1) = input.prodwell.grad_T_g(i,1); % temperature gradient [m]
    output.prodwell.eps_pipe(i,1) = input.prodwell.eps_pipe(i,1); % absolute pipe roughness [m]
    output.prodwell.k_r(i,1)       = input.prodwell.k_r(i,1);      % rock thermal conductiv.[W/m/K]
    output.prodwell.alfa_r(i,1)    = input.prodwell.alfa_r(i,1);   % rock thermal diffusivity[m2/s]
end
output.prodwell.l(1,1) = 0;                                       % length at begin segment [m]
for i = 2:max(input.prodwell.segment);
    output.prodwell.l(i,1) = output.prodwell.l(i-1,1) + output.prodwell.dl(i-1,1);
end

% Get initial geothermal fluid properties from reservoir output
output.prodwell.geofprops(1,:) = output.reservoir.geofprops(3,:);   % geothermal fluid properties
output.prodwell.P(1,1)         = output.reservoir.geofprops(3,1);   % pressure [bar]
output.prodwell.T(1,1)         = output.reservoir.geofprops(3,2);   % temperature [C]
output.prodwell.h(1,1)         = output.reservoir.geofprops(3,11);  % enthalpy [J/kg]
output.prodwell.chi(1,1)       = output.reservoir.geofprops(3,7);   % gas mass fraction [-]
output.prodwell.v_spec(1,1)    = 1/output.reservoir.geofprops(3,9); % specific volume [m3/kg]
output.prodwell.rho(1,1)       = output.reservoir.geofprops(3,9);   % density [kg/m3]
output.prodwell.c_p(1,1)       = output.reservoir.geofprops(3,10);  % heat capacity [J/kg/K]
output.prodwell.mu(1,1)        = output.reservoir.geofprops(3,12);  % viscosity [Pa*s]
output.prodwell.eps_G(1,1)     = output.reservoir.geofprops(3,8);   % void fraction [-]

% Initial geothermal fluid composition
output.prodwell.w_NaCl_l(1,1) = output.reservoir.geofprops(3,13);   % mass fraction NaCl in liquid
output.prodwell.w_CO2_l(1,1)  = output.reservoir.geofprops(3,16);   % mass fraction CO2 in liquid
output.prodwell.w_CO2_g(1,1)  = output.reservoir.geofprops(3,24);   % mass fraction CO2 in gas
output.prodwell.w_H2O_l(1,1)  = output.reservoir.geofprops(3,19);   % mass fraction H2O in liquid
output.prodwell.w_H2O_g(1,1)  = output.reservoir.geofprops(3,27);   % mass fraction H2O in gas

% Calculate initial properties at inlet production well (first segment)
output.prodwell.u(1,1)      = fCalc_u(input.general.m_gf, output.prodwell.rho(1,1), ...
                              output.prodwell.D_i(1,1));                          % velocity [m/s]
output.prodwell.Re(1,1)     = fCalc_Re(output.prodwell.D_i(1,1), output.prodwell.rho(1,1), ...
                              output.prodwell.u(1,1), output.prodwell.mu(1,1));
                                                                                  % Reynolds number [-]
output.prodwell.f(1,1)      = fCalc_f(output.prodwell.chi(1,1), output.prodwell.eps_pipe(1,1), ...
                              output.prodwell.D_i(1,1), output.prodwell.Re(1,1));
                                                                                  % friction factor [-]
output.prodwell.T_g         = fCalc_T_g(output.prodwell.T(1,1), output.prodwell.grad_T_g, ...
                              output.prodwell.tvd);                % Geothermal temperature [C]
output.prodwell.dQ(1,1)     = fCalc_dQ(output.prodwell.T(1,1), output.prodwell.T_g(1,1), ...
                              output.prodwell.D_i(1,1), output.prodwell.dl(1,1), ...
                              input.general.m_gf, input.general.gamma, input.general.t, ...
                              output.prodwell.k_r(1,1), output.prodwell.alfa_r(1,1));
                                                        % Heat exchange with surroundings [J/kg]
output.prodwell.dE_pot(1,1) = fCalc_dE_pot(input.general.g, output.prodwell.dz(1,1));
                                                                % potential energy change [J/kg]
output.prodwell.dP_f(1,1)   = fCalc_dP_f(output.prodwell.D_i (1,1), output.prodwell.f(1,1), ...
                              output.prodwell.rho(1,1), output.prodwell.u(1,1), ...
                              output.prodwell.dl(1,1));        % frictional pressure change [J/kg]
output.prodwell.dP_hs(1,1)  = fCalc_dP_hs(input.general.g, output.prodwell.rho(1,1), ...
                              output.prodwell.dz(1,1));       % hydrostatic pressure change [J/kg]
output.prodwell.dE_k(1,1)   = 0;
output.prodwell.dP_k(1,1)   = 0;
%% Production well simulation from segment 2 to top
j = 2;                          % 2nd segment number
k = max(input.prodwell.segment); % last segment number

formatSpec = 'Production well calculation.\nPlease wait...';
str        = sprintf(formatSpec);
h          = waitbar(0,str);

% Calculate segments until two segments have a chi > 0 according to the Francke Model
for l = 1:7 %input.settings.nr_it_dp % number of iterations
    for i = j:k

        waitbar(i/max(input.prodwell.segment))
        output.prodwell.P(i,1) = output.prodwell.P(i-1,1) - output.prodwell.dP_hs(i-1,1) - ...
                                 output.prodwell.dP_f(i-1,1) - output.prodwell.dP_k(i-1,1);
                                                                          % pressure pipe [bar]
        output.prodwell.h(i,1) = output.prodwell.h(i-1,1) - output.prodwell.dQ(i-1,1) - ...
                                 output.prodwell.dE_pot(i-1,1) - output.prodwell.dE_k(i-1,1);
                                                                          % enthalpy [J/kg]
if output.prodwell.P(i,1) < 1
    disp('ERROR: Pressure loss in wellbore too high. ACTION: Decrease mass flow')
    close(h)
    msgbox('Pressure loss in wellbore too high. ACTION: Decrease mass flow', 'Error','error');
    stat = status.FAILURE; return;
end;

[geofprops, T_new, w_table] = fCalc_geofprops2 (output.prodwell.P(i,1), ...
                              output.prodwell.T(i-1,1), input.general.w_NaCl, ...
                              input.general.w_CO2, data.H2O_sat,output.prodwell.h(i,1),...
                              output, output.prodwell.h(i-1,1), input, data);
output.prodwell.T(i,1)      = T_new;            % temperature [C]
output.prodwell.chi(i,1)    = geofprops(1,2);   % gas mass fraction [-]
output.prodwell.v_spec(i,1) = 1/geofprops(1,4); % specific volume [m3/kg]
output.prodwell.rho(i,1)    = geofprops(1,4);   % density [kg/m3]
output.prodwell.c_p(i,1)    = geofprops(1,5);   % specific heat capacity [J/kg/K]
output.prodwell.mu(i,1)     = geofprops(1,7);   % viscosity [Pa*s]
output.prodwell.eps_G(i,1)  = geofprops(1,3);   % void fraction [-]

% Drift flux model
if output.prodwell.chi(i,1) > 0 && input.prodwell.DF_model > 1
                              % quality larger than zero && DF_model = 1 --> homogeneous
    output.prodwell.rho_l(i,1) = geofprops(1,15);  % density liquid phase [kg/m3]
    output.prodwell.rho_v(i,1) = geofprops(1,23);  % density vapor phase [kg/m3]
    output.prodwell.mu_l(i,1)  = geofprops(1,18);  % viscosity liquid phase [Pa*s]
    output.prodwell.mu_v(i,1)  = geofprops(1,26);  % viscosity vapor phase [Pa*s]
    output.prodwell.l_E(i,1)   = output.prodwell.l(i,1);
                                                        % length from entrance [m]
    output.prodwell.u_sg(i,1)  = ((output.prodwell.chi(i,1) * input.general.m_gf)/...
                                 geofprops(1,23))/(pi*(output.prodwell.D_i(i,1)/2)^2);
                                                        % superficial gas velocity [m/s]
    output.prodwell.u_sl(i,1)  = (((1-output.prodwell.chi(i,1)) * input.general.m_gf)/...
                                 geofprops(1,15))/(pi*(output.prodwell.D_i(i,1)/2)^2);
                                                        % superficial liquid velocity [m/s]
    [eps_G, FP, u_gu, C_0] = fCalc_eps_G(output.prodwell.T(i,1), geofprops(1,15), ...
                             geofprops(1,23), geofprops(1,18), geofprops(1,26), ...
                             output.prodwell.l_E(i,1), output.prodwell.D_i(i,1), ...
                             output.prodwell.eps_pipe(i,1), output.prodwell.u_sg(i,1), ...
                             output.prodwell.u_sl(i,1), input.general.g, ...
                             output.prodwell.chi(i,1), input.prodwell.DF_model);
    output.prodwell.eps_G(i,1) = eps_G;                                % void fraction
    output.prodwell.FP(i,1)    = cellstr(FP);                          % flow pattern
    output.prodwell.rho(i,1)   = output.prodwell.rho_v(i,1)*output.prodwell.eps_G(i,1)...
                                 + output.prodwell.rho_l(i,1)*...
                                 (1-output.prodwell.eps_G(i,1));       % density [kg/m3]
    output.prodwell.u_gu(i,1)  = u_gu; % drift-flux velocity, u_gas relative to u_m
    output.prodwell.C_0(i,1)   = C_0;  % distribution parameter
end

% Output geothermal fluid composition - mass fractions
output.prodwell.w_NaCl_l(i,1) = w_table(3,2);
output.prodwell.w_CO2_l(i,1)  = w_table(3,3);
output.prodwell.w_CO2_g(i,1)  = w_table(3,4);
output.prodwell.w_H2O_l(i,1)  = w_table(3,5);
output.prodwell.w_H2O_g(i,1)  = w_table(3,6);

% Not used for now - mass fraction at transition
output.prodwell.w_NaCl_l_t(i,1) = w_table(1,2);
output.prodwell.w_CO2_l_t(i,1)  = w_table(1,3);
output.prodwell.w_CO2_g_t(i,1)  = w_table(1,4);
output.prodwell.w_H2O_l_t(i,1)  = w_table(1,5);
output.prodwell.w_H2O_g_t(i,1)  = w_table(1,6);

% Calculate segment properties
output.prodwell.u(i,1)      = fCalc_u(input.general.m_gf, output.prodwell.rho(i,1), ...
                              output.prodwell.D_i(i,1));                  % velocity [m/s]
output.prodwell.Re(i,1)     = fCalc_Re(output.prodwell.D_i(i,1), ...
                              output.prodwell.rho(i,1), output.prodwell.u(i,1), ...
                              output.prodwell.mu(i,1));          % Reynolds number [-]
output.prodwell.f(i,1)      = fCalc_f(output.prodwell.chi(i,1), ...
                              output.prodwell.eps_pipe(i,1), output.prodwell.D_i(i,1),...
                              output.prodwell.Re(i,1));          % friction factor [-]
output.prodwell.dQ(i,1)     = fCalc_dQ(output.prodwell.T(i,1), ...
                              output.prodwell.T_g(i,1), output.prodwell.D_i(i,1), ...
                              output.prodwell.dl(i,1), input.general.m_gf, ...
                              input.general.gamma, input.general.t, ...
                              output.prodwell.k_r(i,1), output.prodwell.alfa_r(i,1));
                                                        % heat exchange with surroundings [J/kg]
```

```
                w_CO2_g(1,1) = 0; w_CO2_g(2,1) = output.prodwell.w_CO2_g(n,1); ...
                    w_CO2_g(3,1) = output.prodwell.w_CO2_g(n+1,1);          % CO2 vapor mass fraction
            end

%               % Create interpolation tables for interpolation between degassing pressures.
%               P(1,1) = output.prodwell.P_degas(m,1); P(2,1) = output.prodwell.P(n-1,1); ...
%                   P(3,1) = output.prodwell.P(n,1); P(4,1) = output.prodwell.P(n+1,1);
% pressure [bar]
%               chi(1,1) = 0; chi(2,1) = output.prodwell.chi(n-1,1); ...
%                   chi(3,1) = output.prodwell.chi(n,1); chi(4,1) = output.prodwell.chi(n+1,1);
% quality [-]
%               w_CO2_l(1,1) = input.general.w_CO2; w_CO2_l(2,1) = output.prodwell.w_CO2_l(n-1,1); ...
%                   w_CO2_l(3,1) = output.prodwell.w_CO2_l(n,1);  w_CO2_l(4,1) = output.prodwell.w_CO2_l(n+1,1);
% CO2 liquid mass fraction
%               w_NaCl_l(1,1) = input.general.w_NaCl; w_NaCl_l(2,1) = output.prodwell.w_NaCl_l(n-1,1); ...
%                   w_NaCl_l(3,1) = output.prodwell.w_NaCl_l(n,1); w_NaCl_l(4,1) =
output.prodwell.w_NaCl_l(n+1,1);          % NaCl liquid mass fraction
%               w_H2O_l(1,1) = 1-input.general.w_NaCl-input.general.w_CO2; ...
%                   w_H2O_l(2,1) = output.prodwell.w_H2O_l(n-1,1); ...
%                   w_H2O_l(3,1) = output.prodwell.w_H2O_l(n,1); w_H2O_l(4,1) = output.prodwell.w_H2O_l(n+1,1);
% H2O liquid mass fraction
%               if input.general.w_CO2 > 0
%                   w_CO2_g(1,1) = 1; w_CO2_g(2,1) = output.prodwell.w_CO2_g(n-1,1) ;...
%                       w_CO2_g(3,1) = output.prodwell.w_CO2_g(n,1); w_CO2_g(4,1) =
output.prodwell.w_CO2_g(n+1,1);         % CO2 vapor mass fraction
%               else
%                   w_CO2_g(1,1) = 0; w_CO2_g(2,1) = output.prodwell.w_CO2_g(n-1,1); ...
%                       w_CO2_g(3,1) = output.prodwell.w_CO2_g(n,1);  w_CO2_g(4,1) =
output.prodwell.w_CO2_g(n+1,1);         % CO2 vapor mass fraction
%               end

                formatSpec = 'Production well calculation (iteration #%d).\nPlease wait...';
                A1         = 1;
                str        = sprintf(formatSpec,A1);
                g          = waitbar(0,str);

                % interpolate properties between P_degas Duan and last segment before P_degas Francke
                for i = m:(n-1)

                    waitbar((i-(m-1))/((n-1)-(m-1)))
                    if i == 1
                        output.prodwell.P(i,1) = output.prodwell.P(1,1);
                        output.prodwell.h(i,1) = output.prodwell.h(1,1);
                    else
                        output.prodwell.P(i,1) = output.prodwell.P(i-1,1) - ...
                                                 output.prodwell.dP_hs(i-1,1) - ...
                                                 output.prodwell.dP_f(i-1,1)
                                                 - output.prodwell.dP_k(i-1,1);  % pressure wellbore [bar]
                        output.prodwell.h(i,1) = output.prodwell.h(i-1,1) - ...
                                                 output.prodwell.dQ(i-1,1) -...
                                                 output.prodwell.dE_pot(i-1,1) - ...
                                                 output.prodwell.dE_k(i-1,1);    % enthalpy [J/kg]
                    end
                    T_int1 = interp1(h_int, T_int, output.prodwell.h(i,1));
                    output.prodwell.chi(i,1)  = interp1(P, chi, output.prodwell.P(i,1),'spline');
                                                                                % quantity [-]
                    output.prodwell.w_CO2_l(i,1)  = interp1(P, w_CO2_l, output.prodwell.P(i,1),'spline');
                                                                                % CO2 liquid mass fraction
                    output.prodwell.w_NaCl_l(i,1) = interp1(P, w_NaCl_l, output.prodwell.P(i,1),'spline');
                                                                                % NaCl liquid mass fraction
                    output.prodwell.w_H2O_l(i,1)  = interp1(P, w_H2O_l, output.prodwell.P(i,1),'spline');
                                                                                % H2O liquid mass fraction
                    output.prodwell.w_CO2_g(i,1)  = interp1(P, w_CO2_g, output.prodwell.P(i,1),'spline');
                                                                                % CO2 vapor mass fraction
                    output.prodwell.w_H2O_g(i,1)  = 1 - output.prodwell.w_CO2_g(i,1);
                                                                                % H2O vapor mass fraction
                    % if calculated quality < 0, than spline interpolation failed, do linear interpolation
%                   if i > 1
%                   if output.prodwell.chi(i,1) < 0 || output.prodwell.chi(i,1) < output.prodwell.chi(i-1,1)
                            output.prodwell.chi(i,1)      = interp1(tvd, chi, output.prodwell.tvd(i,1));
                            %output.prodwell.chi(i,1)      = interp1(P, chi, output.prodwell.P(i,1));
                            output.prodwell.w_CO2_l(i,1)  = interp1(P, w_CO2_l, output.prodwell.P(i,1));
                            output.prodwell.w_NaCl_l(i,1) = interp1(P, w_NaCl_l, output.prodwell.P(i,1));
                            output.prodwell.w_H2O_l(i,1)  = interp1(P, w_H2O_l, output.prodwell.P(i,1));
                            output.prodwell.w_CO2_g(i,1)  = interp1(P, w_CO2_g, output.prodwell.P(i,1));
                            output.prodwell.w_H2O_g(i,1)  = 1 - output.prodwell.w_CO2_g(i,1);
%                       end
%                   end
                    if output.prodwell.w_CO2_g(i,1) > 1
```



```
    output.prodwell.dE_pot(i,1) = fCalc_dE_pot(input.general.g, output.prodwell.dz(i,1));
                                                                    % potential energy [J/kg]

    output.prodwell.dP_f(i,1)   = fCalc_dP_f(output.prodwell.D_i(i,1), ...
                                  output.prodwell.f(i,1), output.prodwell.rho(i,1), ...
                                  output.prodwell.u(i,1), output.prodwell.dl(i,1));
                                                        % frictional pressure change [bar]
    output.prodwell.dP_hs(i,1)  = fCalc_dP_hs(input.general.g, output.prodwell.rho(i,1), ...
                                  output.prodwell.dz(i,1));% hydrostatic pressure change [bar]
    output.prodwell.dE_k(i,1)   = fCalc_dE_k(output.prodwell.u(i,1),output.prodwell.u(i-1,1));
                                                                    % kinetic energy [J/kg]
    output.prodwell.dP_k(i,1)   = fCalc_dP_k(output.prodwell.rho(i,1), ...
                                  output.prodwell.u(i,1),output.prodwell.u(i-1,1));
                                                        % kinetic pressure change [J/kg]
    % if i == size(input.prodwell.tvd,1)
    %    output.prodwell.P(i+1,1) = output.prodwell.P(i,1) - output.prodwell.dP_hs(i,1) - ...
    %                               output.prodwell.dP_f(i,1); % pressure pipe [bar]
    %    output.prodwell.h(i+1,1) = output.prodwell.h(i,1) - output.prodwell.Q(i,1) - ...
    %                               output.prodwell.E_pot(i,1); %enthalpy [J/kg]
    % end

    %% Check if two segments have a significant gas mass fraction
    if l == 1
        if output.prodwell.chi(i,1) > 0.0001 && output.prodwell.chi(i-1,1) > 0.0001
            output.prodwell.P_old = output.prodwell.P(i-2,1);
            break % start interpolation from P_degas Duan and Sun (2003)
        end
    elseif l > 1 && i >= (j+1)
        if output.prodwell.chi(i,1) > 0.0001 && output.prodwell.chi(i-1,1) > 0.0001
            output.prodwell.P_old = output.prodwell.P(i-2,1);
            break % start interpolation from P_degas Duan and Sun (2003)
        end
    end
end

% Find P_degas from Duan and Sun (2003)
output.prodwell.P_degas = interp3(data.m_NaCl_degas, data.T_degas, data.m_CO2_degas, ...
                          data.P_degas, input.general.m_NaCl, output.prodwell.T + 273.15,...
                          input.general.m_CO2);                     % degassing pressure [bar]
if i == k
    if output.prodwell.chi(k,1) < 0.0001 && output.prodwell.P(k,1) < ...
       output.prodwell.P_degas(k,1)
        [input, output, geofprops, i] = fCalc_prodwell_virtual(input, output, data, k);
    end
    if output.prodwell.chi(k,1) > 0.0001 && output.prodwell.P(k,1) < ...
            output.prodwell.P_degas(k,1) && output.prodwell.chi(k-1) == 0
        [input, output, geofprops, i] = fCalc_prodwell_virtual(input, output, data, k);
    end
end

% Find first segment number where degassing pressure Duan and Sun(2003) is above segment ...
% pressure from Francke Model.
m = find((output.prodwell.P_degas - output.prodwell.P) > 0,1);
if isempty(m) == 1 % if P_degas Duan is not above P_degas Francke
    m = i;
end
n = i-1; % find(output.prodwell.chi(n:i-1,1) > 0.001,1) + (n - 1);


if m < n % if P_degas Duan is above P_degas Francke
    % Create interpolation tables for interpolation between degassing pressures.
    P(1,1) = output.prodwell.P_degas(m,1); P(2,1) = output.prodwell.P(n,1); ...
        P(3,1) = output.prodwell.P(n+1,1);                          % pressure [bar]
    T_int (1,1) = output.prodwell.T(m-1,1); T_int (2,1) = output.prodwell.T(n,1);
    h_int (1,1) = output.prodwell.h(m-1,1); h_int (2,1) = output.prodwell.h(n,1);
    tvd(1,1) = output.prodwell.tvd(m-1,1); tvd(2,1) = output.prodwell.tvd(n,1);
    chi(1,1) = 0; chi(2,1) = output.prodwell.chi(n,1); ...
        chi(3,1) = output.prodwell.chi(n+1,1);                      % quality [-]
    w_CO2_l(1,1) = input.general.w_CO2; w_CO2_l(2,1) = output.prodwell.w_CO2_l(n,1); ...
        w_CO2_l(3,1) = output.prodwell.w_CO2_l(n+1,1);              % CO2 liquid mass fraction
    w_NaCl_l(1,1) = input.general.w_NaCl; w_NaCl_l(2,1) = output.prodwell.w_NaCl_l(n,1); ...
        w_NaCl_l(3,1) = output.prodwell.w_NaCl_l(n+1,1);            % NaCl liquid mass fraction
    w_H2O_l(1,1) = 1-input.general.w_NaCl-input.general.w_CO2; ...
        w_H2O_l(2,1) = output.prodwell.w_H2O_l(n,1); ...
        w_H2O_l(3,1) = output.prodwell.w_H2O_l(n+1,1);              % H2O liquid mass fraction
    if input.general.w_CO2 > 0
        w_CO2_g(1,1) = 1; w_CO2_g(2,1) = output.prodwell.w_CO2_g(n,1) ;...
            w_CO2_g(3,1) = output.prodwell.w_CO2_g(n+1,1);          % CO2 vapor mass fraction
    else
```

```
            end
            close(g)
            i = n;
            j = n;
            k = max(output.prodwell.segnr);
        end

        if m >= n % If degassing according to Duan and Sun (2003) starts later than Francke (2014)
            break
        end
        if abs(output.prodwell.P(i-1,1) - output.prodwell.P_old) < input.settings.dP_abs_pw
                                                    % If calculation has iterated to user-defined error
            break
        end

    end

    if max(input.prodwell.segment) < max(output.prodwell.segnr)
        output = fChange_prodwell(input, output);
        close(h)
        return
    end
    %n = 2; %validation of sodium chloride solution
    %% Proceed with segment of flash horizon of Francke (2014)
    for i = n+1:max(input.prodwell.segment) + 1

        if i == max(input.prodwell.segment) + 1
            output.prodwell.segnr(i,1)     = output.prodwell.segnr(i-1,1)+1; % segment nr.
            output.prodwell.D_i(i,1)       = input.prodwell.D_i(i-1,1);   % inner diameter wellbore [m]
            output.prodwell.dl(i,1)        = input.prodwell.dl(i-1,1);    % length [m]
            output.prodwell.dz(i,1)        = input.prodwell.dz(i-1,1);    % dz [m]
            output.prodwell.tvd(i,1)       = 0;                           % true vertical depth tvd [m]
            output.prodwell.grad_T_g(i,1)  = input.prodwell.grad_T_g(i-1,1); % temperature gradient [m]
            output.prodwell.eps_pipe(i,1)  = input.prodwell.eps_pipe(i-1,1); % abs. pipe roughness [m]
            output.prodwell.k_r(i,1)       = input.prodwell.k_r(i-1,1);% rock thermal conductiv.[W/m/K]
            output.prodwell.alfa_r(i,1)= input.prodwell.alfa_r(i-1,1);% rock thermal diffusivity[m2/s]
            output.prodwell.l(i,1)         = output.prodwell.l(i-1,1) + output.prodwell.dl(i-1,1);
            output.prodwell.T_g(i,1)       = output.prodwell.T_g(i-1,1);
        end
        waitbar(i/max(input.prodwell.segment))

%         if i == n || output.prodwell.chi(i-1,1) < output.prodwell.chi(n-1,1)
%             output.prodwell.P(i,1) = output.prodwell.P(i-1,1) - (output.prodwell.P(i-2,1) -
output.prodwell.P(i-1,1));
%             output.prodwell.h(i,1) = output.prodwell.h(i-1,1) - (output.prodwell.h(i-2,1) -
output.prodwell.h(i-1,1));
%         else
            output.prodwell.P(i,1) = output.prodwell.P(i-1,1) - output.prodwell.dP_hs(i-1,1) - ...
                                     output.prodwell.dP_f(i-1,1) - output.prodwell.dP_k(i-1,1);
                                                                          % pressure wellbore [bar]
            output.prodwell.h(i,1) = output.prodwell.h(i-1,1) - output.prodwell.dQ(i-1,1) - ...
                                     output.prodwell.dE_pot(i-1,1) - output.prodwell.dE_k(i-1,1);
                                                                                % enthalpy [J/kg]
%         end

        if output.prodwell.P(i,1) < input.general.P_atm         % minimum pressure of wellbore
            disp('ERROR: Pressure loss in wellbore too high. ACTION: Decrease mass flow')
            close(h)
            msgbox('Pressure loss in wellbore too high. ACTION: Decrease mass flow', 'Error','error');
            stat = status.FAILURE; return;
        end;

        [geofprops, T_new, w_table] = fCalc_geofprops2(output.prodwell.P(i,1), ...
                                      output.prodwell.T(i-1,1), input.general.w_NaCl, ...
                                      input.general.w_CO2, data.H2O_sat, output.prodwell.h(i,1), ...
                                      output, output.prodwell.h(i-1,1), input, data);
        output.prodwell.T(i,1)      = T_new;            % temperature [C]
        output.prodwell.chi(i,1)    = geofprops(1,2);   % gas mass fraction [-]
        output.prodwell.v_spec(i,1) = 1/geofprops(1,4); % specific volume [m3/kg]
        output.prodwell.rho(i,1)    = geofprops(1,4);   % density [kg/m3]
        output.prodwell.c_p(i,1)    = geofprops(1,5);   % specific heat capacity [J/kg/K]
        output.prodwell.mu(i,1)     = geofprops(1,7);   % viscosity [Pa*s]
        output.prodwell.eps_G(i,1)  = geofprops(1,3);   % void fraction [-]

        % Drift flux model
        if input.prodwell.DF_model > 1 && output.prodwell.chi(i,1) > 0 % DF_model = 1 --> homogeneous
            output.prodwell.rho_l(i,1) = geofprops(1,15); % density liquid phase [kg/m3]
            output.prodwell.rho_v(i,1) = geofprops(1,23); % density gas phase [kg/m3]
            output.prodwell.mu_l(i,1)  = geofprops(1,18); % viscosity liquid phase [Pa*s]
```



```
    output.prodwell.w_CO2_g(i,1) = 1;
end

% Invoke fCalc_geofprops3 for single liquid and single vapor properties calculation...
% and calculate total properties
[T_new, rho_m, c_p_m, mu_m, eps_G, rho_v, rho_l, mu_v, mu_l] = fCalc_geofprops3 ...
    (output.prodwell.P(i,1), output.prodwell.T(i-1,1), ...
    output.prodwell.w_CO2_l(i,1), output.prodwell.w_NaCl_l(i,1), ...
    output.prodwell.w_CO2_g(i,1), output.prodwell.h(i,1), output.prodwell.chi(i,1),...
    output.prodwell.h(i-1,1), input, output.prodwell.T(i-2,1),1,T_int1);
                                                    % geothermal fluid properties
output.prodwell.T(i,1)      = T_new;                % temperature [C]
output.prodwell.rho(i,1)    = rho_m;                % density [kg/m3]
output.prodwell.c_p(i,1)    = c_p_m;                % specific heat capacity [J/kg/K]
output.prodwell.mu(i,1)     = mu_m;                 % viscosity [Pa*s]
output.prodwell.eps_G(i,1) = eps_G;                 % void fraction [-]

% Drift flux model
if input.prodwell.DF_model > 1          % DF_model = 1 --> homogeneous
    output.prodwell.rho_l(i,1) = rho_l;  % density liquid phase [kg/m3]
    output.prodwell.rho_v(i,1) = rho_v;  % density gas phase [kg/m3]
    output.prodwell.mu_l(i,1)  = mu_l;   % viscosity liquid phase [Pa*s]
    output.prodwell.mu_v(i,1)  = mu_v;   % viscosity gas phase [Pa*s]
    p = find(output.prodwell.chi > 0,1); % segment number with flash horizon
    output.prodwell.l_E(i,1)   = output.prodwell.l(i,1);
                                                    % length from entrance [m]
    output.prodwell.u_sg(i,1)  = ((output.prodwell.chi(i,1)* input.general.m_gf)/...
                                  rho_v)/(pi*(output.prodwell.D_i(i,1)/2)^2);
                                            % superficial gas velocity [m/s]
    output.prodwell.u_sl(i,1)  = (((1-output.prodwell.chi(i,1)) * ...
                                  input.general.m_gf)/rho_l)/(pi*...
                                  (output.prodwell.D_i(i,1)/2)^2);
                                           % superficial liquid velocity [m/s]
    [eps_G,FP,u_gu,C_0] = fCalc_eps_G(output.prodwell.T(i,1), rho_l, rho_v, mu_l, ...
                          mu_v, output.prodwell.l_E(i,1), output.prodwell.D_i(i,1),...
                          output.prodwell.eps_pipe(i,1), output.prodwell.u_sg(i,1),...
                          output.prodwell.u_sl(i,1), input.general.g, ...
                          output.prodwell.chi(i,1), input.prodwell.DF_model);
                                                    % void fraction [-]
    output.prodwell.eps_G(i,1) = eps_G;       % void fraction [-]
    output.prodwell.FP(i,1)    = cellstr(FP); % flow pattern
    output.prodwell.rho(i,1)   = output.prodwell.rho_v(i,1)*...
                                 output.prodwell.eps_G(i,1) ...
                                 + output.prodwell.rho_l(i,1)*...
                                 (1-output.prodwell.eps_G(i,1));     % density [kg/m3]
    output.prodwell.u_gu(i,1)  = u_gu; % drift-flux velocity, u_g relative to u_m[m/s]
    output.prodwell.C_0(i,1)   = C_0;  % distribution parameter
end

% Recalculate segment properties
output.prodwell.u(i,1)      = fCalc_u(input.general.m_gf, output.prodwell.rho(i,1),...
                              output.prodwell.D_i(i,1));          % velocity [m/s]
output.prodwell.Re(i,1)     = fCalc_Re(output.prodwell.D_i(i,1), ...
                              output.prodwell.rho(i,1), output.prodwell.u(i,1), ...
                              output.prodwell.mu(i,1));           % Reynolds number [-]
output.prodwell.f(i,1)      = fCalc_f(output.prodwell.chi(i,1), ...
                              output.prodwell.eps_pipe(i,1), ...
                              output.prodwell.D_i(i,1), output.prodwell.Re(i,1));
                                                    % friction factor [-]
output.prodwell.dQ(i,1)     = fCalc_dQ(output.prodwell.T(i,1), ...
                              output.prodwell.T_g(i,1), output.prodwell.D_i(i,1), ...
                              output.prodwell.dl(i,1), input.general.m_gf, ...
                              input.general.gamma, input.general.t, ...
                              output.prodwell.k_r(i,1), output.prodwell.alfa_r(i,1));
                                    % Heat exchange with surroundings [J/kg]
output.prodwell.dE_pot(i,1) = fCalc_dE_pot(input.general.g, output.prodwell.dz(i,1));
                                            % potential energy change [J/kg]
output.prodwell.dP_f(i,1)   = fCalc_dP_f(output.prodwell.D_i(i,1), ...
                              output.prodwell.f(i,1), output.prodwell.rho(i,1), ...
                              output.prodwell.u(i,1), output.prodwell.dl(i,1));
                                           % frictional pressure change [bar]
output.prodwell.dP_hs(i,1)  = fCalc_dP_hs(input.general.g, ...
                              output.prodwell.rho(i,1), output.prodwell.dz(i,1));
                                         % hydrostatic pressure change [bar]
output.prodwell.dE_k(i,1)   = fCalc_dE_k(output.prodwell.u(i,1),...
                              output.prodwell.u(i-1,1));       % kinetic energy [J/kg]
output.prodwell.dP_k(i,1)  =  fCalc_dP_k(output.prodwell.rho(i,1), ...
                              output.prodwell.u(i,1),output.prodwell.u(i-1,1));
                                           % kinetic pressure change [J/kg]
```

```
    output.prodwell.mu_v(i,1)  = geofprops(1,26); % viscosity gas phase [Pa*s]
    p = find(output.prodwell.chi > 0,1); % segment number with flash horizon
    output.prodwell.l_E(i,1)   = output.prodwell.l(i,1);
                                                  % length from entrance [m]
    output.prodwell.u_sg(i,1)  = ((output.prodwell.chi(i,1) * input.general.m_gf)/...
                                  geofprops(1,23))/(pi*(output.prodwell.D_i(i,1)/2)^2);
                                                  % superficial gas velocity [m/s]
    output.prodwell.u_sl(i,1)  = (((1-output.prodwell.chi(i,1)) * input.general.m_gf)/...
                                  geofprops(1,15))/(pi*(output.prodwell.D_i(i,1)/2)^2);
                                                  % superficial liquid velocity [m/s]
    [eps_G,FP,u_gu,C_0] = fCalc_eps_G(output.prodwell.T(i,1), geofprops(1,15), ...
                          geofprops(1,23), geofprops(1,18), geofprops(1,26), ...
                          output.prodwell.l_E(i,1), output.prodwell.D_i(i,1), ...
                          output.prodwell.eps_pipe(i,1), output.prodwell.u_sg(i,1), ...
                          output.prodwell.u_sl(i,1), input.general.g, ...
                          output.prodwell.chi(i,1), input.prodwell.DF_model);
                                                                  % void fraction [-]
    output.prodwell.eps_G(i,1) = eps_G;       % void fraction [-]
    output.prodwell.FP(i,1)    = cellstr(FP); % flow pattern
    output.prodwell.rho(i,1)   = output.prodwell.rho_v(i,1)*output.prodwell.eps_G(i,1) + ...
                                 output.prodwell.rho_l(i,1)*(1-output.prodwell.eps_G(i,1));
                                                                  % density [kg/m3]
    output.prodwell.u_gu(i,1)  = u_gu; % drift-flux velocity, u_g relative to u_m [m/s] [m/s]
    output.prodwell.C_0(i,1)   = C_0;  % distribution parameter
end
% Output geofluid composition
output.prodwell.w_NaCl_l(i,1) = w_table(3,2);
output.prodwell.w_CO2_l(i,1)  = w_table(3,3);
output.prodwell.w_CO2_g(i,1)  = w_table(3,4);
output.prodwell.w_H2O_l(i,1)  = w_table(3,5);
output.prodwell.w_H2O_g(i,1)  = w_table(3,6);

output.prodwell.u(i,1)      = fCalc_u(input.general.m_gf, output.prodwell.rho(i,1), ...
                              output.prodwell.D_i(i,1)); % velocity [m/s]
output.prodwell.Re(i,1)     = fCalc_Re(output.prodwell.D_i(i,1), output.prodwell.rho(i,1),...
                              output.prodwell.u(i,1), output.prodwell.mu(i,1));
                                                                  % Reynolds number [-]
output.prodwell.f(i,1)      = fCalc_f(output.prodwell.chi(i,1), ...
                              output.prodwell.eps_pipe(i,1), output.prodwell.D_i(i,1), ...
                              output.prodwell.Re(i,1));  % friction factor [-]
output.prodwell.dQ(i,1)     = fCalc_dQ(output.prodwell.T(i,1), output.prodwell.T_g(i,1), ...
                              output.prodwell.D_i (i,1), output.prodwell.dl(i,1), ...
                              input.general.m_gf, input.general.gamma, input.general.t, ...
                              output.prodwell.k_r(i,1), output.prodwell.alfa_r(i,1));
                                                  % Heat exchange with surroundings [J/kg]
output.prodwell.dE_pot(i,1) = fCalc_dE_pot(input.general.g, output.prodwell.dz(i,1));
                                                  % potential energy change [J/kg]
output.prodwell.dP_f(i,1)   = fCalc_dP_f(output.prodwell.D_i(i,1), output.prodwell.f(i,1),...
                              output.prodwell.rho(i,1), output.prodwell.u(i,1), ...
                              output.prodwell.dl(i,1));  % frictional pressure change [bar]
output.prodwell.dP_hs(i,1)  = fCalc_dP_hs(input.general.g, output.prodwell.rho(i,1), ...
                              output.prodwell.dz(i,1));  % hydrostatic pressure change [bar]
output.prodwell.dE_k(i,1)   = fCalc_dE_k(output.prodwell.u(i,1),...
                                  output.prodwell.u(i-1,1));       % kinetic energy [J/kg]
output.prodwell.dP_k(i,1)  =  fCalc_dP_k(output.prodwell.rho(i,1), ...
                               output.prodwell.u(i,1),output.prodwell.u(i-1,1));
                                                  % kinetic pressure change [J/kg]
    end
    close(h)
end
```

```
%% fCalc_prodwell_GL

% Simulation of a production well with a gas lift system
% Frank Niewold
% Released version 1.0, February 2017

function [input, output, stat, geofprops] = fCalc_prodwell_GL(input, output, data, status)

    % Succesfull simulation
    stat = status.SUCCES;

    % Create output from input production well dimensions
    for i = 1:max(input.prodwell_GL.segment);
        output.prodwell_GL.segnr(i,1)    = input.prodwell_GL.segment(i,1);  % segment nr.
        output.prodwell_GL.D_i(i,1)      = input.prodwell_GL.D_i(i,1);      % in diameter wellbore [m]
        output.prodwell_GL.dl(i,1)       = input.prodwell_GL.dl(i,1);       % length [m]
        output.prodwell_GL.dz(i,1)       = input.prodwell_GL.dz(i,1);       % dz [m]
        output.prodwell_GL.tvd(i,1)      = input.prodwell_GL.tvd(i,1);      % tvd [m]
        output.prodwell_GL.grad_T_g(i,1) = input.prodwell_GL.grad_T_g(i,1); % temperature gradient [m]
        output.prodwell_GL.eps_pipe(i,1) = input.prodwell_GL.eps_pipe(i,1); % abs. pipe roughness [m]
        output.prodwell_GL.k_r(i,1)      = input.prodwell_GL.k_r(i,1);      % rock thermal cond.[W/m/K]
        output.prodwell_GL.alfa_r(i,1)   = input.prodwell_GL.alfa_r(i,1);   % rock thermal diff.[m2/s]
        output.prodwell_GL.m_gf(i,1)     = input.general.m_gf;              % mass flow rate [kg/s]
        output.prodwell_GL.m_GL          = input.prodwell_GL.m_GL;          % mass flow rate GL [kg/s]
        output.prodwell_GL.w_NaCl(i,1)   = input.general.w_NaCl;
        output.prodwell_GL.w_CO2(i,1)    = input.general.w_CO2;
        output.prodwell_GL.w_H2O(i,1)    = 1 - input.general.w_NaCl - input.general.w_CO2;
        output.prodwell_GL.m_NaCl(i,1)   = input.general.m_NaCl;
        output.prodwell_GL.m_CO2(i,1)    = input.general.m_CO2;
    end
    output.prodwell_GL.l(1,1) = 0;                                  % length at begin segment [m]
    for i = 2:max(input.prodwell_GL.segment);
        output.prodwell_GL.l(i,1) = output.prodwell_GL.l(i-1,1) + output.prodwell_GL.dl(i-1,1);
    end

    % Get initial geothermal fluid properties from reservoir output
    output.prodwell_GL.geofprops(1,:) = output.reservoir.geofprops(3,:);   % geothermal fluid props
    output.prodwell_GL.P(1,1)         = output.reservoir.geofprops(3,1);   % pressure [bar]
    output.prodwell_GL.T(1,1)         = output.reservoir.geofprops(3,2);   % temperature [C]
    output.prodwell_GL.h(1,1)         = output.reservoir.geofprops(3,11);  % enthalpy [J/kg]
    output.prodwell_GL.chi(1,1)       = output.reservoir.geofprops(3,7);   % gas mass fraction [-]
    output.prodwell_GL.v_spec(1,1)    = 1/output.reservoir.geofprops(3,9); % specific volume [m3/kg]
    output.prodwell_GL.rho(1,1)       = output.reservoir.geofprops(3,9);   % density [kg/m3]
    output.prodwell_GL.c_p(1,1)       = output.reservoir.geofprops(3,10);  % heat capacity [J/kg/K]
    output.prodwell_GL.mu(1,1)        = output.reservoir.geofprops(3,12);  % viscosity [Pa*s]
    output.prodwell_GL.eps_G(1,1)     = output.reservoir.geofprops(3,8);   % void fraction [-]

    % Initial geothermal fluid composition
    output.prodwell_GL.w_NaCl_l(1,1) = output.reservoir.geofprops(3,13);   % mass fraction NaCl in liquid
    output.prodwell_GL.w_CO2_l(1,1)  = output.reservoir.geofprops(3,16);   % mass fraction CO2 in liquid
    output.prodwell_GL.w_CO2_g(1,1)  = output.reservoir.geofprops(3,24);   % mass fraction CO2 in gas
    output.prodwell_GL.w_H2O_l(1,1)  = output.reservoir.geofprops(3,19);   % mass fraction H2O in liquid
    output.prodwell_GL.w_H2O_v(1,1)  = output.reservoir.geofprops(3,27);   % mass fraction H2O in gas

    % Calculate initial properties at inlet production well (first segment)
    output.prodwell_GL.u(1,1)      = fCalc_u(output.prodwell_GL.m_gf(1,1), ...
                                     output.prodwell_GL.rho(1,1), output.prodwell_GL.D_i(1,1));
                                                                        % velocity [m/s]
    output.prodwell_GL.Re(1,1)     = fCalc_Re(output.prodwell_GL.D_i(1,1), ...
                                     output.prodwell_GL.rho(1,1), output.prodwell_GL.u(1,1), ...
                                     output.prodwell_GL.mu(1,1));       % Reynolds number [-]
    output.prodwell_GL.f(1,1)      = fCalc_f(output.prodwell_GL.chi(1,1), ...
                                     output.prodwell_GL.eps_pipe(1,1), output.prodwell_GL.D_i(1,1),...
                                     output.prodwell_GL.Re(1,1));       % friction factor [-]
    output.prodwell_GL.T_g     = fCalc_T_g(output.prodwell_GL.T(1,1), output.prodwell_GL.grad_T_g, ...
                                 output.prodwell_GL.tvd);               % Geothermal temperature [C]
    output.prodwell_GL.dQ(1,1) = fCalc_dQ(output.prodwell_GL.T(1,1), output.prodwell_GL.T_g(1,1), ...
                                 output.prodwell_GL.D_i(1,1), output.prodwell_GL.dl(1,1), ...
                                 output.prodwell_GL.m_gf(1,1), input.general.gamma, ...
                                 input.general.t, output.prodwell_GL.k_r(1,1),...
                                 output.prodwell_GL.alfa_r(1,1)); % Heat exchange with surround.[J/kg]
    output.prodwell_GL.dE_pot(1,1) = fCalc_dE_pot(input.general.g, output.prodwell_GL.dz(1,1));
                                                                % potential energy change [J/kg]
    output.prodwell_GL.dP_f(1,1)   = fCalc_dP_f(output.prodwell_GL.D_i (1,1), ...
                                     output.prodwell_GL.f(1,1), output.prodwell_GL.rho(1,1), ...
                                     output.prodwell_GL.u(1,1), output.prodwell_GL.dl(1,1));
                                                                % frictional pressure change [J/kg]
    output.prodwell_GL.dP_hs(1,1)  = fCalc_dP_hs(input.general.g, output.prodwell_GL.rho(1,1), ...
                                     output.prodwell_GL.dz(1,1)); % hydrostatic pressure change [J/kg]

    %% Production well simulation from segment 2 to top
    j = 2;                                % 2nd segment number
    k = max(input.prodwell_GL.segment); % last segment number

    formatSpec = 'Calculation production well with gas lift.\nPlease wait...';
    str        = sprintf(formatSpec);
    h          = waitbar(0,str);

    for a = 1:1
    % Calculate segments until two segments have a chi > 0 according to the Francke Model
    for l = 1:10 %input.settings.nr_it_dp % number of iterations
        for i = j:k

            waitbar(i/max(input.prodwell_GL.segment))

            output.prodwell_GL.P(i,1) = output.prodwell_GL.P(i-1,1) - ...
```

```
                                output.prodwell_GL.dP_hs(i-1,1) - ...
                                output.prodwell_GL.dP_f(i-1,1);          % pressure pipe [bar]
            output.prodwell_GL.h(i,1) = output.prodwell_GL.h(i-1,1) - output.prodwell_GL.dQ(i-1,1) ...
                                - output.prodwell_GL.dE_pot(i-1,1);          % enthalpy [J/kg]

            if i == input.prodwell_GL.segnr_GL
                output.prodwell_GL.m_gf(i:k,1) = output.prodwell_GL.m_gf(i,1) + ...
                                                 output.prodwell_GL.m_GL;
                                                 % m_GL = mass flow rate lift gas
                output.prodwell_GL.w_NaCl(i:k,1) = output.prodwell_GL.w_NaCl(i,1) * ...
                                                   output.prodwell_GL.m_gf(i-1,1)/...
                                                   output.prodwell_GL.m_gf(i,1);
                output.prodwell_GL.w_CO2(i:k,1)  = (output.prodwell_GL.w_CO2(i,1) * ...
                                                   output.prodwell_GL.m_gf(i-1,1) + ...
                                                   output.prodwell_GL.m_GL)/...
                                                   output.prodwell_GL.m_gf(i,1);
                output.prodwell_GL.w_H2O(i:k,1)  = 1 - output.prodwell_GL.w_NaCl(i,1) - ...
                                                   output.prodwell_GL.w_CO2(i,1);
                output.prodwell_GL.w_CO2_GL = 1;

                %% obtain new molalities from Excel
                Excel  = actxGetRunningServer('Excel.Application');
                Sheets = Excel.ActiveWorkBook.Sheets;
                sheet2 = get(Sheets, 'Item', 1);
                invoke(sheet2, 'Activate');
                sheet  = Excel.Activesheet;
                sheet.set('Range', 'C3', output.prodwell.P(i,1));
                sheet.set('Range', 'C4', output.prodwell.T(i,1));
                sheet.set('Range', 'C8', output.prodwell_GL.w_NaCl(i,1));
                sheet.set('Range', 'C11', output.prodwell_GL.w_CO2(i,1));
                range  = sheet.get('Range', 'D8:D13');
                range2 = sheet.get('Range', 'I9');
                range.Value;
                range2.Value;
                data_FM  = range.Value;
                data1_FM = range2.Value;
                output.prodwell_GL.m_NaCl(i:k,1) = cell2mat(data_FM(1,1));
                output.prodwell_GL.m_CO2(i:k,1)  = cell2mat(data_FM(4,1));
                output.prodwell_GL.h(i,1) = data1_FM;

                sheet2 = get(Sheets, 'Item', 2);
                invoke(sheet2, 'Activate');
                sheet  = Excel.Activesheet;
                sheet.set('Range', 'C2', output.prodwell.P(i,1));
                sheet.set('Range', 'C3', output.prodwell.T(i,1));
                sheet.set('Range', 'C5', output.prodwell_GL.w_CO2_GL);
                range  = sheet.get('Range', 'G5');
                range.Value;
                output.prodwell_GL.h_GL = range.Value;

                sheet2 = get(Sheets, 'Item', 3);
                invoke(sheet2, 'Activate');
                sheet  = Excel.Activesheet;
                sheet.set('Range', 'C3', output.prodwell.P(i,1));
                sheet.set('Range', 'C4', output.prodwell.T(i,1));
                sheet.set('Range', 'C6', output.prodwell_GL.w_NaCl(i,1));
                range  = sheet.get('Range', 'G6');
                range.Value;
                output.prodwell_GL.h_GL_l = range.Value;


%                 output.prodwell_GL.h(i,1) = output.prodwell_GL.h(i-1,1) - output.prodwell_GL.dQ(i-1,1) - ...
%                                             output.prodwell_GL.dE_pot(i-1,1);          % enthalpy [J/kg]
%                 output.prodwell_GL.h(i,1) = (output.prodwell_GL.h(i,1) * output.prodwell_GL.m_gf(i-1,1) +
...
%                                             output.prodwell_GL.h_GL *
output.prodwell_GL.m_GL)/output.prodwell_GL.m_gf(i,1);
            end

            if output.prodwell_GL.P(i,1) < 1
                disp('ERROR: Pressure loss in wellbore too high. ACTION: Decrease mass flow')
                close(h)
                msgbox('Pressure loss in wellbore too high. ACTION: Decrease mass flow', 'Error','error');
                stat = status.FAILURE; return;
            end
%             if i == input.prodwell_GL.segnr_GL
%                 return
%             end
```



```
[geofprops, T_new, w_table] = fCalc_geofprops2(output.prodwell_GL.P(i,1), ...
                              output.prodwell_GL.T(i-1,1), output.prodwell_GL.w_NaCl(i,1), ...
                              output.prodwell_GL.w_CO2(i,1), data.H2O_sat, ...
                              output.prodwell_GL.h(i,1), output,...
                              output.prodwell_GL.h(i-1,1), input, data);
output.prodwell_GL.T(i,1)      = T_new;            % temperature [C}
output.prodwell_GL.P_dFM(i,1)  = geofprops(1,1);   % degassing pressure Francke Model[bar]
output.prodwell_GL.chi(i,1)    = geofprops(1,2);   % gas mass fraction [-}
output.prodwell_GL.v_spec(i,1) = 1/geofprops(1,4); % specific volume {m3/kg]
output.prodwell_GL.rho(i,1)    = geofprops(1,4);   % density [kg/m3]
output.prodwell_GL.c_p(i,1)    = geofprops(1,5);   % specific heat capacity [J/kg/K]
output.prodwell_GL.mu(i,1)     = geofprops(1,7);   % viscosity [Pa*s]
output.prodwell_GL.eps_G(i,1)  = geofprops(1,3);   % void fraction [-]

% Drift flux model
if output.prodwell_GL.chi(i,1) > 0 && input.prodwell_GL.DF_model > 1
                                % quality larger than zero && DF_model = 1 --> homogeneous
    output.prodwell_GL.rho_l(i,1) = geofprops(1,15);  % density liquid phase [kg/m3]
    output.prodwell_GL.rho_v(i,1) = geofprops(1,23);  % density vapor phase [kg/m3]
    output.prodwell_GL.mu_l(i,1)  = geofprops(1,18);  % viscosity liquid phase [Pa*s]
    output.prodwell_GL.mu_v(i,1)  = geofprops(1,26);  % viscosity vapor phase [Pa*s]
    output.prodwell_GL.l_E(i,1)   = output.prodwell_GL.l(i,1) - output.prodwell_GL.l(i,1);
                                    % length from entrance or flash horizon [m]
    output.prodwell_GL.u_sg(i,1)  = ((output.prodwell_GL.chi(i,1) * ...
                                    output.prodwell_GL.m_gf(i,1))/...
                                    geofprops(1,23))/...
                                    (pi*(output.prodwell_GL.D_i(i,1)/2)^2);
                                                % superficial gas velocity [m/s]
    output.prodwell_GL.u_sl(i,1)  = (((1-output.prodwell_GL.chi(i,1)) * ...
                                    output.prodwell_GL.m_gf(i,1))/...
                                    geofprops(1,15))/...
                                    (pi*(output.prodwell_GL.D_i(i,1)/2)^2);
                                                % superficial liquid velocity [m/s]
    [eps_G, FP, u_gu, C_0] = fCalc_eps_G(output.prodwell_GL.T(i,1), geofprops(1,15), ...
                             geofprops(1,23), geofprops(1,18), geofprops(1,26), ...
                             output.prodwell_GL.l_E(i,1), output.prodwell_GL.D_i(i,1), ...
                             output.prodwell_GL.eps_pipe(i,1), ...
                             output.prodwell_GL.u_sg(i,1), ...
                             output.prodwell_GL.u_sl(i,1), input.general.g, ...
                             output.prodwell_GL.chi(i,1), input.prodwell_GL.DF_model);
    output.prodwell_GL.eps_G(i,1) = eps_G;                          % void fraction
    output.prodwell_GL.FP(i,1)    = cellstr(FP);                    % flow pattern
    output.prodwell_GL.rho(i,1)   = output.prodwell_GL.rho_v(i,1) * ...
                                    output.prodwell_GL.eps_G(i,1)...
                                    + output.prodwell_GL.rho_l(i,1)*...
                                    (1-output.prodwell_GL.eps_G(i,1));   % density [kg/m3]
    output.prodwell_GL.u_gu(i,1)  = u_gu; % drift-flux velocity, u_gas relative to u_m
    output.prodwell_GL.C_0(i,1)   = C_0;  % distribution parameter
end

% Output geothermal fluid composition - mass fractions
output.prodwell_GL.w_NaCl_l(i,1) = w_table(3,2);
output.prodwell_GL.w_CO2_l(i,1)  = w_table(3,3);
output.prodwell_GL.w_CO2_g(i,1)  = w_table(3,4);
output.prodwell_GL.w_H2O_l(i,1)  = w_table(3,5);
output.prodwell_GL.w_H2O_v(i,1)  = w_table(3,6);

% Not used for now - mass fraction at transition
output.prodwell_GL.w_NaCl_l_t(i,1) = w_table(1,2);
output.prodwell_GL.w_CO2_l_t(i,1)  = w_table(1,3);
output.prodwell_GL.w_CO2_g_t(i,1)  = w_table(1,4);
output.prodwell_GL.w_H2O_l_t(i,1)  = w_table(1,5);
output.prodwell_GL.w_H2O_v_t(i,1)  = w_table(1,6);

% Calculate segment properties
output.prodwell_GL.u(i,1)   = fCalc_u(output.prodwell_GL.m_gf(i,1), ...
                              output.prodwell_GL.rho(i,1), ...
                              output.prodwell_GL.D_i(i,1));          % velocity [m/s]
output.prodwell_GL.Re(i,1)  = fCalc_Re(output.prodwell_GL.D_i(i,1), ...
                              output.prodwell_GL.rho(i,1), output.prodwell_GL.u(i,1), ...
                              output.prodwell_GL.mu(i,1));           % Reynolds number [-]
output.prodwell_GL.f(i,1)   = fCalc_f(output.prodwell_GL.chi(i,1), ...
                              output.prodwell_GL.eps_pipe(i,1), output.prodwell_GL.D_i(i,1),...
                              output.prodwell_GL.Re(i,1));           % friction factor [-]
output.prodwell_GL.dQ(i,1) = fCalc_dQ(output.prodwell_GL.T(i,1), ...
                             output.prodwell_GL.T_g(i,1), output.prodwell_GL.D_i(i,1),...
                             output.prodwell_GL.dl(i,1), output.prodwell_GL.m_gf(i,1),...
                             input.general.gamma, input.general.t, ...
                             output.prodwell_GL.k_r(i,1), output.prodwell_GL.alfa_r(i,1));
```

```
    w_NaCl_l(2,1) = output.prodwell_GL.w_NaCl_l(n,1);
    w_NaCl_l(3,1) = output.prodwell_GL.w_NaCl_l(n+1,1);        % NaCl liquid mass fraction
    w_H2O_l(1,1) = 1-output.prodwell_GL.w_NaCl(m-1,1)-output.prodwell_GL.w_CO2(m-1,1);
    w_H2O_l(2,1) = output.prodwell_GL.w_H2O_l(n,1);
    w_H2O_l(3,1) = output.prodwell_GL.w_H2O_l(n+1,1);          % H2O liquid mass fraction
    if output.prodwell_GL.w_CO2(m-1,1) > 0
        w_CO2_g(1,1) = 1; w_CO2_g(2,1) = output.prodwell_GL.w_CO2_g(n,1) ;...
            w_CO2_g(3,1) = output.prodwell_GL.w_CO2_g(n+1,1);    % CO2 vapor mass fraction
    else
        w_CO2_g(1,1) = 0; w_CO2_g(2,1) = output.prodwell_GL.w_CO2_g(n,1); ...
            w_CO2_g(3,1) = output.prodwell_GL.w_CO2_g(n+1,1);    % CO2 vapor mass fraction
    end
else
    % Create interpolation tables for interpolation between degassing pressures.
    P(1,1) = output.prodwell_GL.P_degas(m,1); P(2,1) = output.prodwell_GL.P(n,1); ...
        P(3,1) = output.prodwell_GL.P(n+1,1);                                  % pressure [bar]
    T_int (1,1) = output.prodwell_GL.T(m-1,1); T_int (2,1) = output.prodwell_GL.T(n,1);
    h_int (1,1) = output.prodwell_GL.h(m-1,1); h_int (2,1) = output.prodwell_GL.h(n,1);
    chi(1,1) = output.prodwell_GL.chi(m-1,1); chi(2,1) = output.prodwell_GL.chi(n,1); ...
        chi(3,1) = output.prodwell_GL.chi(n+1,1);                              % quality [-]
    w_CO2_l(1,1) = output.prodwell_GL.w_CO2_l(m-1,1);
    w_CO2_l(2,1) = output.prodwell_GL.w_CO2_l(n,1); ...
    w_CO2_l(3,1) = output.prodwell_GL.w_CO2_l(n+1,1);          % CO2 liquid mass fraction
    w_NaCl_l(1,1) = output.prodwell_GL.w_NaCl(m-1,1);
    w_NaCl_l(2,1) = output.prodwell_GL.w_NaCl_l(n,1);
    w_NaCl_l(3,1) = output.prodwell_GL.w_NaCl_l(n+1,1);        % NaCl liquid mass fraction
    w_H2O_l(1,1) = 1-output.prodwell_GL.w_NaCl(m-1,1)-output.prodwell_GL.w_CO2(i,1); ...
    w_H2O_l(2,1) = output.prodwell_GL.w_H2O_l(n,1);
    w_H2O_l(3,1) = output.prodwell_GL.w_H2O_l(n+1,1);          % H2O liquid mass fraction
    if output.prodwell_GL.w_CO2(m-1,1) > 0
        w_CO2_g(1,1) = output.prodwell_GL.w_CO2_g(m-1,1);
        w_CO2_g(2,1) = output.prodwell_GL.w_CO2_g(n,1);
        w_CO2_g(3,1) = output.prodwell_GL.w_CO2_g(n+1,1);       % CO2 vapor mass fraction
    else
        w_CO2_g(1,1) = 0; w_CO2_g(2,1) = output.prodwell_GL.w_CO2_g(n,1);
        w_CO2_g(3,1) = output.prodwell_GL.w_CO2_g(n+1,1);       % CO2 vapor mass fraction
    end
end

formatSpec = 'Calculation production well with gas lift(iteration #%d).\nPlease wait...';
A1         = 1;
str        = sprintf(formatSpec,A1);
g          = waitbar(0,str);

% interpolate properties between P_degas Duan and last segment before P_degas Francke
for i = m:(n-1)
    waitbar((i-(m-1))/(n-1-(m-1)))
    output.prodwell_GL.P(i,1) = output.prodwell_GL.P(i-1,1) - ...
                                output.prodwell_GL.dP_hs(i-1,1)...
                                - output.prodwell_GL.dP_f(i-1,1);% pressure wellbore [bar]
    T_int1 = interp1(h_int, T_int, output.prodwell_GL.h(i,1));
    output.prodwell_GL.chi(i,1)  = interp1(P, chi, output.prodwell_GL.P(i,1),'spline');
                                                                        % quantity [-]
    output.prodwell_GL.w_CO2_l(i,1) = interp1(P, w_CO2_l, output.prodwell_GL.P(i,1),...
                                      'spline');                % CO2 liquid mass fraction
    output.prodwell_GL.w_NaCl_l(i,1) = interp1(P, w_NaCl_l, output.prodwell_GL.P(i,1),...
                                       'spline');               % NaCl liquid mass fraction
    output.prodwell_GL.w_H2O_l(i,1)  = interp1(P, w_H2O_l, output.prodwell_GL.P(i,1),...
                                       'spline');               % H2O liquid mass fraction
    output.prodwell_GL.w_CO2_g(i,1)  = interp1(P, w_CO2_g, output.prodwell_GL.P(i,1),...
                                       'spline');               % CO2 vapor mass fraction
    output.prodwell_GL.w_H2O_v(i,1)  = 1 - output.prodwell_GL.w_CO2_g(i,1);
                                                                % H2O vapor mass fraction
    % if calculated quality < 0, than spline interpolation failed, do linear interpolation
    if output.prodwell_GL.chi(i,1) < 0 || output.prodwell.chi(i,1) < ...
            output.prodwell.chi(i-1,1)
        output.prodwell_GL.chi(i,1)      = interp1(P, chi, output.prodwell_GL.P(i,1));
        output.prodwell_GL.w_CO2_l(i,1)  = interp1(P, w_CO2_l, output.prodwell_GL.P(i,1));
        output.prodwell_GL.w_NaCl_l(i,1) = interp1(P, w_NaCl_l, output.prodwell_GL.P(i,1));
        output.prodwell_GL.w_H2O_l(i,1)  = interp1(P, w_H2O_l, output.prodwell_GL.P(i,1));
        output.prodwell_GL.w_CO2_g(i,1)  = interp1(P, w_CO2_g, output.prodwell_GL.P(i,1));
        output.prodwell_GL.w_H2O_v(i,1)  = 1 - output.prodwell_GL.w_CO2_g(i,1);
    end
    if output.prodwell_GL.w_CO2_g(i,1) > 1
        output.prodwell_GL.w_CO2_g(i,1) = 1;
    end

    if i == input.prodwell_GL.segnr_GL
```



```
                                                % heat exchange with surroundings [J/kg]
    output.prodwell_GL.dE_pot(i,1) = fCalc_dE_pot(input.general.g, output.prodwell_GL.dz(i,1));
                                                    % potential energy [J/kg]
    output.prodwell_GL.dP_f(i,1) = fCalc_dP_f(output.prodwell_GL.D_i(i,1), ...
                                   output.prodwell_GL.f(i,1), output.prodwell_GL.rho(i,1), ...
                                   output.prodwell_GL.u(i,1), output.prodwell_GL.dl(i,1));
                                                    % frictional pressure change [bar]
    output.prodwell_GL.dP_hs(i,1) = fCalc_dP_hs(input.general.g, ...
                                   output.prodwell_GL.rho(i,1), output.prodwell_GL.dz(i,1));
                                                % hydrostatic pressure change [bar]

    % if i == size(input.prodwell_GL.tvd,1)
    %     output.prodwell_GL.P(i+1,1) = output.prodwell_GL.P(i,1) - output.prodwell_GL.dP_hs(i,1) - ...
    %                                   output.prodwell_GL.dP_f(i,1); % pressure pipe [bar]
    %     output.prodwell_GL.h(i+1,1) = output.prodwell_GL.h(i,1) - output.prodwell_GL.Q(i,1) - ...
    %                                   output.prodwell_GL.E_pot(i,1); % enthalpy [J/kg]
    % end

    %% Check if two segments have a significant gas mass fraction
    if l == 1
        if output.prodwell_GL.chi(i,1) > 0.0001 && output.prodwell_GL.chi(i-1,1) > 0.0001
            output.prodwell_GL.P_old = output.prodwell_GL.P(i-2,1);
            break % start interpolation from P_degas Duan and Sun (2003)
        end
    elseif l > 1 && i >= (j+1)
        if output.prodwell_GL.chi(i,1) > 0.0001 && output.prodwell_GL.chi(i-1,1) > 0.0001
            output.prodwell_GL.P_old = output.prodwell_GL.P(i-2,1);
            break % start interpolation from P_degas Duan and Sun (2003)
        end
    end

end

% Find P_degas from Duan and Sun (2003)
output.prodwell_GL.P_degas = interp3(data.m_NaCl_degas, data.T_degas, data.m_CO2_degas, ...
                             data.P_degas, output.prodwell_GL.m_NaCl(1:i,1), ...
                             output.prodwell_GL.T + 273.15,...
                             output.prodwell_GL.m_CO2(1:i,1));      % degassing pressure [bar]

% Check if degassing pressure Francke Model is above and Duan Model is below wellhead
if i == k
    if output.prodwell_GL.chi(k,1) < 0.0001 && output.prodwell_GL.P(k,1) < ...
            output.prodwell_GL.P_degas(k,1)
        [input, output, geofprops, i] = fCalc_prodwell_GL_virtual(input, output, data, k);
    end
    % Alternative to previous if-loop, for 2nd or higher iteration
    if output.prodwell_GL.chi(k,1) > 0.0001 && output.prodwell_GL.P(k,1) < ...
            output.prodwell_GL.P_degas(k,1) && output.prodwell_GL.chi(k-1) == 0
        [input, output, geofprops, i] = fCalc_prodwell_GL_virtual(input, output, data, k);
    end
end

% Find first segment number where degassing pressure Duan and Sun(2003) is above segment ...
% pressure from Francke Model.
m = find((output.prodwell_GL.P_degas - output.prodwell_GL.P) > 0,1);
if isempty(m) == 1 % if P_degas Duan is not above P_degas Francke
    m = i;
end
% Check if valve is located equal or above segment degassing pressure Duan
if input.prodwell_GL.segnr_GL >= m
    m      = input.prodwell_GL.segnr_GL;
    output = fCalc_Duan2valve(output,m);
end

n = i-1; % find(output.prodwell_GL.chi(n:i-1,1) > 0.001,1) + (n - 1);

if m < n % if P_degas Duan is above P_degas Francke
    if m > input.prodwell_GL.segnr_GL
        % Create interpolation tables for interpolation between degassing pressures.
        P(1,1) = output.prodwell_GL.P_degas(m,1); P(2,1) = output.prodwell_GL.P(n,1); ...
            P(3,1) = output.prodwell_GL.P(n+1,1);                       % pressure [bar]
        T_int (1,1) = output.prodwell_GL.T(m-1,1); T_int (2,1) = output.prodwell_GL.T(n,1);
        h_int (1,1) = output.prodwell_GL.h(m-1,1); h_int (2,1) = output.prodwell_GL.h(n,1);
        chi(1,1) = output.prodwell_GL.chi(m-1,1); chi(2,1) = output.prodwell_GL.chi(n,1); ...
            chi(3,1) = output.prodwell_GL.chi(n+1,1);                   % quality [-]
        w_CO2_l(1,1) = output.prodwell_GL.w_CO2_l(m-1,1);
        w_CO2_l(2,1) = output.prodwell_GL.w_CO2_l(n,1);
        w_CO2_l(3,1) = output.prodwell_GL.w_CO2_l(n+1,1);           % CO2 liquid mass fraction
        w_NaCl_l(1,1) = output.prodwell_GL.w_NaCl(m-1,1);
```

```
                                output.prodwell_GL.D_i(i,1),...
                                output.prodwell_GL.eps_pipe(i,1), ...
                                output.prodwell_GL.u_sg(i,1),...
                                output.prodwell_GL.u_sl(i,1), input.general.g, ...
                                output.prodwell_GL.chi(i,1), input.prodwell_GL.DF_model);
                                                                    % void fraction [-]
            output.prodwell_GL.eps_G(i,1) = eps_G;       % void fraction [-]
            output.prodwell_GL.FP(i,1)    = cellstr(FP); % flow pattern
            output.prodwell_GL.rho(i,1)   = output.prodwell_GL.rho_v(i,1)*...
                                output.prodwell_GL.eps_G(i,1) ...
                                + output.prodwell_GL.rho_l(i,1)*...
                                (1-output.prodwell_GL.eps_G(i,1));  % density [kg/m3]
            output.prodwell_GL.u_gu(i,1)  = u_gu; % drift-flux velocity, u_g relative to u_m
            output.prodwell_GL.C_0(i,1)   = C_0;  % distribution parameter
        end

        % Recalculate segment properties
        output.prodwell_GL.u(i,1)      = fCalc_u(output.prodwell_GL.m_gf(i,1), ...
                                output.prodwell_GL.rho(i,1),...
                                output.prodwell_GL.D_i(i,1));        % velocity [m/s]
        output.prodwell_GL.Re(i,1)     = fCalc_Re(output.prodwell_GL.D_i(i,1), ...
                                output.prodwell_GL.rho(i,1), ...
                                output.prodwell_GL.u(i,1), ...
                                output.prodwell_GL.mu(i,1));    % Reynolds number [-]
        output.prodwell_GL.f(i,1)      = fCalc_f(output.prodwell_GL.chi(i,1), ...
                                output.prodwell_GL.eps_pipe(i,1), ...
                                output.prodwell_GL.D_i(i,1), ...
                                output.prodwell_GL.Re(i,1));
                                                                % friction factor [-]
        output.prodwell_GL.dQ(i,1)     = fCalc_dQ(output.prodwell_GL.T(i,1), ...
                                output.prodwell_GL.T_g(i,1), ...
                                output.prodwell_GL.D_i(i,1), ...
                                output.prodwell_GL.dl(i,1), ...
                                output.prodwell_GL.m_gf(i,1), ...
                                input.general.gamma, input.general.t, ...
                                output.prodwell_GL.k_r(i,1), ...
                                output.prodwell_GL.alfa_r(i,1));
                                        % Heat exchange with surroundings [J/kg]
        output.prodwell_GL.dE_pot(i,1) = fCalc_dE_pot(input.general.g, ...
                                output.prodwell_GL.dz(i,1));
                                                % potential energy change [J/kg]
        output.prodwell_GL.dP_f(i,1)   = fCalc_dP_f(output.prodwell_GL.D_i(i,1), ...
                                output.prodwell_GL.f(i,1), ...
                                output.prodwell_GL.rho(i,1), ...
                                output.prodwell_GL.u(i,1), ...
                                output.prodwell_GL.dl(i,1));
                                                % frictional pressure change [bar]
        output.prodwell_GL.dP_hs(i,1)  = fCalc_dP_hs(input.general.g, ...
                                output.prodwell_GL.rho(i,1), ...
                                output.prodwell_GL.dz(i,1));
                                                % hydrostatic pressure change [bar]
    end
    close(g)
    i = n;
    j = n;
    k = max(output.prodwell_GL.segnr);
end

if m >= n % If degassing according to Duan and Sun (2003) starts later than Francke (2014)
    break
end
if abs(output.prodwell_GL.P(i-1,1) - output.prodwell_GL.P_old) < input.settings.dP_abs_pw
                                        % If calculation has iterated to user-defined error
    break
end

end


%% Proceed with segment of flash horizon of Francke (2014)
if max(input.prodwell_GL.segment) >= max(output.prodwell_GL.segnr)
    for i = n:max(input.prodwell_GL.segment) + 1

    if i == max(input.prodwell.segment) + 1
        output.prodwell_GL.segnr(i,1)    = output.prodwell_GL.segnr(i-1,1)+1;  % segment nr.
        output.prodwell_GL.D_i(i,1)      = input.prodwell.D_i(i-1,1);      % diameter wellbore [m]
        output.prodwell_GL.dl(i,1)       = input.prodwell.dl(i-1,1);       % length [m]
        output.prodwell_GL.dz(i,1)       = input.prodwell.dz(i-1,1);       % dz [m]
        output.prodwell_GL.tvd(i,1)      = 0;                              % tvd [m]
```

```
%                   output.prodwell_GL.h(i,1) = output.prodwell_GL.h(i-1,1) - output.prodwell_GL.dQ(i-1,1) -
...
%                                       output.prodwell_GL.dE_pot(i-1,1);             % enthalpy [J/kg]
%                   %output.prodwell_GL.h(i,1) = (output.prodwell_GL.h(i,1) * output.prodwell_GL.m_gf(i-1,1)
+ ...
%                   %                    output.prodwell_GL.h_GL *
output.prodwell_GL.m_GL)/output.prodwell_GL.m_gf(i,1);
%                   output.prodwell_GL.h(i,1) = (output.prodwell_GL.h_GL_l * output.prodwell_GL.m_gf(i,1) *
(1 - output.prodwell_GL.chi(i,1)) + ...
%                                       output.prodwell_GL.h_GL * output.prodwell_GL.m_gf(i,1) *
output.prodwell_GL.chi(i,1))/output.prodwell_GL.m_gf(i,1);

                    sheet2 = get(Sheets, 'Item', 2);
                    invoke(sheet2, 'Activate');
                    sheet  = Excel.Activesheet;
                    sheet.set('Range', 'C2', output.prodwell.P(i,1));
                    sheet.set('Range', 'C3', output.prodwell.T(i,1));
                    sheet.set('Range', 'C5', output.prodwell_GL.w_CO2_g(i,1));
                    range  = sheet.get('Range', 'G5');
                    range.Value;
                    output.prodwell_GL.h_GL = range.Value;

                    sheet2 = get(Sheets, 'Item', 3);
                    invoke(sheet2, 'Activate');
                    sheet  = Excel.Activesheet;
                    sheet.set('Range', 'C3', output.prodwell.P(i,1));
                    sheet.set('Range', 'C4', output.prodwell.T(i,1));
                    sheet.set('Range', 'C6', output.prodwell_GL.w_NaCl_l(i,1));
                    range  = sheet.get('Range', 'G6');
                    range.Value;
                    output.prodwell_GL.h_GL_l = range.Value;

                    output.prodwell_GL.h(i,1) = (output.prodwell_GL.h_GL_l * ...
                                                 output.prodwell_GL.m_gf(i,1) * (1 - ...
                                                 output.prodwell_GL.chi(i,1)) + ...
                                                 output.prodwell_GL.h_GL * ...
                                                 output.prodwell_GL.m_gf(i,1) * ...
                                                 output.prodwell_GL.chi(i,1))/...
                                                 output.prodwell_GL.m_gf(i,1);
                else
                    output.prodwell_GL.h(i,1) = output.prodwell_GL.h(i-1,1) - ...
                                                 output.prodwell_GL.dQ(i-1,1) - ...
                                                 output.prodwell_GL.dE_pot(i-1,1);    % enthalpy [J/kg]
                end

                % Invoke fCalc_geofprops3 for single liquid and single vapor properties calculation...
                % and calculate total properties
                [T_new, rho_m, c_p_m, mu_m, eps_G, rho_v, rho_l, mu_v, mu_l] = fCalc_geofprops3 ...
                    (output.prodwell_GL.P(i,1), output.prodwell_GL.T(i-1,1), ...
                    output.prodwell_GL.w_CO2_l(i,1), output.prodwell_GL.w_NaCl_l(i,1), ...
                    output.prodwell_GL.w_CO2_g(i,1), output.prodwell_GL.h(i,1), ...
                    output.prodwell_GL.chi(i,1), output.prodwell_GL.h(i-1,1),...
                    input, output.prodwell.T(i-2,1),l,T_int1);          % geothermal fluid properties
                output.prodwell_GL.T(i,1)     = T_new;                  % temperature [C]
                output.prodwell_GL.rho(i,1)   = rho_m;                  % density [kg/m3]
                output.prodwell_GL.c_p(i,1)   = c_p_m;                  % specific heat capacity [J/kg/K]
                output.prodwell_GL.mu(i,1)    = mu_m;                   % viscosity [Pa*s]
                output.prodwell_GL.eps_G(i,1) = eps_G;                  % void fraction [-]

                % Drift flux model
                if input.prodwell_GL.DF_model > 1       % DF_model = 1 --> homogeneous
                    output.prodwell_GL.rho_l(i,1) = rho_l;  % density liquid phase [kg/m3]
                    output.prodwell_GL.rho_v(i,1) = rho_v;  % density gas phase [kg/m3]
                    output.prodwell_GL.mu_l(i,1)  = mu_l;   % viscosity liquid phase [Pa*s]
                    output.prodwell_GL.mu_v(i,1)  = mu_v;   % viscosity gas phase [Pa*s]
                    p = find(output.prodwell_GL.chi > 0,1); % segment number with flash horizon
                    output.prodwell_GL.l_E(i,1)  = output.prodwell_GL.l(i,1) - ...
                                                   output.prodwell_GL.l(p,1);
                                                     % length from entrance or flash horizon [m]
                    output.prodwell_GL.u_sg(i,1) = ((output.prodwell_GL.chi(i,1)* ...
                                                   output.prodwell_GL.m_gf(i,1))/...
                                                   rho_v)/(pi*(output.prodwell_GL.D_i(i,1)/2)^2);
                                                     % superficial gas velocity [m/s]
                    output.prodwell_GL.u_sl(i,1) = (((1-output.prodwell_GL.chi(i,1)) * ...
                                                   output.prodwell_GL.m_gf(i,1))/rho_l)/(pi*...
                                                   (output.prodwell_GL.D_i(i,1)/2)^2);
                                                     % superficial liquid velocity [m/s]
                    [eps_G,FP,u_gu,C_0] = fCalc_eps_G(output.prodwell_GL.T(i,1), rho_l, rho_v, ...
                                          mu_l, mu_v, output.prodwell_GL.l_E(i,1),...
```

```
    output.prodwell_GL.grad_T_g(i,1) = input.prodwell.grad_T_g(i-1,1); % temperature grad [m]
    output.prodwell_GL.eps_pipe(i,1) = input.prodwell.eps_pipe(i-1,1); % abs pipe roughness[m]
    output.prodwell_GL.k_r(i,1)      = input.prodwell.k_r(i-1,1);    % rock therm cond.[W/m/K]
    output.prodwell_GL.alfa_r(i,1)   = input.prodwell.alfa_r(i-1,1);   % rock therm diff[m2/s]
    output.prodwell_GL.l(i,1)    = output.prodwell_GL.l(i-1,1) + output.prodwell_GL.dl(i-1,1);
    output.prodwell_GL.T_g(i,1)  = output.prodwell_GL.T_g(i-1,1);
    output.prodwell_GL.m_gf(i,1) = output.prodwell_GL.m_gf(i-1,1);
    output.prodwell_GL.w_NaCl(i,1) = output.prodwell_GL.w_NaCl(i-1,1);
    output.prodwell_GL.w_CO2(i,1)  = output.prodwell_GL.w_CO2(i-1,1);
    output.prodwell_GL.w_H2O(i,1)  = output.prodwell_GL.w_H2O(i-1,1);
    output.prodwell_GL.m_NaCl(i,1) = output.prodwell_GL.m_NaCl(i-1,1);
    output.prodwell_GL.m_CO2(i,1)  = output.prodwell_GL.m_CO2(i-1,1);
end

    waitbar(i/max(input.prodwell_GL.segment))
    output.prodwell_GL.P(i,1) = output.prodwell_GL.P(i-1,1) - ...
                                output.prodwell_GL.dP_hs(i-1,1) - ...
                                output.prodwell_GL.dP_f(i-1,1);   % pressure wellbore [bar]
    output.prodwell_GL.h(i,1) = output.prodwell_GL.h(i-1,1) - ...
                                output.prodwell_GL.dQ(i-1,1) - ...
                                output.prodwell_GL.dE_pot(i-1,1); % enthalpy [J/kg]
    if output.prodwell_GL.P(i,1) < input.general.P_atm        % minimum pressure of wellbore
        disp('ERROR: Pressure loss in wellbore too high. ACTION: Decrease mass flow')
        close(h)
        msgbox('Pressure loss in wellbore too high. ACTION: Decrease mass flow', 'Error','error');
        stat = status.FAILURE; return;
    end

    if i == input.prodwell_GL.segnr_GL
        sheet2 = get(Sheets, 'Item', 1);
        invoke(sheet2, 'Activate');
        sheet  = Excel.Activesheet;
        sheet.set('Range', 'C3', output.prodwell.P(i,1));
        sheet.set('Range', 'C4', output.prodwell.T(i,1));
        sheet.set('Range', 'C8', output.prodwell_GL.w_NaCl(i,1));
        sheet.set('Range', 'C11', output.prodwell_GL.w_CO2(i,1));
        range  = sheet.get('Range', 'D8:D13');
        range2 = sheet.get('Range', 'I9');
        range.Value;
        range2.Value;
        data_FM  = range.Value;
        data1_FM = range2.Value;
        output.prodwell_GL.m_NaCl(i:max(input.prodwell_GL.segment),1)= cell2mat(data_FM(1,1));
        output.prodwell_GL.m_CO2(i:max(input.prodwell_GL.segment),1) = cell2mat(data_FM(4,1));
        output.prodwell_GL.h(i,1) = data1_FM;
    end

    [geofprops, T_new, w_table] = fCalc_geofprops2(output.prodwell_GL.P(i,1), ...
        output.prodwell_GL.T(i-1,1), output.prodwell_GL.w_NaCl(i,1), ...
        output.prodwell_GL.w_CO2(i,1), data.H2O_sat, output.prodwell_GL.h(i,1), ...
        output, output.prodwell_GL.h(i-1,1), input, data);
    output.prodwell_GL.T(i,1)      = T_new;            % temperature [C]
    output.prodwell_GL.chi(i,1)    = geofprops(1,2);   % gas mass fraction [-]
    output.prodwell_GL.v_spec(i,1) = 1/geofprops(1,4); % specific volume [m3/kg]
    output.prodwell_GL.rho(i,1)    = geofprops(1,4);   % density [kg/m3]
    output.prodwell_GL.c_p(i,1)    = geofprops(1,5);   % specific heat capacity [J/kg/K]
    output.prodwell_GL.mu(i,1)     = geofprops(1,7);   % viscosity [Pa*s]
    output.prodwell_GL.eps_G(i,1)  = geofprops(1,3);   % void fraction [-]

    % Drift-flux model
    if input.prodwell_GL.DF_model > 1                  % DF_model = 1 --> homogeneous
        output.prodwell_GL.rho_l(i,1) = geofprops(1,15); % density liquid phase [kg/m3]
        output.prodwell_GL.rho_v(i,1) = geofprops(1,23); % density gas phase [kg/m3]
        output.prodwell_GL.mu_l(i,1)  = geofprops(1,18); % viscosity liquid phase [Pa*s]
        output.prodwell_GL.mu_v(i,1)  = geofprops(1,26); % viscosity gas phase [Pa*s]
        p = find(output.prodwell_GL.chi > 0,1); % segment number with flash horizon
        output.prodwell_GL.l_E(i,1)   = output.prodwell_GL.l(i,1) - output.prodwell_GL.l(p,1);
        % length from entrance or flash horizon [m]
        output.prodwell_GL.u_sg(i,1)  = ((output.prodwell_GL.chi(i,1) * ...
                                        output.prodwell_GL.m_gf(i,1))/...
                                        geofprops(1,23))/(pi*(output.prodwell_GL.D_i(i,1)/2)^2);
        % superficial gas velocity [m/s]
        output.prodwell_GL.u_sl(i,1)  = (((1-output.prodwell_GL.chi(i,1)) * ...
                                        output.prodwell_GL.m_gf(i,1))/...
                                        geofprops(1,15))/(pi*(output.prodwell_GL.D_i(i,1)/2)^2);
        % superficial liquid velocity [m/s]
        [eps_G,FP,u_gu,C_0] = fCalc_eps_G(output.prodwell_GL.T(i,1), geofprops(1,15), ...
            geofprops(1,23), geofprops(1,18), geofprops(1,26), ...
            output.prodwell_GL.l_E(i,1), output.prodwell_GL.D_i(i,1), ...
            output.prodwell_GL.eps_pipe(i,1), output.prodwell_GL.u_sg(i,1), ...
            output.prodwell_GL.u_sl(i,1), input.general.g, ...
            output.prodwell_GL.chi(i,1), input.prodwell_GL.DF_model);
        % void fraction [-]
        output.prodwell_GL.eps_G(i,1) = eps_G;       % void fraction [-]
        output.prodwell_GL.FP(i,1)    = cellstr(FP); % flow pattern
        output.prodwell_GL.rho(i,1)   = output.prodwell_GL.rho_v(i,1) * ...\
                                        output.prodwell_GL.eps_G(i,1) + ...
                                        output.prodwell_GL.rho_l(i,1) * ...
                                        (1-output.prodwell_GL.eps_G(i,1)); % density [kg/m3]
        output.prodwell_GL.u_gu(i,1)  = u_gu; % drift-flux velocity, u_g relative to u_m [m/s]
        output.prodwell_GL.C_0(i,1)   = C_0;  % distribution parameter
    end
    % Output geofluid composition
    output.prodwell_GL.w_NaCl_l(i,1) = w_table(3,2);
    output.prodwell_GL.w_CO2_l(i,1)  = w_table(3,3);
    output.prodwell_GL.w_CO2_g(i,1)  = w_table(3,4);
    output.prodwell_GL.w_H2O_l(i,1)  = w_table(3,5);
    output.prodwell_GL.w_H2O_v(i,1)  = w_table(3,6);

    output.prodwell_GL.u(i,1)      = fCalc_u(output.prodwell_GL.m_gf(i,1), ...
                                     output.prodwell_GL.rho(i,1), ...
                                     output.prodwell_GL.D_i(i,1)); % velocity [m/s]
    output.prodwell_GL.Re(i,1)     = fCalc_Re(output.prodwell_GL.D_i(i,1), ...
                                     output.prodwell_GL.rho(i,1),...
                                     output.prodwell_GL.u(i,1), output.prodwell_GL.mu(i,1));
                                                          % Reynolds number [-]
    output.prodwell_GL.f(i,1)      = fCalc_f(output.prodwell_GL.chi(i,1), ...
        output.prodwell_GL.eps_pipe(i,1), output.prodwell_GL.D_i(i,1), ...
        output.prodwell_GL.Re(i,1));  % friction factor [-]
    output.prodwell_GL.dQ(i,1)     = fCalc_dQ(output.prodwell_GL.T(i,1), output.prodwell_GL.T_g(i,1),
...
        output.prodwell_GL.D_i (i,1), output.prodwell_GL.dl(i,1), ...
        output.prodwell_GL.m_gf(i,1), input.general.gamma, input.general.t, ...
        output.prodwell_GL.k_r(i,1), output.prodwell_GL.alfa_r(i,1));
    % Heat exchange with surroundings [J/kg]
    output.prodwell_GL.dE_pot(i,1) = fCalc_dE_pot(input.general.g, output.prodwell_GL.dz(i,1));
    % potential energy change [J/kg]
    output.prodwell_GL.dP_f(i,1)   = fCalc_dP_f(output.prodwell_GL.D_i(i,1), ...
                                     output.prodwell_GL.f(i,1),...
                                     output.prodwell_GL.rho(i,1),output.prodwell_GL.u(i,1),...
                                     output.prodwell_GL.dl(i,1));
                                                          % frictional pressure change [bar]
    output.prodwell_GL.dP_hs(i,1)  = fCalc_dP_hs(input.general.g, ...
                                     output.prodwell_GL.rho(i,1), ...
                                     output.prodwell_GL.dz(i,1));  % hydros. P change [bar]
        end
    end
    end
    close(h)

    %output                        = fChange_prodwell_GL(input, output);
    output.prodwell_GL.P(:,a)     = output.prodwell_GL.P;
    output.prodwell_GL.T(:,a)     = output.prodwell_GL.T;
    output.prodwell_GL.chi(:,a)   = output.prodwell_GL.chi;
    output.prodwell_GL.h(:,a)     = output.prodwell_GL.h;
    output.prodwell_GL.eps_G(:,a) = output.prodwell_GL.eps_G;

%% Calculation of the gas lift annulus properties

    load P_CO2; load T_CO2; load cp_CO2; load h_CO2; load k_CO2; load mu_CO2; load rho_CO2; load s_CO2;
    load H2O_sat_props;

    output.prodwell_GL.T_GL(input.prodwell_GL.segnr_GL,1) = ...
        output.prodwell_GL.T(input.prodwell_GL.segnr_GL,1);
    output.prodwell_GL.P_GL(input.prodwell_GL.segnr_GL,1) = ...
        output.prodwell_GL.P(input.prodwell_GL.segnr_GL,1);
    output.prodwell_GL.h_GL(input.prodwell_GL.segnr_GL,1) = ...
        interp2(P_CO2,T_CO2,h_CO2,output.prodwell_GL.P_GL(input.prodwell_GL.segnr_GL,1),...
        output.prodwell_GL.T_GL(input.prodwell_GL.segnr_GL,1));
    options = optimset('Display','off');

    for j = 1:1
    for i = input.prodwell_GL.segnr_GL:max(input.prodwell_GL.segment)

        output.prodwell_GL.dQ_g(i,j) = 102;
        output.prodwell_GL.dQ_gf_GL(i,j) = 1;

        output.prodwell_GL.T_GL(i+1,j) = output.prodwell_GL.T_GL(i,j);
```

```
output.prodwell_GL.rho(i+1,1)    = geofprops(1,4);   % density [kg/m3]
output.prodwell_GL.c_p(i+1,1)    = geofprops(1,5);   % specific heat capacity [J/kg/K]
output.prodwell_GL.mu(i+1,1)     = geofprops(1,7);   % viscosity [Pa*s]
output.prodwell_GL.eps_G(i+1,1)  = geofprops(1,3);   % void fraction [-]

% Drift-flux model
if input.prodwell_GL.DF_model > 1                    % DF_model = 1 --> homogeneous
    output.prodwell_GL.rho_l(i+1,1) = geofprops(1,15); % density liquid phase [kg/m3]
    output.prodwell_GL.rho_v(i+1,1) = geofprops(1,23); % density gas phase [kg/m3]
    output.prodwell_GL.mu_l(i+1,1)  = geofprops(1,18); % viscosity liquid phase [Pa*s]
    output.prodwell_GL.mu_v(i+1,1)  = geofprops(1,26); % viscosity gas phase [Pa*s]
    p = find(output.prodwell_GL.chi > 0,1); % segment number with flash horizon
    output.prodwell_GL.l_E(i+1,1) = output.prodwell_GL.l(i+1,1)-output.prodwell_GL.l(p,1);
    % length from entrance or flash horizon [m]
    output.prodwell_GL.u_sg(i+1,1)  = ((output.prodwell_GL.chi(i+1,1) * ...
                                       output.prodwell_GL.m_gf(i+1,1))/...
                                       geofprops(1,23))/(pi*...
                                       (output.prodwell_GL.D_i(i+1,1)/2)^2);
                                                % superficial gas velocity [m/s]
    output.prodwell_GL.u_sl(i+1,1)  = (((1-output.prodwell_GL.chi(i+1,1)) * ...
                                       output.prodwell_GL.m_gf(i+1,1))/...
                                       geofprops(1,15))/(pi*...
                                       (output.prodwell_GL.D_i(i+1,1)/2)^2);
                                                % superficial liquid velocity [m/s]
    [eps_G,FP,u_gu,C_0] = fCalc_eps_G(output.prodwell_GL.T(i+1,1), geofprops(1,15), ...
        geofprops(1,23), geofprops(1,18), geofprops(1,26), ...
        output.prodwell_GL.l_E(i+1,1), output.prodwell_GL.D_i(i+1,1), ...
        output.prodwell_GL.eps_pipe(i+1,1), output.prodwell_GL.u_sg(i+1,1), ...
        output.prodwell_GL.u_sl(i+1,1), input.general.g, ...
        output.prodwell_GL.chi(i+1,1), input.prodwell_GL.DF_model);
    % void fraction [-]
    output.prodwell_GL.eps_G(i+1,1) = eps_G;       % void fraction [-]
    output.prodwell_GL.FP(i+1,1)    = cellstr(FP); % flow pattern
    output.prodwell_GL.rho(i+1,1)   = output.prodwell_GL.rho_v(i+1,1) * ...
                                      output.prodwell_GL.eps_G(i+1,1) + ...
                                      output.prodwell_GL.rho_l(i+1,1) * ...
                                      (1-output.prodwell_GL.eps_G(i+1,1));
                                                % density [kg/m3]
    output.prodwell_GL.u_gu(i+1,1)  = u_gu; % drift-flux velocity, u_g rel. to u_m [m/s]
    output.prodwell_GL.C_0(i+1,1)   = C_0;  % distribution parameter
end
% Output geofluid composition
output.prodwell_GL.w_NaCl_l(i+1,1) = w_table(3,2);
output.prodwell_GL.w_CO2_l(i+1,1)  = w_table(3,3);
output.prodwell_GL.w_CO2_g(i+1,1)  = w_table(3,4);
output.prodwell_GL.w_H2O_l(i+1,1)  = w_table(3,5);
output.prodwell_GL.w_H2O_v(i+1,1)  = w_table(3,6);

output.prodwell_GL.u(i+1,1)      = fCalc_u(output.prodwell_GL.m_gf(i+1,1), ...
                                   output.prodwell_GL.rho(i+1,1), ...
                                   output.prodwell_GL.D_i(i+1,1)); % velocity [m/s]
output.prodwell_GL.Re(i+1,1)     = fCalc_Re(output.prodwell_GL.D_i(i+1,1), ...
                                   output.prodwell_GL.rho(i+1,1),...
                                   output.prodwell_GL.u(i+1,1), ...
                                   output.prodwell_GL.mu(i+1,1));
                                                % Reynolds number [-]
output.prodwell_GL.f(i+1,1) = fCalc_f(output.prodwell_GL.chi(i+1,1), ...
                              output.prodwell_GL.eps_pipe(i+1,1), ...
                              output.prodwell_GL.D_i(i+1,1), ...
                              output.prodwell_GL.Re(i+1,1)); % friction factor [-]
output.prodwell_GL.dQ(i+1,1) = fCalc_dQ(output.prodwell_GL.T(i+1,1), ...
                               output.prodwell_GL.T_g(i+1,1), ...
                               output.prodwell_GL.D_i (i+1,1), ...
                               output.prodwell_GL.dl(i+1,1), ...
                               output.prodwell_GL.m_gf(i+1,1), input.general.gamma, ...
                               input.general.t, output.prodwell_GL.k_r(i+1,1), ...
                               output.prodwell_GL.alfa_r(i+1,1));
                                                % Heat exchange with surroundings [J/kg]
output.prodwell_GL.dE_pot(i+1,1) = fCalc_dE_pot(input.general.g, ...
                                   output.prodwell_GL.dz(i+1,1));
                                                % potential energy change [J/kg]
output.prodwell_GL.dP_f(i+1,1)   = fCalc_dP_f(output.prodwell_GL.D_i(i+1,1), ...
                                   output.prodwell_GL.f(i+1,1),...
                                   output.prodwell_GL.rho(i+1,1), ...
                                   output.prodwell_GL.u(i+1,1), ...
                                   output.prodwell_GL.dl(i+1,1));
                                                % frictional pressure change [bar]
output.prodwell_GL.dP_hs(i+1,1)  = fCalc_dP_hs(input.general.g, ...
                                   output.prodwell_GL.rho(i+1,1), ...
```



```
while abs(output.prodwell_GL.dQ_g(i,j) - output.prodwell_GL.dQ_gf_GL(i,j)) >= 100
if output.prodwell_GL.dQ_g(i,j) > output.prodwell_GL.dQ_gf_GL(i,j)
    output.prodwell_GL.T_GL(i+1,j) = output.prodwell_GL.T_GL(i+1,j) - 0.01;
else
    output.prodwell_GL.T_GL(i+1,j) = output.prodwell_GL.T_GL(i+1,j) + 0.01;
end
output.prodwell_GL.dQ_g(i,j) = fCalc_dQ(output.prodwell_GL.T_GL(i+1,j), ...
                               output.prodwell_GL.T_g(i,1), output.prodwell_GL.D_i(i,1) + ...
                               0.05, output.prodwell_GL.dl(i,1), output.prodwell_GL.m_GL, ...
                               input.general.gamma, input.general.t, ...
                               output.prodwell_GL.k_r(i,1), output.prodwell_GL.alfa_r(i,1));
                                                % Heat exchange with surroundings [J/kg]
output.prodwell_GL.dE_pot_GL(i,j) = fCalc_dE_pot(input.general.g, output.prodwell_GL.dz(i,1));
                                                % potential energy change [J/kg]
output.prodwell_GL.dE_pot(i,1) = fCalc_dE_pot(input.general.g, output.prodwell_GL.dz(i,1));
%output.prodwell_GL.dQ_gf(i,j)

[output] = fCalc_dQgf(output,input,i,j);
output.prodwell_GL.dP_hs_GL(i,j) = fCalc_dP_hs(input.general.g, ...
                                   output.prodwell_GL.rho_GL(i,j), ...
                                   output.prodwell_GL.dz(i,1));  % hydrostatic p change [bar]
output.prodwell_GL.c_p_GL(i,j) = interp2(P_CO2,T_CO2,cp_CO2,output.prodwell_GL.P_GL(i,j),...
                                 output.prodwell_GL.T_GL(i,j));
%output.prodwell_GL.T_GL(i+1,j) = output.prodwell_GL.T_GL(i,j) + (output.prodwell_GL.dQ_gf_GL(i,j) -
output.prodwell_GL.dQ_g(i,j) - output.prodwell_GL.dE_pot_GL(i,j))/output.prodwell_GL.c_p_GL(i,j);
output.prodwell_GL.u_GL(i,j)      = fCalc_u(output.prodwell_GL.m_GL, ...
                                    output.prodwell_GL.rho_GL(i,j), 0.05);    % velocity [m/s]
output.prodwell_GL.f(i,j)      = fCalc_f(0, output.prodwell_GL.eps_pipe(1,1), ...
                                 0.05, output.prodwell_GL.Re_GL(i,j));   % friction factor [-]
output.prodwell_GL.dP_f_GL(i,j)   = fCalc_dP_f(0.05, output.prodwell_GL.f(i,j), ...
                                    output.prodwell_GL.rho_GL(i,j), ...
                                    output.prodwell_GL.u_GL(i,j), output.prodwell_GL.dl(i,j));
                                                % frictional pressure change [J/kg]

if i >= 4
     output.prodwell_GL.dP_k_GL(i,j) = output.prodwell_GL.rho_GL(i,j) *...
         (output.prodwell_GL.u_GL(i,j)^2-output.prodwell_GL.u_GL(i-1,j)^2);
end
output.prodwell_GL.P_GL(i+1,j) = output.prodwell_GL.P_GL(i,j) - ...
    output.prodwell_GL.dP_hs_GL(i,j) + output.prodwell_GL.dP_f_GL(i,j);

end


    if i == input.prodwell_GL.segnr_GL
        Excel  = actxGetRunningServer('Excel.Application');
        Sheets = Excel.ActiveWorkBook.Sheets;
        sheet2 = get(Sheets, 'Item', 1);
        invoke(sheet2, 'Activate');
        sheet  = Excel.Activesheet;
        sheet.set('Range', 'C3', output.prodwell.P(i,1));
        sheet.set('Range', 'C4', output.prodwell.T(i,1));
        sheet.set('Range', 'C8', output.prodwell_GL.w_NaCl(i,1));
        sheet.set('Range', 'C11', output.prodwell_GL.w_CO2(i,1));
        range  = sheet.get('Range', 'D8:D13');
        range2 = sheet.get('Range', 'I9');
        range.Value;
        range2.Value;
        data_FM  = range.Value;
        data1_FM = range2.Value;
        output.prodwell_GL.m_NaCl(i:max(input.prodwell_GL.segment),1) = cell2mat(data_FM(1,1));
        output.prodwell_GL.m_CO2(i:max(input.prodwell_GL.segment),1)  = cell2mat(data_FM(4,1));
        output.prodwell_GL.h(i,1) = data1_FM;
    end

    waitbar(i/max(input.prodwell_GL.segment))
    output.prodwell_GL.P(i+1,1) = output.prodwell_GL.P(i,1) - output.prodwell_GL.dP_hs(i,1)...
                                  - output.prodwell_GL.dP_f(i,1);   % pressure wellbore [bar]
    output.prodwell_GL.h(i+1,1) = output.prodwell_GL.h(i,1) + ...
                                  output.prodwell_GL.dQ_gf_gf(i,1) - ...
                                  output.prodwell_GL.dE_pot(i,1); % enthalpy [J/kg]
     [geofprops, T_new, w_table] = fCalc_geofprops2(output.prodwell_GL.P(i+1,1), ...
        output.prodwell_GL.T(i,1), output.prodwell_GL.w_NaCl(i+1,1), ...
        output.prodwell_GL.w_CO2(i+1,1), data.H2O_sat, output.prodwell_GL.h(i+1,1), ...
        output, output.prodwell_GL.h(i,1), input, data);
    output.prodwell_GL.T(i+1,1)      = T_new;           % temperature [C]
    output.prodwell_GL.chi(i+1,1)    = geofprops(1,2);  % gas mass fraction [-]
    output.prodwell_GL.v_spec(i+1,1) = 1/geofprops(1,4); % specific volume [m3/kg]
```

```
                                                output.prodwell_GL.dz(i+1,1));
                                                    % hydrostatic pressure change [bar]

            if output.prodwell_GL.P(i+1,1) < input.general.P_atm        % minimum pressure of wellbore
                disp('ERROR: Pressure loss in wellbore too high. ACTION: Decrease mass flow')
                close(h)
                msgbox('Pressure loss in wellbore too high. ACTION: Decrease mass flow', 'Error','error');
                stat = status.FAILURE; return;
            end
        end
    end

    %% Compressor calculation from wellhead conditions (pressure, temperature)
    output.prodwell_GL.s_CO2 = interp2(P_CO2,T_CO2,s_CO2,output.prodwell_GL.P(end,1),...
                               output.prodwell_GL.T(end,1));

    % iterative procedure compressor
    x0    = [output.prodwell_GL.T(end,1)];                                  % iteration variable
    y0    = [output.prodwell_GL.P_GL(end,1), output.prodwell_GL.s_CO2,1];   % iteration constants
    f     = @(x0)fCalc_T_s_com(x0,y0);
    output.prodwell_GL.T_CO2_2s = fsolve(f,x0,options);
    output.prodwell_GL.h_CO2_2s = interp2(P_CO2,T_CO2,h_CO2,output.prodwell_GL.P_GL(end,1),...
                                  output.prodwell_GL.T_CO2_2s);
    output.prodwell_GL.h_CO2_1  = interp2(P_CO2,T_CO2,h_CO2,output.prodwell_GL.P(end,1),...
                                  output.prodwell_GL.T(end,1));
    output.prodwell_GL.h_CO2_2  = ((output.prodwell_GL.h_CO2_2s - output.prodwell_GL.h_CO2_1)/...
                                  input.B.eta_com) + output.prodwell_GL.h_CO2_1;
    x0    = [output.prodwell_GL.T_CO2_2s];                                  % iteration variable
    y0    = [output.prodwell_GL.P_GL(end,1), output.prodwell_GL.h_CO2_2,2];  % iteration constants
    f     = @(x0)fCalc_T_s_com(x0,y0);
    output.prodwell_GL.T_CO2_2 = fsolve(f,x0,options);

    %% Compressor calculation from ambient conditions (pressure, temperature)
    output.prodwell_GL.s_CO2_atm = interp2(P_CO2,T_CO2,s_CO2,input.general.P_atm,...
                                   input.general.T_surf_w);
    output.prodwell_GL.h_CO2_1_atm = interp2(P_CO2,T_CO2,h_CO2,input.general.P_atm,...
                                     input.general.T_surf_w);

    % iterative procedure compressor
    x0    = [output.prodwell_GL.T(end,1)];                                   % iteration variable
    y0    = [output.prodwell_GL.P_GL(end,1), output.prodwell_GL.s_CO2_atm,1]; % iteration constants
    f     = @(x0)fCalc_T_s_com(x0,y0);
    output.prodwell_GL.T_CO2_2s_atm = fsolve(f,x0,options);
    output.prodwell_GL.h_CO2_2s_atm = interp2(P_CO2,T_CO2,h_CO2,output.prodwell_GL.P_GL(end,1),...
                                      output.prodwell_GL.T_CO2_2s_atm);
    output.prodwell_GL.h_CO2_2_atm  = ((output.prodwell_GL.h_CO2_2s_atm - ...
                                      output.prodwell_GL.h_CO2_1_atm)/ input.B.eta_com) + ...
                                      output.prodwell_GL.h_CO2_1_atm;
    x0    = [output.prodwell_GL.T_CO2_2s_atm];                                % iteration variable
    y0    = [output.prodwell_GL.P_GL(end,1), output.prodwell_GL.h_CO2_2_atm,2];  % iteration constants
    f     = @(x0)fCalc_T_s_com(x0,y0);
    output.prodwell_GL.T_CO2_2_atm = fsolve(f,x0,options);
end
```

```
%% fCalc_SF

% Simulation of single-flash power plant
% Frank Niewold
% Released version 1.0, February 2017

function [input, output, stat] = fCalc_SF(input, output, status, data, algorithm)
    % numbers in output parameters correspond to single-flash power plant figure in report

    % Succesfull simulation
    stat = status.SUCCES;
    %input.SF.P_out_t = 0.1;
    %input.SF.T_out_cd = 35;
    if algorithm == 1 % first part of single-flash power plant calculation until injection pump
        % dummy parameters and constants input settings
        out             = zeros(1,1);
        Newoutput       = zeros(1,1);
        error_eta_t_SF  = input.settings.error_eta_t_SF; % accepted error iteration
        T0_12           = input.settings.T0_12;          % initial temperature @ state 12
        dP              = input.settings.dP_step_SF;     % stepsize pressure [bar]
        options         = optimset('Display','off');
        output.SF.T_11  = input.SF.T_out_cd;

        compressor = 0;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Begin check for gas mass fraction AND determining initial flash properties  %

if output.prodwell.chi(end) == 0 % geothermal fluid is in liquid phase at top of production
    % well
    output.SF.T_2     = output.prodwell.T(end);   % [C]
    output.SF.P_2     = output.prodwell.P(end);   % [bar]
    output.SF.h_mix_2 = output.prodwell.h(end);   % [J/kg]
    output.SF.chi_2   = output.prodwell.chi(end); % [-]

    while output.SF.chi_2 < input.settings.chi_2_min % Do until significant quality is present
        output.SF.P_2                = output.SF.P_2 - dP; % pressure [bar]
        [geofprops, T_new, w_table, ~] = fCalc_geofprops2(output.SF.P_2, output.SF.T_2, ...
                                         input.general.w_NaCl, input.general.w_CO2, ...
                                         input.general.H2O_sat, output.SF.h_mix_2, output, ...
                                         output.SF.h_mix_2, input, data);
                                                    % geothermal fluid properties
        output.SF.T_2                = T_new;       % temperature [C]
        output.SF.chi_2              = geofprops(1,2);    % quality [-]
    end

    output.SF.w_H2O_g_2(1,1) = w_table(3,6);  % gas mass fraction H2O state 2
    output.SF.w_CO2_g_2(1,1) = w_table(3,4);  % gas mass fraction CO2 state 2
    output.SF.T_2(1,1)       = output.SF.T_2; % [C]
    output.SF.P_2(1,1)       = output.SF.P_2; % [bar]
else
    % initial conditions
    output.SF.T_2(1,1)       = output.prodwell.T(end);       % [C]
    output.SF.P_2(1,1)       = output.prodwell.P(end);       % [bar]
    output.SF.h_mix_2        = output.prodwell.h(end);       % [J/kg]
    output.SF.chi_2          = output.prodwell.chi(end);     % quality [-]
    output.SF.w_H2O_g_2(1,1) = output.prodwell.w_H2O_g(end); % gas mass fraction H2O state 2
    output.SF.w_CO2_g_2(1,1) = output.prodwell.w_CO2_g(end); % gas mass fraction CO2 state 2
end

% End check for gas mass fraction AND determining of initial flash properties %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% user-defined input parameters single-flash power plant
output.SF.P_out_t  = input.SF.P_out_t;  % pressure outlet turbine [bar]
output.SF.eta_t    = input.SF.eta_t;    % turbine efficiency
output.SF.eta_td   = input.SF.eta_td;   % dry turbine efficiency
output.SF.eta_p    = input.SF.eta_p;    % pump efficiency
output.SF.eta_g    = input.SF.eta_g;    % generator efficiency
output.SF.T_out_cd = input.SF.T_out_cd; % outlet temperature condenser

%% Geothermal fluid properties @ state 4
output.SF.chi_4 = 1;                                    % saturated gas at inlet turbine
output.SF.n_H2O_v_4(1,1) = (output.SF.w_H2O_g_2/input.general.M_H2O) / (output.SF.w_H2O_g_2...
                           /input.general.M_H2O + output.SF.w_CO2_g_2/input.general.M_CO2);
                                                        % mole fraction H2O in gas phase state 4
output.SF.n_CO2_v_4(1,1) = (output.SF.w_CO2_g_2/input.general.M_CO2) / (output.SF.w_H2O_g_2...
                           /input.general.M_H2O + output.SF.w_CO2_g_2/input.general.M_CO2);
                                                        % mole fraction CO2 in gas phase state 4
output.SF.P_H2O_4(1,1)   = output.SF.n_H2O_v_4(1,1) * output.SF.P_2(1,1);
                                                        % partial pressure H2O at state 4 [bar]
output.SF.P_CO2_4(1,1)   = output.SF.n_CO2_v_4(1,1) * output.SF.P_2(1,1);
                                                        % partial pressure CO2 at state 4 [bar]
output.SF.h_H2O_v_4(1,1) = interp1(data.H2O_sat_props(:,1), data.H2O_sat_props(:,4), ...
                           output.SF.P_H2O_4(1,1),'spline');
                                                        % enthalpy H2O in gas phase state 4 [J/kg]
output.SF.h_CO2_v_4(1,1) = interp2(data.P_CO2, data.T_CO2, data.h_CO2, ...
                           output.SF.P_CO2_4(1,1), output.SF.T_2(1,1),'spline');
                                                        % enthalpy CO2 in gas phase state 4 [J/kg]
output.SF.h_mix_v_4(1,1) = output.SF.h_H2O_v_4(1,1) * output.SF.w_H2O_g_2 + ...
                           output.SF.h_CO2_v_4(1,1) * output.SF.w_CO2_g_2;
                                                        % enthalpy vapor mixture at state 4 [J/kg]
output.SF.s_H2O_v_4(1,1) = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,6), ...
                           output.SF.T_2(1,1),'spline');
                                                        % entropy H2O in gas phase state 4 [J/kg/K]
output.SF.s_CO2_v_4(1,1) = interp2(data.P_CO2, data.T_CO2, data.s_CO2, ...
                           output.SF.P_CO2_4(1,1), output.SF.T_2(1,1),'spline');
                                                        % entropy CO2 in gas phase state 4 [J/kg/K]
output.SF.s_mix_v_4(1,1) = output.SF.s_H2O_v_4(1,1) * output.SF.w_H2O_g_2 + ...
                           output.SF.s_CO2_v_4(1,1) * output.SF.w_CO2_g_2;
                                                        % entropy vapor mixture at state 4 [J/kg/K]

%% Turbine expansion calculation state 5s
output.SF.P_5 = output.SF.P_out_t; % outlet pressure turbine (model input)
```

```
% initial variables for fCalc_chi5s iteration
output.SF.P0_H2O_7 = output.SF.P_5 * output.SF.n_H2O_v_4(1,1);
                                    % initial partial pressure H2O @ outlet turbine (5s)
output.SF.T0_5     = interp1(data.H2O_sat_props(:,1), data.H2O_sat_props(:,2), ...
                     output.SF.P0_H2O_7,'spline');
                                    % initial temperature @ outlet turbine (5s)
output.SF.chi_5s   = 1;             % initial quantity  @ outlet turbine (5s)

% Iterative procedure fCalc_chi5s
x0     = [output.SF.chi_5s, output.SF.T0_5]; % iteration variables
y0     = [output.SF.w_CO2_g_2, output.SF.P_5, output.SF.s_mix_v_4(1,1), input.general.M_CO2,...
          input.general.M_H2O];     % constant variables
f      = @(x0)fCalc_chi_5s(x0,y0);  % function
[out] = fsolve(f,x0,options);
save('output.mat','output');        % save all output so far

% Repeat calculation with output from solution to obtain all other relevant output.
x0          = [out(1),out(2)];
y0          = [output.SF.w_CO2_g_2, output.SF.P_5, output.SF.s_mix_v_4(1,1), ...
              input.general.M_CO2, input.general.M_H2O];
[~,output] = fCalc_chi_5s(x0,y0);
Newoutput  = output;
load('output.mat');                 % load all output so far

% Write Newoutput from fCalc_chi5s to output file
output.SF.chi_5s      = Newoutput.SF.chi_5s;      output.SF.T_5s      = Newoutput.SF.T_5s;
output.SF.w_CO2_g_2   = Newoutput.SF.w_CO2_g_2;   output.SF.P_5       = Newoutput.SF.P_5;
output.SF.s_mix_v_4   = Newoutput.SF.s_mix_v_4;   output.SF.w_CO2_7   = Newoutput.SF.w_CO2_7;
output.SF.w_H2O_7     = Newoutput.SF.w_H2O_7;     output.SF.n_H2O_7   = Newoutput.SF.n_H2O_7;
output.SF.P_H2O_7     = Newoutput.SF.P_H2O_7;     output.SF.P_CO2_7   = Newoutput.SF.P_CO2_7;
output.SF.T_5s_check  = Newoutput.SF.T_5s_check;  output.SF.h_H2O_6   = Newoutput.SF.h_H2O_6;
output.SF.s_H2O_6     = Newoutput.SF.s_H2O_6;     output.SF.h_H2O_7   = Newoutput.SF.h_H2O_7;
output.SF.s_H2O_7     = Newoutput.SF.s_H2O_7;     output.SF.h_CO2_7   = Newoutput.SF.h_CO2_7;
output.SF.s_CO2_7     = Newoutput.SF.s_CO2_7;     output.SF.h_mix_7   = Newoutput.SF.h_mix_7;
output.SF.s_mix_7     = Newoutput.SF.s_mix_7;     output.SF.h_mix_5s  = Newoutput.SF.h_mix_5s;
output.SF.s_mix_5s    = Newoutput.SF.s_mix_5s;

%% Turbine expansion calculation state 5
output.SF.h_mix_5   = output.SF.h_mix_v_4 - output.SF.eta_t * (output.SF.h_mix_v_4 - ...
                      output.SF.h_mix_5s);          % initial enthalpy @ outlet turbine (5)
output.SF.chi_5     = (output.SF.h_mix_5 - output.SF.h_H2O_6) / (output.SF.h_mix_7 - ...
                      output.SF.h_H2O_6);           % initial quality @ outlet turbine (5)
output.SF.eta_t_old = 0;                            % initial old turbine efficiency
output.SF.eta_t_new = output.SF.eta_t;              % initial new turbine efficiency

while abs(output.SF.eta_t_new - output.SF.eta_t_old) > error_eta_t_SF
    x0      = [output.SF.chi_5, output.SF.T_5s];        % Iteration variables
    y0      = [output.SF.w_CO2_g_2, output.SF.P_5, output.SF.h_mix_5(1,1), ...
              input.general.M_CO2, input.general.M_H2O]; % Constant variables
    f       = @(x0)fCalc_chi_5(x0,y0);
    [out]   = fsolve(f,x0,options);
    save('output.mat','output');                    % save all output so far

    % Repeat calculation with output from solution to obtain all other relevant output.
    x0          = [out(1), out(2)];
    y0          = [output.SF.w_CO2_g_2, output.SF.P_5, output.SF.h_mix_5(1,1), ...
                  input.general.M_CO2, input.general.M_H2O];
    [~,output] = fCalc_chi_5(x0,y0);
    Newoutput  = output;
    load('output.mat');                             % load all output so far

    % Write Newoutput from fCalc_chi5s to output file
    output.SF.chi_5     = Newoutput.SF.chi_5;     output.SF.T_5       = Newoutput.SF.T_5;
    output.SF.w_CO2_g_2 = Newoutput.SF.w_CO2_g_2;output.SF.P_5       = Newoutput.SF.P_5;
    output.SF.w_CO2_7   = Newoutput.SF.w_CO2_7;   output.SF.w_H2O_7   = Newoutput.SF.w_H2O_7;
    output.SF.n_H2O_7   = Newoutput.SF.n_H2O_7;   output.SF.P_H2O_7   = Newoutput.SF.P_H2O_7;
    output.SF.P_CO2_7   = Newoutput.SF.P_CO2_7;   output.SF.T_5_check = Newoutput.SF.T_5_check;
    output.SF.h_H2O_6   = Newoutput.SF.h_H2O_6;   output.SF.s_H2O_6   = Newoutput.SF.s_H2O_6;
    output.SF.h_H2O_7   = Newoutput.SF.h_H2O_7;   output.SF.s_H2O_7   = Newoutput.SF.s_H2O_7;
    output.SF.h_CO2_7   = Newoutput.SF.h_CO2_7;   output.SF.s_CO2_7   = Newoutput.SF.s_CO2_7;
    output.SF.h_mix_7   = Newoutput.SF.h_mix_7;   output.SF.s_mix_7   = Newoutput.SF.s_mix_7;
    output.SF.h_mix_5   = Newoutput.SF.h_mix_5;   output.SF.s_mix_5   = Newoutput.SF.s_mix_5;

    % Calculate new wet turbine efficiency and mixture enthalpy @ state 5
    output.SF.eta_t_old = output.SF.eta_t_new;
    output.SF.eta_t_new = output.SF.eta_td * (output.SF.chi_4 + output.SF.chi_5)/2;
                                                    % wet turbine efficiency
    output.SF.h_mix_5   = output.SF.h_mix_v_4 - output.SF.eta_t_new * (output.SF.h_mix_v_4 ...
                          - output.SF.h_mix_5s);    % mixture enthalpy [J/kg] state 5
end

output.SF.eta_t(1,1) = output.SF.eta_t_new;                     % final turbine efficiency


%% Calculate state 11 - condenser outlet gas
% Extra check to make sure condensation is isobaric and isothermal for pure water
if output.SF.w_CO2_g_2 == 0
    output.SF.T_11 = output.SF.T_5;
end

output.SF.P_H2O_11   = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,1), ...
                       output.SF.T_11,'spline');        % partial pressure H2O @ state 11
output.SF.P_CO2_11   = output.SF.P_5 - output.SF.P_H2O_11;  % partial pressure CO2 @ state 11
output.SF.P_mix_11   = output.SF.P_H2O_11 + output.SF.P_CO2_11;
                                                % total pressure ideal gas mixture @ state 11
output.SF.n_CO2_v_11 = output.SF.P_CO2_11/output.SF.P_5; % mole fraction CO2 in gas @ state 11
output.SF.n_H2O_v_11 = 1 - output.SF.n_CO2_v_11;        % mole fraction H2O in gas @ state 11
output.SF.w_CO2_g_11 = output.SF.n_CO2_v_11/((input.general.M_H2O/input.general.M_CO2)...
                       -((input.general.M_H2O/input.general.M_CO2) - 1)*output.SF.n_CO2_v_11);
                                                % mass fraction CO2 in gas @ state 11
output.SF.w_H2O_g_11 = 1 - output.SF.w_CO2_g_11;        % mass fraction H2O in gas @ state 11

output.SF.h_H2O_v_11 = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,4), ...
                       output.SF.T_11,'spline');        % enthalpy H2O in gas @ state 11
output.SF.s_H2O_v_11 = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,6), ...
                       output.SF.T_11,'spline');        % entropy H2O in gas @ state 11
output.SF.h_CO2_v_11 = interp2(data.P_CO2, data.T_CO2, data.h_CO2, output.SF.P_CO2_11, ...
                       output.SF.T_11,'spline');        % enthalpy CO2 in gas @ state 11
output.SF.s_CO2_v_11 = interp2(data.P_CO2, data.T_CO2, data.s_CO2, output.SF.P_CO2_11, ...
                       output.SF.T_11,'spline');        % entropy CO2 in gas @ state 11
output.SF.h_mix_v_11 = output.SF.h_H2O_v_11 * output.SF.w_H2O_g_11 + output.SF.h_CO2_v_11...
                       * output.SF.w_CO2_g_11;          % enthalpy gas mix @ state 11
output.SF.s_mix_v_11 = output.SF.s_H2O_v_11 * output.SF.w_H2O_g_11 + output.SF.s_CO2_v_11...
                       * output.SF.w_CO2_g_11;          % entropy gas mix @ state 11

%% Calculate state 12 steam ejector/condenser
output.SF.m_4(1,1)  = input.general.m_gf * output.SF.chi_2(1,1);
output.SF.m_5(1,1)  = output.SF.m_4(1,1); % initial value
output.SF.m_mf(1,1) = 0;                  % initial value
output.SF.m_mix_v_11(1,1) = (output.SF.w_CO2_g_2/output.SF.w_CO2_g_11) * output.SF.m_5(1,1);
if output.SF.w_CO2_g_11 > 0 && compressor == 0
load ASR_curves; load CR_data; load f_TCF_air; load f_TCF_steam; load f_WER;
output.SF.m_4(1,1)  = input.general.m_gf * output.SF.chi_2(1,1);
output.SF.m_mix_v_11(1,1) = (output.SF.w_CO2_g_2/output.SF.w_CO2_g_11) * output.SF.m_5(1,1);
                                        % massflow CO2 + H2O mixture through @ state 11

output.SF.m_mix_v_11_old = output.SF.m_mix_v_11(1,1) + 0.2;
output.SF.m_mix_v_11_new = output.SF.m_mix_v_11(1,1);

output.SF.TCF_CO2(1,1) = f_TCF_air(output.SF.T_11);   % temperature correction factor
output.SF.TCF_H2O(1,1) = f_TCF_steam(output.SF.T_11); % temperature correction factor
output.SF.WER_CO2(1,1) = f_WER(input.general.M_CO2);  % weigth entrainment ratio
output.SF.WER_H2O(1,1) = f_WER(input.general.M_H2O);  % weigth entrainment ratio

while abs(output.SF.m_mix_v_11_old - output.SF.m_mix_v_11_new) > 0.001;
    output.SF.m_mix_v_11(1,1) = output.SF.m_mix_v_11_new;
    output.SF.DAE_H2O_11(1,1) = output.SF.m_mix_v_11(1,1) * output.SF.w_H2O_g_11(1,1)/...
                                (output.SF.TCF_H2O(1,1) * output.SF.WER_H2O(1,1));
                                                % Dry air equivalent water
    output.SF.DAE_CO2_11(1,1) = output.SF.m_mix_v_11(1,1) * output.SF.w_CO2_g_11/...
                                (output.SF.TCF_CO2(1,1) * output.SF.WER_CO2(1,1));
                                                % Dry air equivalent CO2
    output.SF.DAE_11(1,1) = output.SF.DAE_CO2_11(1,1) + output.SF.DAE_H2O_11(1,1);
                                                % Dry air equivalent mix
    output.SF.P_mix_d11(1,1) = sqrt(output.SF.P_mix_11(1,1) * input.general.P_atm);
                                                % pressure 2nd stage SE/C
    output.SF.CR(1,1) = output.SF.P_mix_d11(1,1)/output.SF.P_mix_11(1,1); % compression ratio
    output.SF.ER_11(1,1) = output.SF.P_2(1,1)/output.SF.P_mix_11(1,1);       % expansion ratio

    % determination of the air to steam ratio (ASR)
    A1 = CR_data(find(CR_data < output.SF.CR(1,1),1));
    A2 = CR_data(find(CR_data > output.SF.CR(1,1)));
    A2 = A2(end);
    B1 = '_';
    B2 = 'f';
    if A1(mod(A1,1) == 0)
        formatSpec = 'f_ASR%s%d';
```

```
            str1 = sprintf(formatSpec,B1,A1);
        else
            A11 = round(A1,1);
            A11 = round(10*rem(A11,1));
            formatSpec = 'f_ASR%s%d%s%d';
            str1 = sprintf(formatSpec,B1,floor(A1),B2,A11);
        end
        if A2(mod(A2,1) == 0)
            formatSpec = 'f_ASR%s%d';
            str2 = sprintf(formatSpec,B1,A2(end));
        else
            A22 = round(A2,1);
            A22 = round(10*rem(A22,1));
            formatSpec = 'f_ASR%s%d%s%d';
            str2 = sprintf(formatSpec,B1,floor(A2),B2,A22);
        end
        C1 = eval(str1);
        C2 = eval(str2);
        D1 = C1(output.SF.ER_11(1,1));
        D2 = C2(output.SF.ER_11(1,1));
        output.SF.ASR_11(1,1) = interp1([A1 A2(end)],[D1 D2], output.SF.CR(1,1));
                                                        % Air to steam ratio
        output.SF.m_mf11(1,1) = output.SF.DAE_11(1,1)/output.SF.ASR_11(1,1);
                                                        % mass flow rate motive flow
%         output.SF.m_mix_v_11_old = output.SF.m_mix_v_11_new;
%         output.SF.m_5(1,1)  = output.SF.m_4(1,1) - output.SF.m_mf_11(1,1); % mass flow outlet turbine
%         output.SF.m_mix_v_11_new = (output.SF.w_CO2_g_2/output.SF.w_CO2_g_11)* output.SF.m_5(1,1);

    output.SF.P_H2O_s12(1,1)   = output.SF.P_H2O_11(1,1);
                                                % outlet temperature SE/C equal to condenser
    output.SF.P_CO2_s12(1,1)   = output.SF.P_mix_d11(1,1) - output.SF.P_H2O_s12(1,1);
    output.SF.P_mix_s12(1,1)   = output.SF.P_mix_d11(1,1);
    output.SF.n_CO2_v_s12(1,1) = output.SF.P_CO2_s12(1,1)/output.SF.P_mix_s12(1,1);
                                                % mole fraction CO2 in gas @ state 12
    output.SF.n_H2O_v_s12(1,1) = 1 - output.SF.n_CO2_v_s12(1,1);% mole fraction H2O in gas @ st 12
    output.SF.w_CO2_g_s12(1,1) = output.SF.n_CO2_v_s12(1,1)/((input.general.M_H2O/...
                                 input.general.M_CO2)-((input.general.M_H2O/input.general.M_CO2)...
                                 - 1)*output.SF.n_CO2_v_s12(1,1));
                                                % mass fraction CO2 in gas @ state 11
    output.SF.w_H2O_g_s12(1,1) = 1 - output.SF.w_CO2_g_s12(1,1);
                                                % mass fraction H2O in gas @ state 11
    output.SF.m_d11(1,1)    = output.SF.m_mf11(1,1) + output.SF.m_mix_v_11_new;
    output.SF.w_CO2_d11(1,1) = (output.SF.m_mf11(1,1) * output.SF.w_CO2_g_2(1,1) + ...
                               output.SF.w_CO2_g_11 * output.SF.m_mix_v_11_new)/...
                               output.SF.m_d11(1,1);
    output.SF.w_H2O_d11(1,1)   = 1 - output.SF.w_CO2_d11(1,1);
    output.SF.m_mix_v_s12(1,1) = (output.SF.w_CO2_d11(1,1)/output.SF.w_CO2_g_s12) * ...
                                 output.SF.m_d11(1,1);

    output.SF.DAE_H2O_12(1,1) = output.SF.m_mix_v_s12(1,1) * output.SF.w_H2O_g_s12(1,1)/...
                                (output.SF.TCF_H2O(1,1) * output.SF.WER_H2O(1,1));
                                                % Dry air equivalent water
    output.SF.DAE_CO2_12(1,1) = output.SF.m_mix_v_s12(1,1) * output.SF.w_CO2_g_s12/...
                                (output.SF.TCF_CO2(1,1) * output.SF.WER_CO2(1,1));
                                                % Dry air equivalent CO2
    output.SF.DAE_12(1,1) = output.SF.DAE_CO2_12(1,1) + output.SF.DAE_H2O_12(1,1);
                                                % Dry air equivalent mix
    output.SF.P_mix_d12(1,1) = input.general.P_atm; % pressure outlet SE/C
    output.SF.ER_12(1,1) = output.SF.P_2(1,1)/output.SF.P_mix_s12(1,1); % expansion ratio

        % determination of the air to steam ratio (ASR)
        A1 = CR_data(find(CR_data < output.SF.CR(1,1),1));
        A2 = CR_data(find(CR_data > output.SF.CR(1,1)));
        A2 = A2(end);
        B1 = '_';
        B2 = 'f';
        if A1(mod(A1,1) == 0)
            formatSpec = 'f_ASR%s%d';
            str1 = sprintf(formatSpec,B1,A1);
        else
            A11 = round(A1,1);
            A11 = round(10*rem(A11,1));
            formatSpec = 'f_ASR%s%d%s%d';
            str1 = sprintf(formatSpec,B1,floor(A1),B2,A11);
        end
        if A2(mod(A2,1) == 0)
            formatSpec = 'f_ASR%s%d';
            str2 = sprintf(formatSpec,B1,A2(end));
        else
            A22 = round(A2,1);
            A22 = round(10*rem(A22,1));
            formatSpec = 'f_ASR%s%d%s%d';
            str2 = sprintf(formatSpec,B1,floor(A2),B2,A22);
        end
        C1 = eval(str1);
        C2 = eval(str2);
        D1 = C1(output.SF.ER_12(1,1));
        D2 = C2(output.SF.ER_12(1,1));
        output.SF.ASR_12(1,1) = interp1([A1 A2(end)],[D1 D2], output.SF.CR(1,1));
                                                        % Air to steam ratio
        output.SF.m_mf12(1,1) = output.SF.DAE_12(1,1)/output.SF.ASR_12(1,1);
                                                        % mass flow rate motive flow
        output.SF.m_mix_v_11_old = output.SF.m_mix_v_11_new;
        output.SF.m_mf = output.SF.m_mf11(1,1) + output.SF.m_mf12(1,1);
        output.SF.m_5(1,1)  = output.SF.m_4(1,1) - output.SF.m_mf; % mass flow outlet turbine
        output.SF.m_mix_v_11_new = (output.SF.w_CO2_g_2/output.SF.w_CO2_g_11)* output.SF.m_5(1,1);
        output.SF.m_mix_v_11_new = output.SF.m_mix_v_11_old + ((output.SF.m_mix_v_11_new - ...
                                   output.SF.m_mix_v_11_old)/2);
    end
    end
%%  old
    if compressor == 1
        output.SF.P_H2O_12_com = input.general.P_atm * output.SF.n_H2O_v_11(1,1); % partial p @ 12
        output.SF.P_CO2_12_com = input.general.P_atm * output.SF.n_CO2_v_11(1,1); % partial p @ 12
        output.SF.T0_12    = T0_12;                         % initial temperature [C] @ state 12

        % iterative procedure centrifugal compressor
        x0    = [output.SF.T0_12];                                % iteration variable
        y0    = [output.SF.P_H2O_12_com, output.SF.P_CO2_12_com, output.SF.s_mix_v_11, ...
            output.SF.w_CO2_g_11, output.SF.w_H2O_g_11];          % iteration constants
        f     = @(x0)fCalc_T_12s(x0,y0);
        [out] = fsolve(f,x0,options);
        save('output.mat','output');                            % save all output so far

        x0 = out(1);
        y0 = [output.SF.P_H2O_12_com, output.SF.P_CO2_12_com, output.SF.s_mix_v_11, ...
            output.SF.w_CO2_g_11, output.SF.w_H2O_g_11];
        [~,output] = fCalc_T_12s(x0,y0);
        Newoutput  = output;
        load('output.mat');                                     % load all output so far

        % Write Newoutput from fCalc_T_12s to output file
        output.SF.T_12s = Newoutput.SF.T_12s;
        output.SF.h_H2O_v_12s_com = Newoutput.SF.h_H2O_v_12s;
        output.SF.s_H2O_v_12s_com = Newoutput.SF.s_H2O_v_12s;
        output.SF.h_CO2_v_12s_com = Newoutput.SF.h_CO2_v_12s;
        output.SF.s_CO2_v_12s_com = Newoutput.SF.s_CO2_v_12s;
        output.SF.h_mix_v_12s_com = Newoutput.SF.h_mix_v_12s;
        output.SF.s_mix_v_12s_com = Newoutput.SF.s_mix_v_12s;

        output.SF.h_mix_v_12_com  = output.SF.h_mix_v_11 + (output.SF.h_mix_v_12s_com - ...
            output.SF.h_mix_v_11)/input.SF.eta_SEC;    % enthalpy gas mixture @ 12

        % Calculate power machinery
        output.SF.W_SEC = (output.SF.h_mix_v_12_com - output.SF.h_mix_v_11) * ...
            output.SF.m_mix_v_11/1000;                      % Required power SE/C [MW]

    end
    output.SF.W_t     = (output.SF.h_mix_v_4 - output.SF.h_mix_5) * output.SF.m_5(1,1)/1000;
                                                                % gross turbine power [MW]
    output.SF.W_g     = output.SF.W_t * output.SF.eta_g;            % generated power [MW]
    output.SF.h_H2O_6 = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,3), ...
                        output.SF.T_11,'spline');
    output.SF.rho_6   = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,7), ...
                        output.SF.T_11,'spline');             % density saturated liquid [kg/m3]
    output.SF.h_out_cd = (output.SF.h_H2O_6 * (output.SF.m_5 - output.SF.m_mix_v_11) + ...
                         output.SF.h_mix_v_11 * output.SF.m_mix_v_11)/output.SF.m_5;
    output.SF.W_cp     = ((1/output.SF.rho_6) * (output.SF.P_2 - output.SF.P_5) * 100000 * ...
                         (input.general.m_gf * output.SF.chi_2 - output.SF.m_mix_v_11))/...
                         output.SF.eta_p/1000000;                % Power condenser pump [MW]
    output.SF.dQ_cd    = (output.SF.h_mix_5 - output.SF.h_out_cd) * output.SF.m_5;
    output.SF.T_cw_out = output.SF.T_11 - input.SF.T_pinch_cd;
    output.SF.T_cw_avg = (output.SF.T_cw_out + input.general.T_surf_w) / 2;
    output.SF.c_p_cw   = interp2(data.P_H2O_SC, data.T_H2O_SC, data.cp_H2O_SC, input.SF.dP_cwp ...
                         + input.general.P_atm, output.SF.T_cw_avg);
    output.SF.m_cw     = output.SF.dQ_cd*1000 / (output.SF.c_p_cw *(output.SF.T_cw_out - ...
                         input.general.T_surf_w));
    output.SF.rho_cw   = interp2(data.P_H2O_SC, data.T_H2O_SC, data.rho_H2O_SC, input.SF.dP_cwp...
```

```
                     + input.general.P_atm, output.SF.T_cw_avg);
output.SF.W_cwp    = ((1/output.SF.rho_cw) * input.SF.dP_cwp * 100000 * output.SF.m_cw / ...
                     input.SF.eta_p)/1000000; %[MW]
if compressor == 0
    output.SF.W_net = output.SF.W_g - output.SF.W_cp-output.SF.W_cwp; % Provisional W_net [MW]
elseif compressor == 1
    output.SF.W_net = output.SF.W_g - output.SF.W_cp - output.SF.W_cwp - output.SF.W_SEC;
                                                                      % Provisional W_net [MW]
end

%%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Calculation of the highest power output                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

n_steps = ceil((output.SF.P_2(1,1)-1) / 0.1);    % maximum number of steps
n       = 2;                                    % repeat calculation procedure with 2nd iteration

formatSpec = 'Single flash power plant calculation.\nPlease wait...';
str        = sprintf(formatSpec);
h          = waitbar(0,str);

for i = 2:n_steps
    waitbar(n/n_steps)

    if i >= 3 && output.SF.P_2(i-1,1) > 10
        output.SF.P_2(i,1)      = 10; %output.SF.P_2(i-1,1) - (dP * 5);
    elseif i >= 3 && (output.SF.W_net(i-1,1) - output.SF.W_net(i-2,1)) < 0.03
        output.SF.P_2(i,1)      = output.SF.P_2(i-1,1) - (dP/5);
    else
        output.SF.P_2(i,1)      = output.SF.P_2(i-1,1) - dP;
                                      % pressure after flashing @ state 2 [bar]
    end

    output.SF.h_mix_2(i,1) = output.prodwell.h(end);
                                    % enthalpy of mixture @ state 2 - isenthalpic flashing
    output.SF.T_flash_old  = output.SF.T_2(i-1,1);
                                    % set initial flash temperature as old flash temperature

    [geofprops, T_new, w_table, ~] = fCalc_geofprops2(output.SF.P_2(i,1), ...
                                    output.SF.T_flash_old, input.general.w_NaCl, ...
                                    input.general.w_CO2, data.H2O_sat, ...
                                    output.SF.h_mix_2(i,1), output, output.SF.h_mix_2(i,1)...
                                    , input, data);  % geothermal fluid properties

    output.SF.T_2(i,1)    = T_new;          % temperature @ state 2
    output.SF.chi_2(i,1) = geofprops(1,2); % gas mass fraction @ state 2

    % Output geothermal fluid composition
    output.SF.w_CO2_g_2(i,1) = w_table(3,4); % mass fraction CO2 in gas @ state 2
    output.SF.w_H2O_g_2(i,1) = w_table(3,6); % mass fraction H2O in gas @ state 2

    %% Geothermal fluid properties @ state 4
    output.SF.chi_4          = 1;                    % saturated vapor quality @ state 4
    output.SF.n_H2O_v_4(i,1) = (output.SF.w_H2O_g_2(i,1)/input.general.M_H2O) / ...
                               (output.SF.w_H2O_g_2(i,1)/input.general.M_H2O + ...
                               output.SF.w_CO2_g_2(i,1)/input.general.M_CO2);
                                                     % mole fraction H2O in gas phase state 4
    output.SF.n_CO2_v_4(i,1) = (output.SF.w_CO2_g_2(i,1)/input.general.M_CO2) / ...
                               (output.SF.w_H2O_g_2(i,1)/input.general.M_H2O + ...
                               output.SF.w_CO2_g_2(i,1)/input.general.M_CO2);
                                                     % mole fraction CO2 in gas phase state 4
    output.SF.P_H2O_4(i,1)   = output.SF.n_H2O_v_4(i,1) * output.SF.P_2(i,1);
                                                     % partial pressure H2O @ state 4 [bar]
    output.SF.P_CO2_4(i,1)   = output.SF.n_CO2_v_4(i,1) * output.SF.P_2(i,1);
                                                     % partial pressure CO2 @ state 4 [bar]
    output.SF.h_H2O_v_4(i,1) = interp1(data.H2O_sat_props(:,1), data.H2O_sat_props(:,4), ...
                               output.SF.P_H2O_4(i,1),'spline');
                                                     % enthalpy H2O in gas phase state 4 [J/kg]
    output.SF.h_CO2_v_4(i,1) = interp2(data.P_CO2, data.T_CO2,data.h_CO2, ...
                               output.SF.P_CO2_4(i,1), output.SF.T_2(i,1),'spline');
                                                     % enthalpy CO2 in gas phase state 4 [J/kg]
    output.SF.h_mix_v_4(i,1) = output.SF.h_H2O_v_4(i,1) * output.SF.w_H2O_g_2(i,1) + ...
                               output.SF.h_CO2_v_4(i,1) * output.SF.w_CO2_g_2(i,1);
                                                     % enthalpy gas mixture at state 4 [J/kg]
    output.SF.s_H2O_v_4(i,1) = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,6), ...
                               output.SF.T_2(i,1),'spline');
                                                     % entropy H2O in gas phase state 4 [J/kg]
    output.SF.s_CO2_v_4(i,1) = interp2(data.P_CO2, data.T_CO2, data.s_CO2, ...
                               output.SF.P_CO2_4(i,1),output.SF.T_2(i,1),'spline');
                                                     % entropy CO2 in gas phase state 4 [J/kg]
    output.SF.s_mix_v_4(i,1) = output.SF.s_H2O_v_4(i,1) * output.SF.w_H2O_g_2(i,1) + ...
                               output.SF.s_CO2_v_4(i,1) * output.SF.w_CO2_g_2(i,1);
                                                     % entropy gas mixture at state 4 [J/kg]

    %% Turbine expansion calculation state 5s
    output.SF.P_5(i,1)       = output.SF.P_out_t;       % outlet pressure turbine (model input)
    output.SF.P0_H2O_7(i,1) = output.SF.P_5(i,1) * output.SF.n_H2O_v_4(i,1);
                                          % initial partial pressure H2O @ outlet turbine (5s)
    output.SF.T0_5(i,1)      = interp1(data.H2O_sat_props(:,1), data.H2O_sat_props(:,2), ...
                               output.SF.P0_H2O_7(i,1),'spline');
                                                   % initial temperature @ outlet turbine (5s)
    output.SF.chi_5s(i,1)    = 1;
                        % initial quantity after isentropic expansion @ outlet turbine (5s)

    % Iterative procedure fCalc_chi5s
    x0    = [output.SF.chi_5s(i,1),output.SF.T0_5(i,1)]; % iteration variables
    y0    = [output.SF.w_CO2_g_2(i,1),output.SF.P_5(i,1),output.SF.s_mix_v_4(i,1), ...
            input.general.M_CO2, input.general.M_H2O];   % iteration constants
    f     = @(x0)fCalc_chi_5s(x0,y0);
    [out] = fsolve(f,x0,options);
    save('output.mat','output');                          % save all output so far

    x0          = [out(1),out(2)];
    y0          = [output.SF.w_CO2_g_2(i,1),output.SF.P_5(i,1),output.SF.s_mix_v_4(i,1), ...
                  input.general.M_CO2, input.general.M_H2O];
    [~,output] = fCalc_chi_5s(x0,y0);
    Newoutput  = output;
    load('output.mat');                                   % load all output so far

    output.SF.chi_5s(i,1) = Newoutput.SF.chi_5s;
    output.SF.T_5s(i,1) = Newoutput.SF.T_5s;
    output.SF.P_5(i,1) = Newoutput.SF.P_5;
    output.SF.w_CO2_7(i,1) = Newoutput.SF.w_CO2_7;
    output.SF.w_H2O_7(i,1) = Newoutput.SF.w_H2O_7;
    output.SF.n_H2O_7(i,1) = Newoutput.SF.n_H2O_7;
    output.SF.P_H2O_7(i,1) = Newoutput.SF.P_H2O_7;
    output.SF.P_CO2_7(i,1) = Newoutput.SF.P_CO2_7;
    output.SF.h_H2O_6(i,1) = Newoutput.SF.h_H2O_6;
    output.SF.s_H2O_6(i,1) = Newoutput.SF.s_H2O_6;
    output.SF.h_H2O_7(i,1) = Newoutput.SF.h_H2O_7;
    output.SF.s_H2O_7(i,1) = Newoutput.SF.s_H2O_7;
    output.SF.h_CO2_7(i,1) = Newoutput.SF.h_CO2_7;
    output.SF.s_CO2_7(i,1) = Newoutput.SF.s_CO2_7;
    output.SF.h_mix_7(i,1) = Newoutput.SF.h_mix_7;
    output.SF.s_mix_7(i,1) = Newoutput.SF.s_mix_7;
    output.SF.h_mix_5s(i,1) = Newoutput.SF.h_mix_5s;
    output.SF.s_mix_5s(i,1) = Newoutput.SF.s_mix_5s;
    output.SF.T_5s_check(i,1) = Newoutput.SF.T_5s_check;
    output.SF.s_mix_v_4(i,1) = Newoutput.SF.s_mix_v_4;
    output.SF.w_CO2_g_2(i,1) = Newoutput.SF.w_CO2_g_2;

    %% Turbine expansion calculation state 5
    output.SF.h_mix_5(i,1) = output.SF.h_mix_v_4(i,1) - output.SF.eta_t(1,1) * ...
                             (output.SF.h_mix_v_4(i,1) - output.SF.h_mix_5s(i,1));
                                                        % initial enthalpy @ outlet turbine (5)
    output.SF.chi_5(i,1)   = (output.SF.h_mix_5(i,1) - output.SF.h_H2O_6(i,1)) / ...
                             (output.SF.h_mix_7(i,1) - output.SF.h_H2O_6(i,1));
                                                        % initial quality @ outlet turbine (5)
    output.SF.eta_t_old    = 0;
    output.SF.eta_t_new    = output.SF.eta_t;

    while abs(output.SF.eta_t_new - output.SF.eta_t_old) > error_eta_t_SF
        x0    = [output.SF.chi_5(i,1), output.SF.T_5s(i,1)];   % iteration variables
        y0    = [output.SF.w_CO2_g_2(i,1), output.SF.P_5(i,1), output.SF.h_mix_5(i,1), ...
                input.general.M_CO2, input.general.M_H2O];     % iteration constants
        f     = @(x0)fCalc_chi_5(x0,y0);
        [out] = fsolve(f,x0,options);
        save('output.mat','output');                           % save all output so far

        x0          = [out(1),out(2)];
        y0          = [output.SF.w_CO2_g_2(i,1), output.SF.P_5(i,1), output.SF.h_mix_5(i,1),...
                      input.general.M_CO2, input.general.M_H2O];
        [~,output] = fCalc_chi_5(x0,y0);
        Newoutput  = output;
        load('output.mat');                                    % load all output so far

        output.SF.chi_5(i,1)      = Newoutput.SF.chi_5;
        output.SF.T_5(i,1)        = Newoutput.SF.T_5;
```

```
        output.SF.w_CO2_g_2(i,1) = Newoutput.SF.w_CO2_g_2;
        output.SF.P_5(i,1)       = Newoutput.SF.P_5;
        output.SF.w_CO2_7(i,1)   = Newoutput.SF.w_CO2_7;
        output.SF.w_H2O_7(i,1)   = Newoutput.SF.w_H2O_7;
        output.SF.n_H2O_7(i,1)   = Newoutput.SF.n_H2O_7;
        output.SF.P_H2O_7(i,1)   = Newoutput.SF.P_H2O_7;
        output.SF.P_CO2_7(i,1)   = Newoutput.SF.P_CO2_7;
        output.SF.T_5_check(i,1) = Newoutput.SF.T_5_check;
        output.SF.h_H2O_6(i,1)   = Newoutput.SF.h_H2O_6;
        output.SF.s_H2O_6(i,1)   = Newoutput.SF.s_H2O_6;
        output.SF.h_H2O_7(i,1)   = Newoutput.SF.h_H2O_7;
        output.SF.s_H2O_7(i,1)   = Newoutput.SF.s_H2O_7;
        output.SF.h_CO2_7(i,1)   = Newoutput.SF.h_CO2_7;
        output.SF.s_CO2_7(i,1)   = Newoutput.SF.s_CO2_7;
        output.SF.h_mix_7(i,1)   = Newoutput.SF.h_mix_7;
        output.SF.s_mix_7(i,1)   = Newoutput.SF.s_mix_7;
        output.SF.h_mix_5(i,1)   = Newoutput.SF.h_mix_5;
        output.SF.s_mix_5(i,1)   = Newoutput.SF.s_mix_5;
        output.SF.h_mix_5(i,1)   = Newoutput.SF.h_mix_5;

        output.SF.eta_t_old   = output.SF.eta_t_new;
        output.SF.eta_t_new   = output.SF.eta_td * (output.SF.chi_4 + output.SF.chi_5(i,1))/2;
        output.SF.h_mix_5(i,1) = output.SF.h_mix_v_4(i,1) - output.SF.eta_t_new * ...
                                 (output.SF.h_mix_v_4(i,1) - output.SF.h_mix_5s(i,1));
    end

    output.SF.eta_t(i,1) = output.SF.eta_t_new;                    % final turbine efficiency

    %% Calculate state 11 - condenser outlet gas
    % Extra check to make sure condensation is isobaric and isothermal for pure water
    if output.SF.w_CO2_g_2 == 0
        output.SF.T_11 = output.SF.T_5(i,1);
    end

    output.SF.P_H2O_11(i,1)   = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,1), ...
                                output.SF.T_11,'spline');    % partial pressure H2O @ state 11
    output.SF.P_CO2_11(i,1)   = output.SF.P_out_t - output.SF.P_H2O_11(i,1);
                                                    % partial pressure CO2 @ state 11
    output.SF.P_mix_11(i,1)   = output.SF.P_H2O_11(i,1) + output.SF.P_CO2_11(i,1);
                                                % total pressure ideal gas mixture @ state 11
    output.SF.n_CO2_v_11(i,1) = output.SF.P_CO2_11(i,1)/output.SF.P_out_t;
                                                    % mole fraction CO2 in gas @ state 11
    output.SF.n_H2O_v_11(i,1) = 1 - output.SF.n_CO2_v_11(i,1);
                                                    % mole fraction H2O in gas @ state 11
    output.SF.w_CO2_g_11(i,1) = output.SF.n_CO2_v_11(i,1)/((input.general.M_H2O/...
                                input.general.M_CO2)-((input.general.M_H2O/...
                                input.general.M_CO2) - 1)*output.SF.n_CO2_v_11(i,1));
                                                    % mass fraction CO2 in gas @ state 11
    output.SF.w_H2O_g_11(i,1) = 1 - output.SF.w_CO2_g_11(i,1);
                                                    % mass fraction H2O in gas @ state 11
    output.SF.m_mix_v_11(i,1) = (output.SF.w_CO2_g_2(i,1)/output.SF.w_CO2_g_11(i,1)) * ...
                                output.SF.chi_2(i,1) * input.general.m_gf;
                                                % massflow CO2 + H2O mixture through @ state 11

    output.SF.h_H2O_v_11(i,1) = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,4), ...
                                output.SF.T_11,'spline');     % enthalpy H2O in gas @ state 11
    output.SF.s_H2O_v_11(i,1) = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,6), ...
                                output.SF.T_11,'spline');     % entropy H2O in gas @ state 11
    output.SF.h_CO2_v_11(i,1) = interp2(data.P_CO2, data.T_CO2, data.h_CO2, ...
                                output.SF.P_CO2_11(i,1), output.SF.T_11,'spline');
                                                    % enthalpy CO2 in gas @ state 11
    output.SF.s_CO2_v_11(i,1) = interp2(data.P_CO2, data.T_CO2, data.s_CO2, ...
                                output.SF.P_CO2_11(i,1), output.SF.T_11,'spline');
                                                    % entropy CO2 in gas @ state 11
    output.SF.h_mix_v_11(i,1) = output.SF.h_H2O_v_11(i,1) * output.SF.w_H2O_g_11(i,1) + ...
                                output.SF.h_CO2_v_11(i,1) * output.SF.w_CO2_g_11(i,1);
                                                    % enthalpy gas mix @ state 11
    output.SF.s_mix_v_11(i,1) = output.SF.s_H2O_v_11(i,1) * output.SF.w_H2O_g_11(i,1) + ...
                                output.SF.s_CO2_v_11(i,1) * output.SF.w_CO2_g_11(i,1);
                                                    % entropy gas mix @ state 11

%% Calculate state 12
output.SF.m_4(i,1)  = input.general.m_gf * output.SF.chi_2(i,1);
output.SF.m_5(i,1)  = output.SF.m_4(i,1); % initial value
output.SF.m_mf(i,1) = 0;                  % initial value
output.SF.m_mix_v_11(i,1) = (output.SF.w_CO2_g_2(i,1)/output.SF.w_CO2_g_11(i,1)) * output.SF.m_5(i,1);
if output.SF.w_CO2_g_11(i,1) > 0 && compressor == 0
load ASR_curves; load CR_data; load f_TCF_air; load f_TCF_steam; load f_WER;
output.SF.m_mix_v_11(i,1) = (output.SF.w_CO2_g_2(i,1)/output.SF.w_CO2_g_11(i,1)) * output.SF.m_5(i,1);
                                                % massflow CO2 + H2O mixture through @ state 11
output.SF.m_mix_v_11_old(i,1) = output.SF.m_mix_v_11(i,1) + 0.2;
output.SF.m_mix_v_11_new(i,1) = output.SF.m_mix_v_11(i,1);

output.SF.TCF_CO2(i,1) = f_TCF_air(output.SF.T_11);   % temperature correction factor
output.SF.TCF_H2O(i,1) = f_TCF_steam(output.SF.T_11); % temperature correction factor
output.SF.WER_CO2(i,1) = f_WER(input.general.M_CO2);  % weigth entrainment ratio
output.SF.WER_H2O(i,1) = f_WER(input.general.M_H2O);  % weigth entrainment ratio

while abs(output.SF.m_mix_v_11_old(i,1) - output.SF.m_mix_v_11_new(i,1)) > 0.001;
    output.SF.m_mix_v_11(i,1) = output.SF.m_mix_v_11_new(i,1);
    output.SF.DAE_H2O_11(i,1) = output.SF.m_mix_v_11(i,1) * output.SF.w_H2O_g_11(i,1)/...
                                (output.SF.TCF_H2O(i,1) * output.SF.WER_H2O(i,1));
                                                        % Dry air equivalent water
    output.SF.DAE_CO2_11(i,1) = output.SF.m_mix_v_11(i,1) * output.SF.w_CO2_g_11(i,1)/...
                                (output.SF.TCF_CO2(i,1) * output.SF.WER_CO2(i,1));
                                                         % Dry air equivalent CO2
    output.SF.DAE_11(i,1) = output.SF.DAE_CO2_11(i,1) + output.SF.DAE_H2O_11(i,1);
                                                        % Dry air equivalent mix
    output.SF.P_mix_d11(i,1) = sqrt(output.SF.P_mix_11(i,1) * input.general.P_atm);
                                                        % pressure 2nd stage SE/C
    output.SF.CR(i,1) = output.SF.P_mix_d11(i,1)/output.SF.P_mix_11(i,1); % compression ratio
    output.SF.ER_11(i,1) = output.SF.P_2(i,1)/output.SF.P_mix_11(i,1);       % expansion ratio

    % determination of the air to steam ratio (ASR)
    A1 = CR_data(find(CR_data < output.SF.CR(i,1),1));
    A2 = CR_data(find(CR_data > output.SF.CR(i,1)));
    A2 = A2(end);
    B1 = '_';
    B2 = 'f';
    if A1(mod(A1,1) == 0)
        formatSpec = 'f_ASR%s%d';
        str1 = sprintf(formatSpec,B1,A1);
    else
        A11 = round(A1,1);
        A11 = round(10*rem(A11,1));
        formatSpec = 'f_ASR%s%d%s%d';
        str1 = sprintf(formatSpec,B1,floor(A1),B2,A11);
    end
    if A2(mod(A2,1) == 0)
        formatSpec = 'f_ASR%s%d';
        str2 = sprintf(formatSpec,B1,A2(end));
    else
        A22 = round(A2,1);
        A22 = round(10*rem(A22,1));
        formatSpec = 'f_ASR%s%d%s%d';
        str2 = sprintf(formatSpec,B1,floor(A2),B2,A22);
    end
    C1 = eval(str1);
    C2 = eval(str2);
    D1 = C1(output.SF.ER_11(i,1));
    D2 = C2(output.SF.ER_11(i,1));
    output.SF.ASR_11(i,1) = interp1([A1 A2(end)],[D1 D2], output.SF.CR(i,1));
                                                                    % Air to steam ratio
    output.SF.m_mf11(i,1) = output.SF.DAE_11(i,1)/output.SF.ASR_11(i,1);
                                                            % mass flow rate motive flow
%       output.SF.m_mix_v_11_old = output.SF.m_mix_v_11_new;
%       output.SF.m_5(1,1)  = output.SF.m_4(1,1) - output.SF.m_mf_11(1,1); % mass flow outlet turbine
%       output.SF.m_mix_v_11_new = (output.SF.w_CO2_g_2/output.SF.w_CO2_g_11)* output.SF.m_5(1,1);

    output.SF.P_H2O_s12(i,1)   = output.SF.P_H2O_11(i,1);
                                                % outlet temperature SE/C equal to condenser
    output.SF.P_CO2_s12(i,1)   = output.SF.P_mix_d11(i,1) - output.SF.P_H2O_s12(i,1);
    output.SF.P_mix_s12(i,1)   = output.SF.P_mix_d11(i,1);
    output.SF.n_CO2_v_s12(i,1) = output.SF.P_CO2_s12(i,1)/output.SF.P_mix_s12(i,1);
                                                    % mole fraction CO2 in gas @ state 12
    output.SF.n_H2O_v_s12(i,1) = 1 - output.SF.n_CO2_v_s12(i,1);% mole fraction H2O in gas @ st 12
    output.SF.w_CO2_g_s12(i,1) = output.SF.n_CO2_v_s12(i,1)/((input.general.M_H2O/...
                                 input.general.M_CO2)-((input.general.M_H2O/input.general.M_CO2)...
                                 - 1)*output.SF.n_CO2_v_s12(i,1));
                                                    % mass fraction CO2 in gas @ state 11
    output.SF.w_H2O_g_s12(i,1) = 1 - output.SF.w_CO2_g_s12(i,1);
                                                    % mass fraction H2O in gas @ state 11
    output.SF.m_d11(i,1)     = output.SF.m_mf11(i,1) + output.SF.m_mix_v_11_new(i,1);
    output.SF.w_CO2_d11(i,1) = (output.SF.m_mf11(i,1) * output.SF.w_CO2_g_2(i,1) + ...
                                output.SF.w_CO2_g_11(i,1) * output.SF.m_mix_v_11_new(i,1))/...
                                output.SF.m_d11(i,1);
    output.SF.w_H2O_d11(i,1)   = 1 - output.SF.w_CO2_d11(i,1);
    output.SF.m_mix_v_s12(i,1) = (output.SF.w_CO2_d11(i,1)/output.SF.w_CO2_g_s12(i,1)) * ...
```

```
                                output.SF.m_d11(i,1);

        output.SF.DAE_H2O_12(i,1) = output.SF.m_mix_v_s12(i,1) * output.SF.w_H2O_g_s12(i,1)/...
                                    (output.SF.TCF_H2O(i,1) * output.SF.WER_H2O(i,1));
                                                                % Dry air equivalent water
        output.SF.DAE_CO2_12(i,1) = output.SF.m_mix_v_s12(i,1) * output.SF.w_CO2_g_s12(i,1)/...
                                    (output.SF.TCF_CO2(i,1) * output.SF.WER_CO2(i,1));
                                                                % Dry air equivalent CO2
        output.SF.DAE_12(i,1)     = output.SF.DAE_CO2_12(i,1) + output.SF.DAE_H2O_12(i,1);
                                                                % Dry air equivalent mix
        output.SF.P_mix_d12(i,1)  = input.general.P_atm; % pressure outlet SE/C
        output.SF.ER_12(i,1)      = output.SF.P_2(i,1)/output.SF.P_mix_s12(i,1); % expansion ratio

            % determination of the air to steam ratio (ASR)
            A1 = CR_data(find(CR_data < output.SF.CR(i,1),1));
            A2 = CR_data(find(CR_data > output.SF.CR(i,1)));
            A2 = A2(end);
            B1 = '_';
            B2 = 'f';
            if A1(mod(A1,1) == 0)
                formatSpec = 'f_ASR%s%d';
                str1 = sprintf(formatSpec,B1,A1);
            else
                A11 = round(A1,1);
                A11 = round(10*rem(A11,1));
                formatSpec = 'f_ASR%s%d%s%d';
                str1 = sprintf(formatSpec,B1,floor(A1),B2,A11);
            end
            if A2(mod(A2,1) == 0)
                formatSpec = 'f_ASR%s%d';
                str2 = sprintf(formatSpec,B1,A2(end));
            else
                A22 = round(A2,1);
                A22 = round(10*rem(A22,1));
                formatSpec = 'f_ASR%s%d%s%d';
                str2 = sprintf(formatSpec,B1,floor(A2),B2,A22);
            end
            C1 = eval(str1);
            C2 = eval(str2);
            D1 = C1(output.SF.ER_12(i,1));
            D2 = C2(output.SF.ER_12(i,1));
            output.SF.ASR_12(i,1) = interp1([A1 A2(end)],[D1 D2], output.SF.CR(i,1));
                                                                % Air to steam ratio
            output.SF.m_mf12(i,1) = output.SF.DAE_12(i,1)/output.SF.ASR_12(i,1); % mass flow rate motive flow
            output.SF.m_mix_v_11_old(i,1) = output.SF.m_mix_v_11_new(i,1);
            output.SF.m_mf(i,1) = output.SF.m_mf11(i,1) + output.SF.m_mf12(i,1);
            output.SF.m_5(i,1)  = output.SF.m_4(i,1) - output.SF.m_mf(i,1); % mass flow outlet turbine
            output.SF.m_mix_v_11_new(i,1) = (output.SF.w_CO2_g_2(i,1)/output.SF.w_CO2_g_11(i,1))*
output.SF.m_5(i,1);
            output.SF.m_mix_v_11_new(i,1) = output.SF.m_mix_v_11_old(i,1) + ...
                                            ((output.SF.m_mix_v_11_new(i,1) - ...
                                            output.SF.m_mix_v_11_old(i,1))/2);
        end
        end

%%  old
        if compressor == 1
        output.SF.P_H2O_12_com(i,1) = input.general.P_atm * output.SF.n_H2O_v_11(i,1);% partial p @ 12
        output.SF.P_CO2_12_com(i,1) = input.general.P_atm * output.SF.n_CO2_v_11(i,1);% partial p @ 12
        output.SF.T0_12     = T0_12;                            % initial temperature [C] @ state 12

        % iterative procedure centrifugal compressor
        x0     = [output.SF.T0_12];                                         % iteration variable
        y0     = [output.SF.P_H2O_12_com(i,1), output.SF.P_CO2_12_com(i,1), ...
                  output.SF.s_mix_v_11(i,1), output.SF.w_CO2_g_11(i,1), output.SF.w_H2O_g_11(i,1)];
                                                                    % iteration constants
        f      = @(x0)fCalc_T_12s(x0,y0);
        [out] = fsolve(f,x0,options);
        save('output.mat','output');                                % save all output so far

        x0 = out(1);
        y0 = [output.SF.P_H2O_12_com(i,1), output.SF.P_CO2_12_com(i,1), output.SF.s_mix_v_11(i,1), ...
              output.SF.w_CO2_g_11(i,1), output.SF.w_H2O_g_11(i,1)];
        [~,output] = fCalc_T_12s(x0,y0);
        Newoutput  = output;
        load('output.mat');                                         % load all output so far

        % Write Newoutput from fCalc_T_12s to output file
        output.SF.T_12s(i,1) = Newoutput.SF.T_12s;
```



```
output.SF.h_H2O_v_12s_com(i,1) = Newoutput.SF.h_H2O_v_12s;
output.SF.s_H2O_v_12s_com(i,1) = Newoutput.SF.s_H2O_v_12s;
output.SF.h_CO2_v_12s_com(i,1) = Newoutput.SF.h_CO2_v_12s;
output.SF.s_CO2_v_12s_com(i,1) = Newoutput.SF.s_CO2_v_12s;
output.SF.h_mix_v_12s_com(i,1) = Newoutput.SF.h_mix_v_12s;
output.SF.s_mix_v_12s_com(i,1) = Newoutput.SF.s_mix_v_12s;

output.SF.h_mix_v_12_com(i,1)  = output.SF.h_mix_v_11(i,1) + ...
                                 (output.SF.h_mix_v_12s_com(i,1) - ...
                                 output.SF.h_mix_v_11(i,1))/input.SF.eta_SEC;
                                                                % enthalpy gas mix @ 12
% Calculate power machinery
output.SF.W_SEC(i,1) = (output.SF.h_mix_v_12_com(i,1) - output.SF.h_mix_v_11(i,1)) * ...
                output.SF.m_mix_v_11(i,1)/1000;                 % Required power SE/C [MW]
end
    output.SF.W_t(i,1)   = (output.SF.h_mix_v_4(i,1) - output.SF.h_mix_5(i,1)) * ...
                           output.SF.m_5(i,1)/1000;
                                                                % gross turbine power [MW]
    output.SF.W_g(i,1)   = output.SF.W_t(i,1) * output.SF.eta_g;    % generated power [MW]
    output.SF.h_H2O_6    = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,3), ...
                           output.SF.T_11,'spline');
    output.SF.rho_6      = interp1(data.H2O_sat_props(:,2), data.H2O_sat_props(:,7), ...
                           output.SF.T_11,'spline');        % density saturated liquid [kg/m3]
    output.SF.h_out_cd(i,1) = (output.SF.h_H2O_6 * (output.SF.m_5(i,1) - ...
                              output.SF.m_mix_v_11(i,1)) + output.SF.h_mix_v_11(i,1) * ...
                              output.SF.m_mix_v_11(i,1))/output.SF.m_5(i,1);
    output.SF.W_cp(i,1)  = ((1/output.SF.rho_6) * (output.SF.P_2(i,1) - output.SF.P_5(i,1))...
                           * 100000 * (output.SF.m_5(i,1) - ...
                           output.SF.m_mix_v_11(i,1)))/output.SF.eta_p/1000000;
                                                                % Power condenser pump [MW]
    output.SF.dQ_cd(i,1) = (output.SF.h_mix_5(i,1) - output.SF.h_out_cd(i,1)) * ...
                            output.SF.m_5(i,1);
    output.SF.T_cw_out(i,1) = output.SF.T_11 - input.SF.T_pinch_cd;
    output.SF.T_cw_avg(i,1) = (output.SF.T_cw_out(i,1) + input.general.T_surf_w) / 2;
    output.SF.c_p_cw(i,1)   = interp2(data.P_H2O_SC, data.T_H2O_SC, data.cp_H2O_SC, ...
                              input.SF.dP_cwp + input.general.P_atm, output.SF.T_cw_avg(i,1));
    output.SF.m_cw(i,1)     = output.SF.dQ_cd(i,1)*1000 / (output.SF.c_p_cw(i,1) *...
                              (output.SF.T_cw_out(i,1) - input.general.T_surf_w));
    output.SF.rho_cw(i,1)   = interp2(data.P_H2O_SC, data.T_H2O_SC, data.rho_H2O_SC, ...
                              input.SF.dP_cwp + input.general.P_atm, output.SF.T_cw_avg(i,1));
    output.SF.W_cwp(i,1)    = ((1/output.SF.rho_cw(i,1)) * input.SF.dP_cwp* 100000 * ...
                              output.SF.m_cw(i,1) / input.SF.eta_p)/1000000; % [MW]
    if compressor == 0
    output.SF.W_net(i,1) = output.SF.W_g(i,1) - output.SF.W_cp(i,1) - output.SF.W_cwp(i,1);
                                                                % Provisional W_net [MW]
    elseif compressor == 1
    output.SF.W_net(i,1) = output.SF.W_g(i,1) - output.SF.W_cp(i,1) - output.SF.W_cwp(i,1)...
                           - output.SF.W_SEC(i,1); % Provisional W_net [MW]
    end
    %% If netto power of previous iteration is larger quit the iteration loop
    if i == n_steps || output.SF.P_2(i,1) < 1.5 % output.SF.W_net(i,1) < output.SF.W_net(i-1,1)
        break
    end
    if i >= 4
        if output.SF.W_net(i,1) < output.SF.W_net(i-1,1) && output.SF.W_net(i-1,1) < ...
                output.SF.W_net(i-2,1) && output.SF.W_net(i-2,1) < output.SF.W_net(i-3,1) ...
                && output.SF.W_net(i-3,1) < output.SF.W_net(i-4,1)
                                            % output.SF.W_net(i,1) < output.SF.W_net(i-1,1)
        break
        end
    end

    % waitbar progress calculation
    n = (n_steps - 2) * ((output.SF.W_net(i,1) - output.SF.W_net(1,1)) - ...
        (output.SF.W_net(i,1) - output.SF.W_net(i-1,1))) / (output.SF.W_net(i,1) - ...
        output.SF.W_net(1,1)) + 2;
end
close(h)

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%     Begin calculation of injection well input     %%%%%%%%%%

% properties @ state 3 and 4
output.SF.m_mix_l_3(:,1)  = input.general.m_gf * (1 - output.SF.chi_2(:,1));
                                                                % mass flow liquid mixture @ state 3
output.SF.w_NaCl_l_3(:,1) = input.general.w_NaCl * (input.general.m_gf / ...
                            output.SF.m_mix_l_3(:,1));% mass fraction NaCl in liquid @ state 3
output.SF.m_mix_v_4(:,1)  = input.general.m_gf * output.SF.chi_2(:,1);
                                                                % mass flowmixture  @ state 4
```

```
    output.SF.P_8(i,1)          = output.SF.P_2(i,1);
end

close(h)

formatSpec = 'Calculation heat capacity mixture @ state 3.\nPlease wait...';
str        = sprintf(formatSpec);
h          = waitbar(0,str);

% Obtain heat capacity c_p_3 @ state 3 from Francke Model
for i = 1:size(output.SF.P_2,1)
    waitbar(i/size(output.SF.P_2,1))
    Excel  = actxGetRunningServer('Excel.Application');
    Sheets = Excel.ActiveWorkBook.Sheets;
    sheet2 = get(Sheets, 'Item', 3); % liquid phase
    invoke(sheet2, 'Activate');
    sheet  = Excel.Activesheet;
    P = output.SF.P_2(i,1);
    T = output.SF.T_2(i,1);
    w_NaCl_l_3 = output.SF.w_NaCl_l_3(i,1);
    sheet.set('Range', 'C3', P);
    sheet.set('Range', 'C4', T);
    sheet.set('Range', 'C6', w_NaCl_l_3);
    sheet.set('Range', 'C7', 0);
    sheet.set('Range', 'C8', 0);
    range = sheet.get('Range', 'G4:G7');
    range.Value;
    data_FM = range.Value;
    output.SF.cp_mix_3(i,1) = cell2mat(data_FM(2,1));
end
close(h)

output.SF.cp_H2O_l_8  = interp2(data.P_H2O_SC, data.T_H2O_SC, data.cp_H2O_SC, ...
                        output.SF.P_8(:,1), output.SF.T_8(:,1),'spline');
                                                % heat capacity liquid H2O @ state 8
output.SF.cp_H2O_l_14 = interp2(data.P_H2O_SC, data.T_H2O_SC, data.cp_H2O_SC, ...
                        output.SF.P_14(:,1), output.SF.T_14(:,1),'spline');
                                                % heat capacity liquid H2O @ state 14

% Calculation temperature @ state 9 by combining streams 3,8 and 14.
output.SF.T_9(:,1) = ((output.SF.m_mix_l_3(:,1).* output.SF.cp_mix_3(:,1).* ...
                      output.SF.T_2(:,1)) + (output.SF.m_mix_l_8(:,1).* ...
                      output.SF.cp_H2O_l_8(:,1).* output.SF.T_8(:,1)) + ...
                      (output.SF.m_H2O_l_14(:,1).* output.SF.cp_H2O_l_14(:,1).* ...
                      output.SF.T_14(:,1)))./ ((output.SF.m_mix_l_3(:,1).* ...
                      output.SF.cp_mix_3(:,1)) + (output.SF.m_mix_l_8(:,1).* ...
                      output.SF.cp_H2O_l_8(:,1)) + (output.SF.m_H2O_l_14(:,1).* ...
                      output.SF.cp_H2O_l_14(:,1)));       % temperature mixture @ state 9

xx = min(output.SF.P_2):0.001:max(output.SF.P_2);
yy = interp1(output.SF.P_2,output.SF.W_net,xx);
z = find(yy == max(yy(:)));
output.SF.P_2_max = xx(1,401);
output.SF.T_9_max = interp1(output.SF.P_2, output.SF.T_9, output.SF.P_2_max,'spline');
figure
plot(output.SF.P_2,output.SF.W_net,xx,yy)
figure
plot(output.SF.P_2,output.SF.T_9)

%% Calculation of power consumption injection pump
elseif algorithm == 2
    output.SF.W_ip = ((1/output.injwell.rho(end)) * (output.injwell.P(end) - ...
                     output.SF.P_2(output.injwell.index,1)) * 100000 * input.general.m_gf)...
                     /output.SF.eta_p/1000000;              % Required power injection pump [MW]

    output.SF.W_net_max = output.SF.W_net(output.injwell.index,1) - output.SF.W_ip;
                                                                % Maximum netto power [MW]
    % exergy analysis
    [geofprops]        = fCalc_geofprops4(input.general.P_atm, input.general.T_surf_w, ...
                         input.general.w_NaCl, input.general.w_CO2);
    output.SF.h_0 = geofprops(10,2); %[J]
    output.SF.s_0 = interp2(data.P_H2O_SC,data.T_H2O_SC,data.s_H2O_SC,input.general.P_atm, ...
                    input.general.T_surf_w); %[kJ/kg/K]
    output.SF.h_res_out = output.prodwell.h(1,1); % [J]
    output.SF.s_res_out = interp2(data.P_H2O_SC,data.T_H2O_SC,data.s_H2O_SC,...
                          output.prodwell.P(1,1), output.prodwell.T(1,1)); %[kJ/kg/K]
    output.SF.e = ((output.SF.h_res_out - output.SF.h_0)/1000) - (input.general.T_surf_w + ...
                  273.15) * (output.SF.s_res_out - output.SF.s_0); %[kJ/kg]
    output.SF.E = (output.SF.e * input.general.m_gf)/1000; %[MW]
```



```
output.SF.m_CO2_v_4(:,1)  = output.SF.m_mix_v_4(:,1).* output.SF.w_CO2_g_2(:,1);
                                                % mass flow CO2 in gas @ state 4
output.SF.m_CO2_mix_1     = input.general.m_gf * input.general.w_CO2;
                                                % mass flow CO2 in mixture @ state 1
output.SF.m_CO2_l_3(:,1)  = output.SF.m_CO2_mix_1 - output.SF.m_CO2_v_4(:,1);
                                                % mass flow CO2 in liquid @ state 3
output.SF.w_CO2_l_3(:,1)  = output.SF.m_CO2_l_3(:,1)./output.SF.m_mix_l_3(:,1);
                                                % mass fraction CO2 in liquid @ state 3
output.SF.w_H2O_l_3(:,1)  = 1 - output.SF.w_NaCl_l_3(:,1) - output.SF.w_CO2_l_3(:,1);
                                                % mass fraction H2O in liquid @ state 3

% properties @ other relavant states
output.SF.w_CO2_l_9(:,1)  = output.SF.w_CO2_l_3(:,1).* (output.SF.m_mix_l_3(:,1)/...
                            input.general.m_gf);        % mass fraction CO2 in liquid @ state 9
output.SF.w_NaCl_l_9(:,1) = output.SF.w_NaCl_l_3(:,1).* (output.SF.m_mix_l_3(:,1)/...
                            input.general.m_gf);        % mass fraction NaCl in liquid @ state 9
output.SF.w_H2O_l_9(:,1)  = 1 - output.SF.w_CO2_l_9(:,1) - output.SF.w_NaCl_l_9(:,1);
                                                % mass fraction H2O in liquid @ state 9
output.SF.m_mix_l_8(:,1)  = output.SF.m_5(:,1) - output.SF.m_mix_v_11(:,1);
                                                % mass flow liquid mixture @ state 8
output.SF.m_H2O_l_13(:,1) = input.general.m_gf - output.SF.m_mix_l_8(:,1) - ...
                            output.SF.m_mix_l_3(:,1);
                                % mass flow liquid H2O @ state 13 (make-up water)
output.SF.m_H2O_l_14(:,1) = output.SF.m_H2O_l_13(:,1);
                                % mass flow liquid H2O @ state 14 (make-up water)
output.SF.rho_H2O_l_13    = interp2(data.P_H2O_SC, data.T_H2O_SC, data.rho_H2O_SC, ...
                            input.general.P_atm, input.general.T_surf_w,'spline');
                                % density liquid H2O @ state 13

% Calculation of netto power (W_ip inclusive)
output.SF.W_mp(:,1) = ((1/output.SF.rho_H2O_l_13) * (output.SF.P_2(:,1) - ...
                      input.general.P_atm) * 100000.* output.SF.m_H2O_l_13(:,1)) / ...
                      output.SF.eta_p /1000000;         % Required power make-up pump in MW
output.SF.W_net(:,1) = output.SF.W_net(:,1) - output.SF.W_mp(:,1);
                                                % netto power (W_ip inclusive) [MW]

formatSpec = 'Condenser pump and make-up pump calculation.\nPlease wait...';
str        = sprintf(formatSpec);
h          = waitbar(0,str);

%% Iterative procedure for determining T_8 and T_14
for i = 1:size(output.SF.P_2,1)
    waitbar(i/size(output.SF.P_2,1))
    output.SF.T_14(i,1) = input.general.T_surf_w;       % initial temperature [C] @ state 14

    % iterative procedure make-up pump
    x0    = [output.SF.T_14(i,1)];                      % iteration variable
    y0    = [output.SF.rho_H2O_l_13, output.SF.P_2(i,1)]; % iteration constants
    f     = @(x0)fCalc_T_14(x0,y0);
    [out] = fsolve(f,x0,options);
    save('output.mat','output');                        % save all output so far

    x0         = out(1);
    y0         = [output.SF.rho_H2O_l_13, output.SF.P_2(i,1)];
    [~,output] = fCalc_T_14(x0,y0);
    Newoutput  = output;
    load('output.mat');                                 % load all output so far

    output.SF.T_14(i,1)         = Newoutput.SF.T_14;
    output.SF.rho_H2O_l_14(i,1) = Newoutput.SF.rho_H2O_l_14;

    output.SF.T_8(i,1) = output.SF.T_11;                % initial temperature [C] @ state 14
    % iterative procedure condenser pump
    x0    = [output.SF.T_8(i,1)];                       % iteration variable
    y0    = [output.SF.rho_6, output.SF.P_2(i,1)];      % iteration constants
    f     = @(x0)fCalc_T_8(x0,y0);
    [out] = fsolve(f,x0,options);
    save('output.mat','output');                        % save all output so far

    x0         = out(1);
    y0         = [output.SF.rho_6, output.SF.P_2(i,1)];
    [~,output] = fCalc_T_8(x0,y0);
    Newoutput  = output;
    load('output.mat');                                 % load all output so far

    output.SF.T_8(i,1)        = Newoutput.SF.T_8;
    output.SF.rho_H2O_l_8(i,1) = Newoutput.SF.rho_H2O_l_8;

    output.SF.P_14(i,1)       = output.SF.P_2(i,1);
```

```
        output.SF.eta_u = output.SF.W_net_max/output.SF.E;

    end
end
```

```
%% fCalc_BC

% Simulation of binary cycle power plant
% Frank Niewold
% Released version 1.0, February 2017

function [input, output, stat] = fCalc_BC(input, output, status, data, algorithm)
    % numbers in output parameters correspond to single-flash power plant figure in report

    options = optimset('Display','off');
    load C5H12_sat_props; load P_C5H12_SC; load s_C5H12_SC; load h_C5H12_SC;
    load h_C5H12_SH; load P_C5H12_SH; load T_C5H12_SH; load s_C5H12_SH;
    load P_H2O_SC; load T_H2O_SC; load rho_H2O_SC; load s_H2O_SC;

    %dT_it = input.settings.dT_it_BC;
    % Succesfull simulation
    stat = status.SUCCES;
    if algorithm == 1 % first part of single-flash power plant calculation until injection pump

    output.BC.T_c_C5H12 = 187.83; % [C] critical temperature isopentane
    %output.BC.T_c_C5H12 = 151.96;

    output.BC.T_pinch_ev   = input.B.T_pinch_ev;     % [C] pinch point temperature
    output.BC.m_gf         = output.prodwell_GL.m_gf(end) * (1 - output.prodwell_GL.chi(end));

    output.BC.P_A(1,1)      = output.prodwell_GL.P(end);        % [bar]
    output.BC.T_A(1,1)      = output.prodwell_GL.T(end);        % [C]
    output.BC.w_NaCl_A(1,1) = output.prodwell_GL.w_NaCl_l(end); % [kg/kg]
    output.BC.w_CO2_A(1,1)  = output.prodwell_GL.w_CO2_l(end);  % [kg/kg]

    output.BC.T_wf_ev(1,1) = min(output.BC.T_c_C5H12-0.1,output.BC.T_A-input.B.T_pinch_ev);
                                                                % initial evaporation T[C]
    output.BC.P_out_t      = interp1(C5H12_sat_props(:,1),C5H12_sat_props(:,2),...
                             input.B.T_out_cd);

            %%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %                   Calculation of the highest power output                 %
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    n = round((output.BC.T_A - input.B.T_pinch_ev - input.B.T_out_cd)/input.settings.dT_evap);
    for i = 1:n-1
        %output.BC.Q_BC(i,1) = 1; output.BC.Q_45(i,1) = 0.12;
        %while abs(output.BC.Q_BC(i,1) - output.BC.Q_45(i,1)) > 0.1
        output.BC.T_B(i,1) = output.BC.T_wf_ev(i,1) + input.B.T_pinch_ev;
                                                        % T geofluid outlet evaporator [C]
        [geofprops]        = fCalc_geofprops4(output.BC.P_A(1,1), output.BC.T_A(1,1), ...
                             output.BC.w_NaCl_A(1,1), output.BC.w_CO2_A(1,1));
        output.BC.h_A(i,1) = geofprops(10,2);
        [geofprops]        = fCalc_geofprops4(output.BC.P_A(1,1), output.BC.T_B(i,1), ...
                             output.BC.w_NaCl_A(1,1), output.BC.w_CO2_A(1,1));
        output.BC.h_B(i,1) = geofprops(10,2);

        output.BC.T_C(1,1) = input.B.T_inj; % temperature outlet evaporator
        [geofprops]        = fCalc_geofprops4(output.BC.P_A(1,1), output.BC.T_C(1,1), ...
            output.BC.w_NaCl_A(1,1), output.BC.w_CO2_A(1,1));
        output.BC.h_C(i,1) = geofprops(10,2); %[J]

        output.BC.Q_AB(i,1) = (output.BC.h_A(i,1) - output.BC.h_B(i,1)) * output.BC.m_gf/10^6; % [MW]
        output.BC.Q_BC(i,1) = (output.BC.h_B(i,1) - output.BC.h_C(i,1)) * output.BC.m_gf/10^6; % [MW]
        output.BC.Q_AC(i,1) = (output.BC.h_A(i,1) - output.BC.h_C(i,1)) * output.BC.m_gf/10^6;

        output.BC.P_3(i,1)  = output.BC.P_out_t; % [bar] condenser pressure is outlet pressure turbine
        output.BC.h_3(i,1)  = interp1(C5H12_sat_props(:,2),C5H12_sat_props(:,3),output.BC.P_3(i,1));
        output.BC.s_3(i,1)  = interp1(C5H12_sat_props(:,2),C5H12_sat_props(:,5),output.BC.P_3(i,1));
        output.BC.s_4s(i,1) = output.BC.s_3(i,1);
        output.BC.P_4(i,1)  = interp1(C5H12_sat_props(:,1),C5H12_sat_props(:,2),...
            output.BC.T_wf_ev(i,1));
        output.BC.h_4s(i,1) = interp2(s_C5H12_SC,P_C5H12_SC,h_C5H12_SC,output.BC.s_4s(i,1),...
            output.BC.P_4(i,1));
        output.BC.h_4(i,1)  = output.BC.h_3(i,1) + (output.BC.h_4s(i,1) - output.BC.h_3(i,1))/...
            input.B.eta_p;
        output.BC.T_5(i,1)  = output.BC.T_wf_ev(i,1);
        output.BC.h_5(i,1)  = interp1(C5H12_sat_props(:,1),C5H12_sat_props(:,3),output.BC.T_5(i,1));
        output.BC.h_6(i,1)  = interp1(C5H12_sat_props(:,1),C5H12_sat_props(:,4),output.BC.T_5(i,1));
        output.BC.s_6(i,1)  = interp1(C5H12_sat_props(:,1),C5H12_sat_props(:,6),output.BC.T_5(i,1));

        output.BC.m_wf_BC(i,1) = output.BC.Q_BC(i,1)*1000/(output.BC.h_5(i,1) - output.BC.h_4(i,1));
        if output.BC.m_wf_BC(i,1) <= 0;
            output.BC.m_wf_BC(i,1) = 1;
        end
        output.BC.h_1(i,1)  = (output.BC.h_5(i,1)*output.BC.m_wf_BC(i,1) + output.BC.Q_AB(i,1)*1000)/...
            output.BC.m_wf_BC(i,1);
        if output.BC.h_1(i,1) > 980
            output.BC.h_1(i,1) = 980;
        end
        if output.BC.h_1(i,1) <= output.BC.h_6(i,1)
            output.BC.h_1(i,1) = output.BC.h_1(i,1);
            output.BC.T_1(i,1) = output.BC.T_5(i,1);
        elseif output.BC.h_1(i,1) >= output.BC.h_6(i,1)
            output.BC.T_1(i,1) = interp2(h_C5H12_SH,P_C5H12_SH,T_C5H12_SH,output.BC.h_1(i,1),...
                output.BC.P_4(i,1));
            if output.BC.T_1(i,1) >= output.BC.T_A(1,1) - input.B.T_pinch_ev;
                output.BC.T_1(i,1) = output.BC.T_A(1,1) - input.B.T_pinch_ev;
                x0   = output.BC.h_1(i,1);                      % iteration variable
                y0   = [output.BC.P_4(i,1), output.BC.T_1(i,1)]; % iteration constants
                f    = @(x0)fCalc_h_1(x0,y0);
                output.BC.h_1(i,1) = fsolve(f,x0,options);
                output.BC.m_wf_BC(i,1) = output.BC.Q_AC(i,1)*1000/...
                    (output.BC.h_1(i,1) - output.BC.h_4(i,1));
            else
                output.BC.h_1(i,1) = output.BC.h_1(i,1);
            end
        end

        output.BC.s_1(i,1)  = interp2(h_C5H12_SH,P_C5H12_SH,s_C5H12_SH,output.BC.h_1(i,1),...
                              output.BC.P_4(i,1));
        output.BC.s_2s(i,1) = output.BC.s_1(i,1);

        output.BC.P_2(i,1) = output.BC.P_3(i,1);
        x0   = output.BC.h_1(i,1);                      % iteration variable
        y0   = [output.BC.P_2(i,1), output.BC.s_2s(i,1)]; % iteration constants
        f    = @(x0)fCalc_h_2s(x0,y0);
        output.BC.h_2s(i,1) = fsolve(f,x0,options);

        output.BC.h_2(i,1) = output.BC.h_1(i,1) - (output.BC.h_1(i,1) - output.BC.h_2s(i,1))*...
            input.B.eta_td;
        output.BC.T_2(i,1) = interp2(h_C5H12_SH,P_C5H12_SH,T_C5H12_SH,output.BC.h_2(i,1),...
                             output.BC.P_2(i,1));
        output.BC.W_t(i,1) = ((output.BC.h_1(i,1) - output.BC.h_2(i,1)) * output.BC.m_wf_BC(i,1)/1000);
        output.BC.W_g(i,1) =  output.BC.W_t(i,1) * input.B.eta_g;
        output.BC.W_p(i,1) = (output.BC.h_4(i,1) - output.BC.h_3(i,1)) * output.BC.m_wf_BC(i,1)/1000;
        output.BC.W_com(i,1) = (output.prodwell_GL.h_CO2_2 - output.prodwell_GL.h_CO2_1) * ...
                               input.prodwell_GL.m_GL/1000;
        output.BC.W_com_atm(i,1) = (output.prodwell_GL.h_CO2_2_atm - output.prodwell_GL.h_CO2_1_atm)...
                                   * input.prodwell_GL.m_GL/1000;
        output.BC.rho_A = interp2(P_H2O_SC,T_H2O_SC,rho_H2O_SC,input.general.P_atm,...
                          input.general.T_surf_w);
        output.BC.W_mp(i,1) = (1/output.BC.rho_A) * (output.BC.P_A - ...
                              input.general.P_atm) * 100000.* (input.general.m_gf-output.BC.m_gf) /...
                              input.B.eta_p /1000000;           % Required power make-up pump in MW

        output.BC.T_cw_out = input.B.T_out_cd - input.B.T_pinch_cd;
        output.BC.T_cw_avg = (output.BC.T_cw_out + input.general.T_surf_w) / 2;
        output.BC.c_p_cw   = interp2(data.P_H2O_SC, data.T_H2O_SC, data.cp_H2O_SC, input.B.dP_cwp +...
                             input.general.P_atm, output.BC.T_cw_avg);
        output.BC.dQ_cd(i,1) = (output.BC.h_2(i,1) - output.BC.h_3(i,1)) * output.BC.m_wf_BC(i,1);
        output.BC.m_cw(i,1)  = output.BC.dQ_cd(i,1)*1000 / (output.BC.c_p_cw *(output.BC.T_cw_out ...
                               - input.general.T_surf_w));
        output.BC.rho_cw(i,1) = interp2(data.P_H2O_SC, data.T_H2O_SC, data.rho_H2O_SC, ...
                                input.B.dP_cwp + input.general.P_atm, output.BC.T_cw_avg);
        output.BC.W_cwp(i,1)  = ((1/output.BC.rho_cw(i,1)) * input.B.dP_cwp* 100000 * ...
                                output.BC.m_cw(i,1) / input.B.eta_p)/1000000; % [MW]
        output.BC.W_net(i,1)  = output.BC.W_g(i,1) - output.BC.W_p(i,1) - output.BC.W_com(i,1) ...
                                - output.BC.W_mp(i,1) - output.BC.W_cwp(i,1);
        output.BC.W_net_atm(i,1) = output.BC.W_g(i,1) - output.BC.W_p(i,1) - ...
                                   output.BC.W_com_atm(i,1) - output.BC.W_mp(i,1) - ...
                                   output.BC.W_cwp(i,1);
        output.BC.Q_45(i,1) = (output.BC.h_5(i,1) - output.BC.h_4(i,1)) * output.BC.m_wf_BC(i,1)/1000;
        output.BC.h_B(i,1)  = output.BC.h_C(i,1)  + (output.BC.Q_45(i,1)/output.BC.m_gf);

        if  i >= 2
            if output.BC.W_net(i,1) < output.BC.W_net(i-1,1) ...
```

```
                && output.BC.h_1(i,1) >= output.BC.h_6(i,1)
                break
            end
        end
        output.BC.T_wf_ev(i+1,1) = output.BC.T_wf_ev(i,1) - input.settings.dT_evap;
    end

    elseif algorithm == 2
        output.BC.W_ip = ((1/output.injwell_BC.rho(end)) * (output.injwell_BC.P(end) - ...
                          output.BC.P_A) * 100000 * input.general.m_gf)...
                          /input.B.eta_p/1000000;                % Required power injection pump [MW]


        output.BC.W_net_max = output.BC.W_net(output.injwell_BC.index,1) - output.BC.W_ip;
        output.BC.W_net_max_atm = output.BC.W_net_atm(output.injwell_BC.index,1) - output.BC.W_ip;
                                                                  % Maximum netto power [MW]
        % exergy analysis
        [geofprops]         = fCalc_geofprops4(input.general.P_atm, input.general.T_surf_w, ...
                              input.general.w_NaCl, input.general.w_CO2);
        output.BC.h_0 = geofprops(10,2); %[J]
        output.BC.s_0 = interp2(P_H2O_SC,T_H2O_SC,s_H2O_SC,input.general.P_atm, ...
                        input.general.T_surf_w); %[kJ/kg/K]
        output.BC.h_res_out = output.prodwell_GL.h(1,1); % [J]
        output.BC.s_res_out = interp2(P_H2O_SC,T_H2O_SC,s_H2O_SC,output.prodwell_GL.P(1,1), ...
                              output.prodwell_GL.T(1,1)); %[kJ/kg/K]
        output.BC.e = ((output.BC.h_res_out - output.BC.h_0)/1000) - (input.general.T_surf_w + ...
                      273.15) * (output.BC.s_res_out - output.BC.s_0); %[kJ/kg]
        output.BC.E = (output.BC.e * input.general.m_gf)/1000; %[MW]
        output.BC.eta_u = output.BC.W_net_max/output.BC.E;
        output.BC.eta_u_atm = output.BC.W_net_max_atm/output.BC.E;
   end
end
```

```
%% fCalc_injwell

% Simulation of the injection well of the single-flash power plant
% Frank Niewold
% Released version 1.0, February 2017

function [input, output, stat, geofprops] = fCalc_injwell(input, output, data, status)

    stat = status.SUCCES;

    % settings
    error_T_9_10 = input.settings.error_T_9_10;

    %% Write injection well dimensions to output file
    for i = 1:max(input.injwell.segment);
        output.injwell.segnr (i,1)   = input.injwell.segment(i,1);  % segment nr.
        output.injwell.D_i (i,1)     = input.injwell.D_i(i,1);      % inner diameter [m]
        output.injwell.dl (i,1)      = input.injwell.dl(i,1);       % length [m]
        output.injwell.dz(i,1)       = input.injwell.dz(i,1);       % height [m]
        output.injwell.tvd(i,1)      = input.injwell.tvd(i,1);      % true vertical depth tvd [m]
        output.injwell.grad_T_g(i,1) = input.injwell.grad_T_g(i,1); % geothermal temperature grad [K]
        output.injwell.eps_pipe(i,1) = input.injwell.eps_pipe(i,1); % absolute pipe roughness [m]
        output.injwell.k_r(i,1)      = input.injwell.k_r(i,1);      % rock thermal conductivity[W/m/K]
        output.injwell.alfa_r(i,1)   = input.injwell.alfa_r(i,1);   % rock thermal diffusivity[m2/s]
    end

    %% Import initial brine properties at bottom injection well from reservoir
    % Import composition from power plant
    output.injwell.P(1,1) = output.reservoir.geofprops(1,1); % pressure [bar]
    [~,index]             = max(output.SF.W_net);            % row number with maximum W_net
    output.injwell.T(1,1) = output.SF.T_9(index,1);          % temperature [C]
    output.injwell.T(2,1) = output.injwell.T(1,1) + 2;       % Initial value for while loop
    output.injwell.w_NaCl = output.SF.w_NaCl_l_9(index,1);   % mass fraction NaCl injection well
    output.injwell.w_CO2  = output.SF.w_CO2_l_9(index,1);    % mass fraction CO2 injection well

    l = 1; % iteration number

    %% Iterative procedure for calculation of injection well properties
    while abs(output.injwell.T(end) - output.SF.T_9(index,1)) > error_T_9_10;
        % geothermal fluid properties @ bottom of injection well
        [geofprops] = fCalc_geofprops1(output.injwell.P(1,1), output.injwell.T(1,1), ...
            output.injwell.w_NaCl, output.injwell.w_CO2, output);
        output.injwell.geofprops(1,1:5) = [output.injwell.P(1,1) output.injwell.T(1,1) ...
            output.injwell.w_NaCl output.injwell.w_CO2 (1 - output.injwell.w_NaCl - ...
            output.injwell.w_CO2)];
```



```
        output.injwell.geofprops(1,6:31) = geofprops(1,1:26);

        output.injwell.h(1,1)      = output.injwell.geofprops(1,11);  % enthalpy [J/kg]
        output.injwell.chi(1,1)    = output.injwell.geofprops(1,7);   % gas mass fraction [-]
        output.injwell.v_spec(1,1) = 1/output.injwell.geofprops(1,9); % specific volume [m3/kg]
        output.injwell.rho(1,1)    = output.injwell.geofprops(1,9);   % density [kg/m3]
        output.injwell.c_p(1,1)    = output.injwell.geofprops(1,10);  % heat capacity [J/kg/K]
        output.injwell.mu(1,1)     = output.injwell.geofprops(1,12);  % viscosity [Pa*s]
        output.injwell.eps_G(1,1)  = output.injwell.geofprops(1,8);   % void fraction [-]

        % Calculate initial properties at bottom injection well
        output.injwell.u(1,1) = fCalc_u(input.general.m_gf, output.injwell.rho(1,1), ...
                                output.injwell.D_i(1,1));                          % velocity [m/s]
        output.injwell.Re(1,1) = fCalc_Re(output.injwell.D_i(1,1), output.injwell.rho(1,1), ...
                                 output.injwell.u(1,1), output.injwell.mu(1,1));   % Reynolds number
        output.injwell.f(1,1) = fCalc_f(output.injwell.chi(1,1), output.injwell.eps_pipe(1,1), ...
                                output.injwell.D_i(1,1), output.injwell.Re(1,1));  % friction factor
        output.injwell.T_g = fCalc_T_g(input.general.T_surf_r, output.injwell.grad_T_g, ...
                             output.injwell.tvd);                      % Geothermal temperature [C]
        output.injwell.dQ(1,1) = fCalc_dQ(output.injwell.T(1,1), output.injwell.T_g(1,1), ...
                                 output.injwell.D_i(1,1), output.injwell.dl(1,1), input.general.m_gf...
                                 , input.general.gamma, input.general.t, output.injwell.k_r(1,1), ...
                                 output.injwell.alfa_r(1,1));
                                                          % heat exchange with surroundings[J/kg]
        output.injwell.dE_pot(1,1) = fCalc_dE_pot(input.general.g, output.injwell.dz(1,1));
                                                          % potential energy change [J/kg]
        output.injwell.dP_f(1,1)   = fCalc_dP_f(output.injwell.D_i (1,1), output.injwell.f(1,1), ...
                                     output.injwell.rho(1,1), output.injwell.u(1,1), ...
                                     output.injwell.dl(1,1));      % frictional pressure change [bar]
        output.injwell.dP_hs(1,1) = fCalc_dP_hs(input.general.g, output.injwell.rho(1,1), ...
                                    output.injwell.dz(1,1));       % hydrostatic pressure change [bar]

        formatSpec = ...
            'Injection well calculation (iteration #%d).\ndT > %d. Iterate until dT < 1.\nPlease wait...';
        A1  = l;
        A2  = floor(output.injwell.T(end) - output.SF.T_9(index,1));
        str = sprintf(formatSpec,A1,A2);
        h   = waitbar(0,str);

        %% Calculation of segment nr.2 to top of the injection well
        for i=2:max(output.injwell.segnr);
            waitbar(i/max(input.injwell.segment))
            output.injwell.P(i,1) = output.injwell.P(i-1,1) - output.injwell.dP_hs(i-1,1) + ...
                                    output.injwell.dP_f(i-1,1);   % pressure pipe [bar]
            output.injwell.h(i,1) = output.injwell.h(i-1,1) - output.injwell.dQ(i-1,1) - ...
                                    output.injwell.dE_pot(i-1,1); % enthalpy [J/kg]
            if output.injwell.P(i,1) < 1 % [bar]
                disp('ERROR: Pressure loss in injection well too high. ACTION: Increase mass flow, redesign
injection well or decrease II')
                close(h)
                msgbox('Pressure loss in injection well too high. ACTION: Increase mass flow, redesign
injection well or decrease II', 'Error','error');
                stat = status.FAILURE; return;
            end;

            % Import Geothermal fluid properties
            [geofprops,T_new] = fCalc_geofprops2 (output.injwell.P(i,1), output.injwell.T(i-1,1), ...
                                output.injwell.w_NaCl, output.injwell.w_CO2, data.H2O_sat, ...
                                output.injwell.h(i,1), output, output.injwell.h(i-1,1), input, data);

            output.injwell.T(i,1)      = T_new;            % temperature [C]
            output.injwell.chi(i,1)    = geofprops(1,2);   % gas mass fraction [-]
            output.injwell.v_spec(i,1) = 1/geofprops(1,4); % specific volume [m3/kg]
            output.injwell.rho(i,1)    = geofprops(1,4);   % density [kg/m3]
            output.injwell.c_p(i,1)    = geofprops(1,5);   % heat capacity [J/kg/K]
            output.injwell.mu(i,1)     = geofprops(1,7);   % viscosity [Pa*s]
            output.injwell.eps_G(i,1)  = geofprops(1,3);   % void fraction [-]

            output.injwell.u(i,1)  = fCalc_u(input.general.m_gf, output.injwell.rho(i,1), ...
                                     output.injwell.D_i(i,1));                     % velocity [m/s]
            output.injwell.Re(i,1) = fCalc_Re(output.injwell.D_i(i,1), output.injwell.rho(i,1), ...
                                     output.injwell.u(i,1), output.injwell.mu(i,1)); % Reynolds number
            output.injwell.f(i,1)  = fCalc_f(output.injwell.chi(i,1), output.injwell.eps_pipe(i,1),...
                                     output.injwell.D_i(i,1), output.injwell.Re(i,1));
                                                                                   % friction factor
            output.injwell.T_g     = fCalc_T_g(input.general.T_surf_r, output.injwell.grad_T_g, ...
                                     output.injwell.tvd);                % Geothermal temperature [C]
            output.injwell.dQ(i,1) = fCalc_dQ(output.injwell.T(i,1), output.injwell.T_g(i,1), ...
                                     output.injwell.D_i(i,1), output.injwell.dl(i,1), ...
```

```
                                input.general.m_gf, input.general.gamma, input.general.t, ...
                                output.injwell.k_r(i,1), output.injwell.alfa_r(i,1));
                                                        % Heat exchange with surroundings[J/kg]
            output.injwell.dE_pot(i,1) = fCalc_dE_pot(input.general.g, output.injwell.dz(i,1));
                                                        % potential energy change [J/kg]
            output.injwell.dP_f(i,1)   = fCalc_dP_f(output.injwell.D_i(i,1), output.injwell.f(i,1),...
                                         output.injwell.rho(i,1), output.injwell.u(i,1), ...
                                         output.injwell.dl(i,1));  % frictional pressure change [bar]
            output.injwell.dP_hs(i,1)  = fCalc_dP_hs(input.general.g, output.injwell.rho(i,1), ...
                                         output.injwell.dz(i,1));  % hydrostatic pressure change [bar]
        end
        close(h)

        if abs(output.injwell.T(end) - output.SF.T_9(index,1)) < error_T_9_10
            break
        end

        l = l + 1; % iteration number

        output.injwell.T(1,1) = output.injwell.T(1,1) - (output.injwell.T(end) - ...
                                output.SF.T_9(index,1));
                                        % recalculate bottomhole temperature injection well
    end

    %% Single-flash power plant simulation
    output.injwell.index  = index;                                  % row number with maximum W_net
    [input, output, stat] = fCalc_SF(input, output, status, data, 2); % 2 is algorithm number

    figure
    plot(output.injwell.T,-input.injwell.tvd)
    title('Temperature Injection well')
    xlabel('T[Celsius]') % x-axis label
    ylabel('Tvd[m]') % y-axis label

    % Succesfull simulation --> This is output from fCalc_SF [input, output, stat]
    % stat = status.SUCCES;
end
```

```
%% fCalc_injwell_BC

% Simulation of the injection well of the binary cycle power plant
% Frank Niewold
% Released version 1.0, February 2017

function [input, output, stat, geofprops] = fCalc_injwell_BC(input, output, data, status)

    stat = status.SUCCES;

    % settings
    error_T_9_10 = input.settings.error_T_9_10; % Same value as in single-flash plant

    %% Write injection well dimensions to output file
    for i = 1:max(input.injwell.segment);
        output.injwell_BC.segnr (i,1)   = input.injwell.segment(i,1);  % segment nr.
        output.injwell_BC.D_i (i,1)     = input.injwell.D_i(i,1);      % inner diameter [m]
        output.injwell_BC.dl (i,1)      = input.injwell.dl(i,1);       % length [m]
        output.injwell_BC.dz(i,1)       = input.injwell.dz(i,1);       % height [m]
        output.injwell_BC.tvd(i,1)      = input.injwell.tvd(i,1);      % true vertical depth tvd [m]
        output.injwell_BC.grad_T_g(i,1) = input.injwell.grad_T_g(i,1); % geothermal temp grad [K]
        output.injwell_BC.eps_pipe(i,1) = input.injwell.eps_pipe(i,1); % absolute pipe roughness [m]
        output.injwell_BC.k_r(i,1)      = input.injwell.k_r(i,1);      % rock thermal cond.[W/m/K]
        output.injwell_BC.alfa_r(i,1)   = input.injwell.alfa_r(i,1);   % rock thermal diff.[m2/s]
    end

    %% Import initial brine properties at bottom injection well from reservoir
    % Import composition from power plant
    output.injwell_BC.P(1,1) = output.reservoir.geofprops(1,1); % pressure [bar]
    [~,index]                = max(output.BC.W_net);            % row number with maximum W_net
    output.injwell_BC.T(1,1) = output.BC.T_C;                   % temperature [C]
    output.injwell_BC.T(2,1) = output.injwell_BC.T(1,1) + 2;    % Initial value for while loop
    output.injwell_BC.w_NaCl = output.BC.w_NaCl_A;              % mass fraction NaCl injection well
    output.injwell_BC.w_CO2  = output.BC.w_CO2_A;               % mass fraction CO2 injection well

    l = 1; % iteration number

    %% Iterative procedure for calculation of injection well properties
    while abs(output.injwell_BC.T(end) - output.BC.T_A) > error_T_9_10;
        % geothermal fluid properties @ bottom of injection well
        [geofprops] = fCalc_geofprops1(output.injwell_BC.P(1,1), output.injwell_BC.T(1,1), ...
                      output.injwell_BC.w_NaCl, output.injwell_BC.w_CO2, output);
        output.injwell_BC.geofprops(1,1:5) = [output.injwell_BC.P(1,1) output.injwell_BC.T(1,1) ...
                                              output.injwell_BC.w_NaCl output.injwell_BC.w_CO2 ...
                                              (1 - output.injwell_BC.w_NaCl -output.injwell_BC.w_CO2)];
        output.injwell_BC.geofprops(1,6:31) = geofprops(1,1:26);

        output.injwell_BC.h(1,1)       = output.injwell_BC.geofprops(1,11);  % enthalpy [J/kg]
        output.injwell_BC.chi(1,1)     = output.injwell_BC.geofprops(1,7);   % gas mass fraction [-]
        output.injwell_BC.v_spec(1,1) = 1/output.injwell_BC.geofprops(1,9); % specific volume [m3/kg]
        output.injwell_BC.rho(1,1)     = output.injwell_BC.geofprops(1,9);   % density [kg/m3]
        output.injwell_BC.c_p(1,1)     = output.injwell_BC.geofprops(1,10);  % heat capacity [J/kg/K]
        output.injwell_BC.mu(1,1)      = output.injwell_BC.geofprops(1,12);  % viscosity [Pa*s]
        output.injwell_BC.eps_G(1,1)   = output.injwell_BC.geofprops(1,8);   % void fraction [-]

        % Calculate initial properties at bottom injection well
        output.injwell_BC.u(1,1) = fCalc_u(input.general.m_gf, output.injwell_BC.rho(1,1), ...
                                   output.injwell_BC.D_i(1,1));                    % velocity [m/s]
        output.injwell_BC.Re(1,1) = fCalc_Re(output.injwell_BC.D_i(1,1),output.injwell_BC.rho(1,1),...
                                    output.injwell_BC.u(1,1), output.injwell_BC.mu(1,1));
                                                                                % Reynolds number
        output.injwell_BC.f(1,1) = fCalc_f(output.injwell_BC.chi(1,1), ...
                                   output.injwell_BC.eps_pipe(1,1), output.injwell_BC.D_i(1,1), ...
                                   output.injwell_BC.Re(1,1));                  % friction factor
        output.injwell_BC.T_g = fCalc_T_g(input.general.T_surf_r, output.injwell_BC.grad_T_g, ...
                                output.injwell_BC.tvd);                 % Geothermal temperature [C]
        output.injwell_BC.dQ(1,1) = fCalc_dQ(output.injwell_BC.T(1,1), output.injwell_BC.T_g(1,1), ...
                                    output.injwell_BC.D_i(1,1), output.injwell_BC.dl(1,1), ...
                                    input.general.m_gf, input.general.gamma, input.general.t, ...
                                    output.injwell_BC.k_r(1,1), output.injwell_BC.alfa_r(1,1));
                                                        % heat exchange with surroundings[J/kg]
        output.injwell_BC.dE_pot(1,1) = fCalc_dE_pot(input.general.g, output.injwell_BC.dz(1,1));
                                                        % potential energy change [J/kg]
        output.injwell_BC.dP_f(1,1)   = fCalc_dP_f(output.injwell_BC.D_i (1,1), ...
                                        output.injwell_BC.f(1,1), ...
                                        output.injwell_BC.rho(1,1), output.injwell_BC.u(1,1), ...
                                        output.injwell_BC.dl(1,1)); % frictional pressure change [bar]
        output.injwell_BC.dP_hs(1,1) = fCalc_dP_hs(input.general.g, output.injwell_BC.rho(1,1), ...
                                       output.injwell_BC.dz(1,1)); % hydrostatic pressure change [bar]

        formatSpec = ...
            'BC injection well calculation (iteration #%d).\ndT > %d. Iterate until dT < 1.\nPlease wait...';
        A1  = l;
        A2  = floor(output.injwell_BC.T(end) - output.BC.T_C);
        str = sprintf(formatSpec,A1,A2);
        h   = waitbar(0,str);

        %% Calculation of segment nr.2 to top of the injection well
        for i = 2:max(output.injwell_BC.segnr);
            waitbar(i/max(input.injwell.segment))
            output.injwell_BC.P(i,1) = output.injwell_BC.P(i-1,1) - output.injwell_BC.dP_hs(i-1,1) + ...
                                       output.injwell_BC.dP_f(i-1,1);   % pressure pipe [bar]
            output.injwell_BC.h(i,1) = output.injwell_BC.h(i-1,1) - output.injwell_BC.dQ(i-1,1) - ...
                                       output.injwell_BC.dE_pot(i-1,1); % enthalpy [J/kg]
            if output.injwell_BC.P(i,1) < 1 % [bar]
                disp('ERROR: Pressure loss in BC injection well too high. ACTION: Increase mass flow, redesign
injection well or decrease II')
                close(h)
                msgbox('Pressure loss in BC injection well too high. ACTION: Increase mass flow, redesign
injection well or decrease II', 'Error','error');
                stat = status.FAILURE; return;
            end;

            % Import Geothermal fluid properties
            [geofprops,T_new] = fCalc_geofprops2 (output.injwell_BC.P(i,1), ...
                                output.injwell_BC.T(i-1,1), output.injwell_BC.w_NaCl, ...
                                output.injwell_BC.w_CO2, data.H2O_sat, ...
                                output.injwell_BC.h(i,1), output, output.injwell_BC.h(i-1,1), ...
                                input, data);

            output.injwell_BC.T(i,1)       = T_new;             % temperature [C]
            output.injwell_BC.chi(i,1)     = geofprops(1,2);    % gas mass fraction [-]
            output.injwell_BC.v_spec(i,1) = 1/geofprops(1,4); % specific volume [m3/kg]
            output.injwell_BC.rho(i,1)     = geofprops(1,4);    % density [kg/m3]
            output.injwell_BC.c_p(i,1)     = geofprops(1,5);    % heat capacity [J/kg/K]
            output.injwell_BC.mu(i,1)      = geofprops(1,7);    % viscosity [Pa*s]
            output.injwell_BC.eps_G(i,1)   = geofprops(1,3);    % void fraction [-]

            output.injwell_BC.u(i,1)   = fCalc_u(input.general.m_gf, output.injwell_BC.rho(i,1), ...
```

```
                                output.injwell_BC.D_i(i,1));                    % velocity [m/s]
        output.injwell_BC.Re(i,1) = fCalc_Re(output.injwell_BC.D_i(i,1), ...
                                    output.injwell_BC.rho(i,1),  output.injwell_BC.u(i,1),...
                                    output.injwell_BC.mu(i,1)); % Reynolds number
        output.injwell_BC.f(i,1)  = fCalc_f(output.injwell_BC.chi(i,1), ...
                                    output.injwell_BC.eps_pipe(i,1),...
                                    output.injwell_BC.D_i(i,1), output.injwell_BC.Re(i,1));
                                                                    % friction factor
        output.injwell_BC.T_g     = fCalc_T_g(input.general.T_surf_r, ...
                                    output.injwell_BC.grad_T_g, ...
                                    output.injwell_BC.tvd);          % Geothermal temperature [C]
        output.injwell_BC.dQ(i,1) = fCalc_dQ(output.injwell_BC.T(i,1), ...
                                    output.injwell_BC.T_g(i,1), ...
                                    output.injwell_BC.D_i(i,1), output.injwell_BC.dl(i,1), ...
                                    input.general.m_gf, input.general.gamma, input.general.t, ...
                                    output.injwell_BC.k_r(i,1), output.injwell_BC.alfa_r(i,1));
                                                        % Heat exchange with surroundings[J/kg]
        output.injwell_BC.dE_pot(i,1) = fCalc_dE_pot(input.general.g, output.injwell_BC.dz(i,1));
                                                        % potential energy change [J/kg]
        output.injwell_BC.dP_f(i,1)   = fCalc_dP_f(output.injwell_BC.D_i(i,1), ...
                                        output.injwell_BC.f(i,1),...
                                        output.injwell_BC.rho(i,1), output.injwell_BC.u(i,1), ...
                                        output.injwell_BC.dl(i,1));  % frictional P change [bar]
        output.injwell_BC.dP_hs(i,1)  = fCalc_dP_hs(input.general.g,output.injwell_BC.rho(i,1),...
                                        output.injwell_BC.dz(i,1));  % hydrostatic P change [bar]
    end
    close(h)

    if abs(output.injwell_BC.T(end) - output.BC.T_C) < error_T_9_10
        break
    end

    l = l + 1; % iteration number

    output.injwell_BC.T(1,1) = output.injwell_BC.T(1,1) - (output.injwell_BC.T(end) - ...
                               output.BC.T_C);
                                        % recalculate bottomhole temperature injection well
end

%% Single-flash power plant simulation
output.injwell_BC.index  = index;                                   % row number with maximum W_net
[input, output, stat] = fCalc_BC(input, output, status, data, 2);  % 2 is algorithm number

figure
plot(output.injwell_BC.T,-input.injwell.tvd)
title('Temperature BC Injection well')
xlabel('T[Celsius]') % x-axis label
ylabel('Tvd[m]') % y-axis label

% Succesfull simulation --> This is output from fCalc_BC [input, output, stat]
% stat = status.SUCCES;
end
```

```
%% fCreate_figures

% Plotting relevant property profiles as a function of true vertical depth
% Frank Niewold
% Released version 1.0, February 2017

function [] = fCreate_figures(input, output, status)
figure
plot(output.prodwell.P,-output.prodwell.tvd)
xlabel('P [bar]') % x-axis label
ylabel('Tvd [m]') % y-axis label

hold on
plot(output.prodwell_GL.P,-output.prodwell_GL.tvd)
xlabel('P[bar]') % x-axis label
ylabel('Tvd[m]') % y-axis label

figure
plot(output.prodwell.T,-output.prodwell.tvd)
xlabel('T[Celsius]') % x-axis label
ylabel('Tvd[m]') % y-axis label

hold on
plot(output.prodwell_GL.T,-output.prodwell_GL.tvd)
xlabel('P[bar]') % x-axis label
ylabel('Tvd[m]') % y-axis label

figure
plot(output.prodwell.chi,-output.prodwell.tvd)
xlabel('\chi[-]') % x-axis label
ylabel('Tvd[m]') % y-axis label

hold on
plot(output.prodwell_GL.chi,-output.prodwell_GL.tvd)
xlabel('\chi[-]') % x-axis label
ylabel('Tvd[m]') % y-axis label

figure
plot(output.prodwell.h,-output.prodwell.tvd)
xlabel('h[J/kg]') % x-axis label
ylabel('Tvd[m]') % y-axis label

hold on
plot(output.prodwell_GL.h,-output.prodwell_GL.tvd)
xlabel('h[J/kg]') % x-axis label
ylabel('Tvd[m]') % y-axis label

figure
plot(output.prodwell.eps_G(:,1),-output.prodwell.tvd)
xlabel('\epsilon_G[-]') % x-axis label
ylabel('Tvd[m]') % y-axis label

hold on
plot(output.prodwell_GL.eps_G(:,1),-output.prodwell_GL.tvd)
xlabel('\epsilon_G[-]') % x-axis label
ylabel('Tvd[m]') % y-axis label

end
```

```
%% fCalc_u

% Calculate velocity [m/s]
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_u(m_gf, rho, D_i)

output = (m_gf/rho)/((pi()/4)*D_i^2);

end
```

```
%% fCalc_Re

% Calculation of Reynolds number
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_Re(D_i, rho, u, mu)

output = D_i*rho*u/mu;

end
```

```
%% fCalc_f

% Calculation of friction factor
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_f(chi, eps_pipe, D_i, Re)

if chi == 0
    %Swamee-Jain equation for liquid flow only
    output = 0.25/((log10(eps_pipe/D_i/3.7)+(5.74/Re^0.9))^2);
else
    %Hasan et al.(2002) two phase flow from Chen (1979) correlation
    output = 0.25/(log10((eps_pipe/D_i/3.7065)-
((5.0452/Re)*log10(((1/2.8257)*(eps_pipe/D_i)^1.1098)+(5.8506/Re^0.8981))))^2);
end
end
```

```
%% fCalc_T_g

% Calculation of geothermal temperature [C]
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_T_g(T_bh, geo_grad, tvd)

j = size(geo_grad,1);
output = zeros(j,1);
T = T_bh;
output(1,1) = T_bh;
for i = 2:j
    output(i,1) = T + geo_grad(i,1) * (tvd(i,1) - tvd(i-1,1));
    T = output(i,1);
end
```

```
%% fCalc_dQ

% Calculation of heat flow to surrounding rocks [J/kg]
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_dQ(T_gf, T_g, D_i, dl, m_gf, gamma, t, k_r, alfa_r)

output = ((4*k_r*pi()*(T_gf-T_g))/log((4*alfa_r*t)/(gamma*(D_i/2)^2)))*dl/m_gf; % [J/kg]

end
```

```
%% fCalc_dE_pot

% Calculation of potential energy change [J/kg]
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_dE_pot(g, dz)

output = g * dz; % [J/kg]

end
```

```
%% fCalc_dP_f

% Calculation of frictional pressure change [bar]
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_dP_f(D_i, f, rho, u, dl)

output = ((1/2) * f * rho * u^2 * dl / D_i)/100000; % [bar]

end
```

```
%% fCalc_dP_hs

% Calculation of hydrostatic pressure change [bar]
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_dP_hs(g, rho, dz)

output = g * rho * dz / 100000; % [bar]

end
```

```
%% fCalc_dP_k

% Calculation of kinetic pressure change [bar]
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_dP_k(rho, u_2, u_1)

output = (rho*(u_2^2-u_1^2))/100000; % [bar]

end
```

```
%% fCalc_dE_k

% Calculation of potential energy change [J/kg]
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_dE_k(u_2, u_1)

output = 0.5 * (u_2^2 - u_1^2); % [J/kg]

end
```

```
%% fCalc_prodwell_virtual

% Simulation of a virtual production well
% If the Francke Model did not experience flashing, while according to Duan and Sun (2003) the
% pressure is below the degassing pressure. The production well is virtually extended above the
% earth's surface in order to obtain the quality in the real production well.
% Frank Niewold
% Released version 1.0, February 2017

function [input, output, geofprops, i] = fCalc_prodwell_virtual(input, output, data, k)

    %% Production well simulation from segment 2 to top
    j = k + 1;                          % count further
    k = 1000;                           % last segment number

    formatSpec = 'Production well virtual calculation.\nPlease wait...';
    str        = sprintf(formatSpec);
    h1         = waitbar(0,str);

    % Calculate segments until two segments have a chi > 0 according to the Francke Model

    for i = j:k

        % Create extra output for virtual production well
        output.prodwell.segnr(i,1)    = output.prodwell.segnr(i-1,1) + 1; % segment nr.
        output.prodwell.D_i(i,1)      = output.prodwell.D_i(i-1,1);       % inner diameter wellbore[m]
        output.prodwell.dl(i,1)       = output.prodwell.dl(i-1,1);        % length [m]
        output.prodwell.dz(i,1)       = output.prodwell.dz(i-1,1);        % dz [m]
        output.prodwell.tvd(i,1)      = output.prodwell.tvd(i-1,1) - output.prodwell.dz(i,1);
                                                                          % true vertical depth tvd[m]
        output.prodwell.grad_T_g(i,1) = output.prodwell.grad_T_g(i-1,1);  % temperature gradient [m]
        output.prodwell.eps_pipe(i,1) = output.prodwell.eps_pipe(i-1,1);  % absolute pipe roughness[m]
        output.prodwell.k_r(i,1)      = output.prodwell.k_r(i-1,1);       % rock thermal condu.[W/m/K]
        output.prodwell.alfa_r(i,1)   = output.prodwell.alfa_r(i-1,1);    % rock thermal diffus.[m2/s]
        output.prodwell.l(i,1) = output.prodwell.l(i-1,1) + output.prodwell.dl(i-1,1);

        output.prodwell.T_g(i,1) = fCalc_T_g(input.general.T_surf_r, output.prodwell.grad_T_g(i,1),...
                                   output.prodwell.tvd(i,1));             % Geothermal temperature [C]

        output.prodwell.P(i,1) = output.prodwell.P(i-1,1) - output.prodwell.dP_hs(i-1,1) - ...
            output.prodwell.dP_f(i-1,1);                                  % pressure pipe [bar]
        output.prodwell.h(i,1) = output.prodwell.h(i-1,1) - output.prodwell.dQ(i-1,1) - ...
            output.prodwell.dE_pot(i-1,1);                                % enthalpy [J/kg]
        if output.prodwell.P(i,1) < 1
            disp('ERROR: Pressure loss in wellbore too high. ACTION: Decrease mass flow')
            close(h)
            msgbox('Pressure loss in wellbore too high. ACTION: Decrease mass flow', 'Error','error');
            stat = status.FAILURE; return;
        end;

        [geofprops, T_new, w_table] = fCalc_geofprops2 (output.prodwell.P(i,1), ...
                                      output.prodwell.T(i-1,1), input.general.w_NaCl, ...
                                      input.general.w_CO2, data.H2O_sat,output.prodwell.h(i,1),...
                                      output, output.prodwell.h(i-1,1), input, data);
        output.prodwell.T(i,1)      = T_new;           % temperature [C}
        output.prodwell.chi(i,1)    = geofprops(1,2);  % gas mass fraction [-}
        output.prodwell.v_spec(i,1) = 1/geofprops(1,4); % specific volume {m3/kg]
        output.prodwell.rho(i,1)    = geofprops(1,4);  % density [kg/m3]
        output.prodwell.c_p(i,1)    = geofprops(1,5);  % specific heat capacity [J/kg/K]
```

```
output.prodwell.mu(i,1)     = geofprops(1,7);   % viscosity [Pa*s]
output.prodwell.eps_G(i,1)  = geofprops(1,3);   % void fraction [-]

% Drift flux model
if output.prodwell.chi(i,1) > 0 && input.prodwell.DF_model > 1
    % quality larger than zero && DF_model = 1 --> homogeneous
    output.prodwell.rho_l(i,1) = geofprops(1,15);  % density liquid phase [kg/m3]
    output.prodwell.rho_v(i,1) = geofprops(1,23);  % density vapor phase [kg/m3]
    output.prodwell.mu_l(i,1)  = geofprops(1,18);  % viscosity liquid phase [Pa*s]
    output.prodwell.mu_v(i,1)  = geofprops(1,26);  % viscosity vapor phase [Pa*s]
    output.prodwell.l_E(i,1)   = output.prodwell.l(i,1) - output.prodwell.l(i,1);
    % length from entrance or flash horizon [m]
    output.prodwell.u_sg(i,1)  = ((output.prodwell.chi(i,1) * input.general.m_gf)/...
                                  geofprops(1,23))/(pi*(output.prodwell.D_i(i,1)/2)^2);
    % superficial gas velocity [m/s]
    output.prodwell.u_sl(i,1)  = (((1-output.prodwell.chi(i,1)) * input.general.m_gf)/...
                                  geofprops(1,15))/(pi*(output.prodwell.D_i(i,1)/2)^2);
    % superficial liquid velocity [m/s]
    [eps_G, FP, u_gu, C_0] = fCalc_eps_G(output.prodwell.T(i,1), geofprops(1,15), ...
                             geofprops(1,23), geofprops(1,18), geofprops(1,26), ...
                             output.prodwell.l_E(i,1), output.prodwell.D_i(i,1), ...
                             output.prodwell.eps_pipe(i,1), output.prodwell.u_sg(i,1), ...
                             output.prodwell.u_sl(i,1), input.general.g, ...
                             output.prodwell.chi(i,1), input.prodwell.DF_model);
    output.prodwell.eps_G(i,1) = eps_G;                                % void fraction
    output.prodwell.FP(i,1)    = cellstr(FP);                          % flow pattern
    output.prodwell.rho(i,1)   = output.prodwell.rho_v(i,1)*output.prodwell.eps_G(i,1)...
                                 + output.prodwell.rho_l(i,1)*...
                                 (1-output.prodwell.eps_G(i,1));       % density [kg/m3]
    output.prodwell.u_gu(i,1)  = u_gu; % drift-flux velocity, u_gas relative to u_m
    output.prodwell.C_0(i,1)   = C_0;  % distribution parameter
end

% Output geothermal fluid composition - mass fractions
output.prodwell.w_NaCl_l(i,1) = w_table(3,2);
output.prodwell.w_CO2_l(i,1)  = w_table(3,3);
output.prodwell.w_CO2_v(i,1)  = w_table(3,4);
output.prodwell.w_H2O_l(i,1)  = w_table(3,5);
output.prodwell.w_H2O_v(i,1)  = w_table(3,6);

% Not used for now - mass fraction at transition
output.prodwell.w_NaCl_l_t(i,1) = w_table(1,2);
output.prodwell.w_CO2_l_t(i,1)  = w_table(1,3);
output.prodwell.w_CO2_v_t(i,1)  = w_table(1,4);
output.prodwell.w_H2O_l_t(i,1)  = w_table(1,5);
output.prodwell.w_H2O_v_t(i,1)  = w_table(1,6);

% Calculate segment properties
output.prodwell.u(i,1)      = fCalc_u(input.general.m_gf, output.prodwell.rho(i,1), ...
                              output.prodwell.D_i(i,1));                % velocity [m/s]
output.prodwell.Re(i,1)     = fCalc_Re(output.prodwell.D_i(i,1), ...
                              output.prodwell.rho(i,1), output.prodwell.u(i,1), ...
                              output.prodwell.mu(i,1));              % Reynolds number [-]
output.prodwell.f(i,1)      = fCalc_f(output.prodwell.chi(i,1), ...
                              output.prodwell.eps_pipe(i,1), output.prodwell.D_i(i,1),...
                              output.prodwell.Re(i,1));              % friction factor [-]
output.prodwell.dQ(i,1)     = fCalc_dQ(output.prodwell.T(i,1), ...
                              output.prodwell.T_g(i,1), output.prodwell.D_i(i,1), ...
                              output.prodwell.dl(i,1), input.general.m_gf, ...
                              input.general.gamma, input.general.t, ...
                              output.prodwell.k_r(i,1), output.prodwell.alfa_r(i,1));
                                                  % heat exchange with surroundings [J/kg]
output.prodwell.dE_pot(i,1) = fCalc_dE_pot(input.general.g, output.prodwell.dz(i,1));
                                                                % potential energy [J/kg]
output.prodwell.dP_f(i,1)   = fCalc_dP_f(output.prodwell.D_i(i,1), ...
                              output.prodwell.f(i,1), output.prodwell.rho(i,1), ...
                              output.prodwell.u(i,1), output.prodwell.dl(i,1));
                                                       % frictional pressure change [bar]
output.prodwell.dP_hs(i,1)  = fCalc_dP_hs(input.general.g, output.prodwell.rho(i,1), ...
                              output.prodwell.dz(i,1));% hydrostatic pressure change [bar]

% if i == size(input.prodwell.tvd,1)
%     output.prodwell.P(i+1,1) = output.prodwell.P(i,1) - output.prodwell.dP_hs(i,1) - ...
%                                output.prodwell.dP_f(i,1); % pressure pipe [bar]
%     output.prodwell.h(i+1,1) = output.prodwell.h(i,1) - output.prodwell.Q(i,1) - ...
%                                output.prodwell.E_pot(i,1); %enthalpy [J/kg]
% end

output.prodwell.P_degas(i,1) = interp3(data.m_NaCl_degas, data.T_degas, data.m_CO2_degas, ...
                               data.P_degas, input.general.m_NaCl, output.prodwell.T(i,1) +...
                               273.15, input.general.m_CO2);        % degassing pressure [bar]

        waitbar(geofprops(1,1)/output.prodwell.P(i,1))

        %% Check if two segments have a significant gas mass fraction
        if output.prodwell.chi(i,1) > 0.0001 && output.prodwell.chi(i-1,1) > 0.0001
            output.prodwell.P_old = output.prodwell.P(i-2,1);
            break % start interpolation from P_degas Duan and Sun (2003)
        end

    end

    output.prodwell.P_degas = interp3(data.m_NaCl_degas, data.T_degas, data.m_CO2_degas, ...
                              data.P_degas, input.general.m_NaCl, output.prodwell.T + 273.15,...
                              input.general.m_CO2);                 % degassing pressure [bar]
    close(h1)
end
```

```
%% fCalc_dQgf

% Calculation of heat flow to surrounding rocks [J/kg]
% Frank Niewold
% Released version 1.0, February 2017

function [output] = fCalc_dQgf(output,input,i,j)

load P_CO2; load T_CO2; load cp_CO2; load h_CO2; load k_CO2; load mu_CO2; load rho_CO2; load s_CO2;
load H2O_sat_props;

%% heat transfer coefficient annulus side
D_Wo = output.prodwell_GL.D_i(i,1) + 0.02;
D_ao = D_Wo + 0.05;
L_E = 1000;
k_Wc = 50;
a = D_Wo/D_ao;
F_a = 0.75*a^-0.17 + (0.9 - 0.15 *a^0.6)/(1+a);
D_h = D_ao - D_Wo;
A = pi * (D_ao^2 - D_Wo^2)/4;

output.prodwell_GL.rho_GL(i,j) = interp2(P_CO2,T_CO2,rho_CO2,output.prodwell_GL.P_GL(i,1),...
                                 output.prodwell_GL.T_GL(i,1));
output.prodwell_GL.mu_GL(i,j)  = interp2(P_CO2,T_CO2,mu_CO2,output.prodwell_GL.P_GL(i,1),...
                                 output.prodwell_GL.T_GL(i,1));
output.prodwell_GL.cp_GL(i,j)  = interp2(P_CO2,T_CO2,cp_CO2,output.prodwell_GL.P_GL(i,1),...
                                 output.prodwell_GL.T_GL(i,1));
output.prodwell_GL.k_GL(i,j)   = interp2(P_CO2,T_CO2,k_CO2,output.prodwell_GL.P_GL(i,1),...
                                 output.prodwell_GL.T_GL(i,1));

output.prodwell_GL.Re_GL(i,j) = output.prodwell_GL.m_GL * D_h/(A * output.prodwell_GL.mu_GL(i,j));
output.prodwell_GL.Pr_GL(i,j) = output.prodwell_GL.cp_GL(i,j) * output.prodwell_GL.mu_GL(i,j)...
                                /output.prodwell_GL.k_GL(i,j);

k_1 = 1.07 +(900/output.prodwell_GL.Re_GL(i,j)) - (0.63/(1+10*output.prodwell_GL.Pr_GL(i,j)));
Re_star = output.prodwell_GL.Re_GL(i,j) * (((1 + a^2) * log(a) + (1 - a^2))/((1-a)^2 *log(a)));
f_a = (1.8 * log10(Re_star) - 1.5)^-2;
output.prodwell_GL.Nu_GL(i,j) = (((f_a/8)*output.prodwell_GL.Re_GL(i,j)*...
                                 output.prodwell_GL.Pr_GL(i,j))/(k_1 + 12.7 * (f_a/8)^0.5 *...
                                 (output.prodwell_GL.Pr_GL(i,j)^(2/3)-1)))...
                                 * (1 + (D_h/L_E)^(2/3)) * F_a;
output.prodwell_GL.h_c_Wo(i,j) = output.prodwell_GL.Nu_GL(i,j) * output.prodwell_GL.k_GL(i,j)/D_h;
term_1 = 1/(output.prodwell_GL.h_c_Wo(i,j)*2*pi*(D_Wo/2)*output.prodwell_GL.dl(i,1));
%% heat transfer casing
term_2 = log(D_Wo/output.prodwell_GL.D_i(i,1))/(2*pi*k_Wc*output.prodwell_GL.dl(i,1));

%% heat transfer geothermal fluid side
output.prodwell_GL.X_tt(i,j) = (output.prodwell_GL.chi(i,1)/(1-output.prodwell_GL.chi(i,1)))^0.9...
                               *(output.prodwell_GL.rho_l(i,1)/output.prodwell_GL.rho_v(i,1))^...
                               0.5*(output.prodwell_GL.mu_v(i,1)/output.prodwell_GL.mu_l(i,1))^0.1;
output.prodwell_GL.F_c(i,j) = 8E-05*output.prodwell_GL.X_tt(i,j)^3 - ...
                              0.0133*output.prodwell_GL.X_tt(i,j)^2 + ...
                              1.2623*output.prodwell_GL.X_tt(i,j) + 1.4214;

output.prodwell_GL.k_gf_l(i,j) = interp1(H2O_sat_props(:,2),H2O_sat_props(:,8),...
                                 output.prodwell_GL.T(i,1));
output.prodwell_GL.k_gf_g(i,j) = interp1(H2O_sat_props(:,2),H2O_sat_props(:,9),...
                                 output.prodwell_GL.T(i,1));
output.prodwell_GL.k_gf(i,j)   = output.prodwell_GL.chi(i,1) * output.prodwell_GL.k_gf_g(i,j) + ...
```

```
                                (1 - output.prodwell_GL.chi(i,1)) * output.prodwell_GL.k_gf_l(i,j);
output.prodwell_GL.Pr_gf(i,j)  = output.prodwell_GL.c_p(i,1) * output.prodwell_GL.mu(i,1)/ ...
                                 output.prodwell_GL.k_gf(i,j);
f_a = (1.8 * log10(output.prodwell_GL.Re(i,1) - 1.5))^-2;
output.prodwell_GL.Nu_gf(i,j)  = (((f_a/8)*output.prodwell_GL.Re(i,1)*...
                                 output.prodwell_GL.Pr_gf(i,j))/(1 + 12.7 * (f_a/8)^0.5 *...
                                 (output.prodwell_GL.Pr_gf(i,j)^(2/3)-1)))...
                                 * (1 + (output.prodwell_GL.D_i(i,1)/L_E)^(2/3)) * F_a;
output.prodwell_GL.h_fc(i,j)   = output.prodwell_GL.Nu_gf(i,j)*output.prodwell_GL.k_gf(i,j)/...
                                 output.prodwell_GL.D_i(i,1);
output.prodwell_GL.h_c_Wi(i,j) = output.prodwell_GL.h_fc(i,j)*output.prodwell_GL.F_c(i,j);
term_3 = 1/(output.prodwell_GL.h_c_Wi(i,j)*2*pi*(output.prodwell_GL.D_i(i,1)/2)*...
         output.prodwell_GL.dl(i,1));

output.prodwell_GL.UA(i,j) = 1/(term_1+term_2+term_3);

output.prodwell_GL.dQ_gf_gf(i,j) = output.prodwell_GL.UA(i,j) * (output.prodwell_GL.T_GL(i+1,j) - ...
                                   output.prodwell_GL.T(i,1))/output.prodwell_GL.m_gf(i,1);
output.prodwell_GL.dQ_gf_GL(i,j) = output.prodwell_GL.UA(i,j) * (output.prodwell_GL.T(i,1) - ...
                                   output.prodwell_GL.T_GL(i+1,j))/output.prodwell_GL.m_GL;

end
```

```
%% fCalc_T_s_com

% Iteration of temperature to calculate inlet properties compressor of production well with gas-lift.
% Frank Niewold
% Released version 1.0, February 2017

function [F] = fCalc_T_s_com(x,y)

load h_CO2; load P_CO2; load s_CO2; load T_CO2;

if y(3) == 1
    T     = x(1);    % temperature @ state 2s
    P_GL  = y(1);    % partial pressure H2O @ state 2
    s_GL  = y(2);    % partial pressure CO2 @ state 2

    s_CO2_check = interp2(P_CO2,T_CO2,s_CO2,P_GL,T);

    F = s_GL - s_CO2_check;
else
    T     = x(1);    % temperature @ state 2s
    P_GL  = y(1);    % partial pressure H2O @ state 2
    h_GL  = y(2);    % partial pressure CO2 @ state 2

    h_CO2_check = interp2(P_CO2,T_CO2,h_CO2,P_GL,T);

    F = h_GL - h_CO2_check;
end
```

```
%% fCalc_chi_5s

% Iteration of quantity and temperature to calculate outlet properties turbine @ state 5s
% Frank Niewold
% Released version 1.0, February 2017

function [F,output] = fCalc_chi_5s(x,y)

    load h_CO2; load P_CO2;load s_CO2;load T_CO2;load H2O_sat_props;

    output.SF.chi_5s     = x(1);
    output.SF.T_5s       = x(2);
    output.SF.w_CO2_g_2 = y(1);
    output.SF.P_5        = y(2);
    output.SF.s_mix_v_4 = y(3);
    M_CO2                = y(4);
    M_H2O                = y(5);

    output.SF.w_CO2_7 = output.SF.w_CO2_g_2/output.SF.chi_5s; % mass fraction CO2 saturated gas
    output.SF.w_H2O_7 = 1 - output.SF.w_CO2_7; % mass fraction H2O saturated gas
    output.SF.n_CO2_7 = (output.SF.w_CO2_7/M_CO2) / (output.SF.w_H2O_7/M_H2O + ...
                        output.SF.w_CO2_7/M_CO2); % mole fraction CO2 saturated gas
    output.SF.n_H2O_7 = (output.SF.w_H2O_7/M_H2O) / (output.SF.w_H2O_7/M_H2O + ...
                        output.SF.w_CO2_7/M_CO2); % mole fraction H2O saturated gas
    output.SF.P_H2O_7 = output.SF.P_5 * output.SF.n_H2O_7; % partial pressure H2O saturated gas
    output.SF.P_CO2_7 = output.SF.P_5 * output.SF.n_CO2_7; % partial pressure CO2 saturated gas
    output.SF.T_5s_check = interp1(H2O_sat_props(:,1), H2O_sat_props(:,2), output.SF.P_H2O_7,'spline');
                                                          % temperature expanded mixture @ state 5s

    output.SF.h_H2O_6 = interp1(H2O_sat_props(:,2), H2O_sat_props(:,3), output.SF.T_5s,'spline');
                                                                        % enthalpy H2O saturated liquid
    output.SF.s_H2O_6 = interp1(H2O_sat_props(:,2), H2O_sat_props(:,5), output.SF.T_5s,'spline');
                                                                        % entropy H2O saturated liquid
    output.SF.h_H2O_7 = interp1(H2O_sat_props(:,2), H2O_sat_props(:,4), output.SF.T_5s,'spline');
                                                                        % enthalpy H2O saturated gas
    output.SF.s_H2O_7 = interp1(H2O_sat_props(:,2), H2O_sat_props(:,6), output.SF.T_5s,'spline');
                                                                        % entropy H2O saturated gas
    output.SF.h_CO2_7 = interp2(P_CO2, T_CO2, h_CO2, output.SF.P_CO2_7, output.SF.T_5s,'spline');
                                                                        % enthalpy CO2 saturated gas
    output.SF.s_CO2_7 = interp2(P_CO2, T_CO2, s_CO2, output.SF.P_CO2_7, output.SF.T_5s,'spline');
                                                                        % entropy CO2 saturated gas
                                                                        output.SF.h_mix_7 = output.SF.h_H2O_7
* output.SF.w_H2O_7 + output.SF.h_CO2_7 * output.SF.w_CO2_7;
                                                                        % enthalpy mix saturated gas
    output.SF.s_mix_7 = output.SF.s_H2O_7 * output.SF.w_H2O_7 + output.SF.s_CO2_7 * output.SF.w_CO2_7;
                                                                        % entropy mix saturated gas
    output.SF.h_mix_5s = output.SF.h_H2O_6 * (1 - output.SF.chi_5s) + output.SF.h_mix_7 * ...
                         output.SF.chi_5s; % enthalpy mix @ state 5s
    output.SF.s_mix_5s = output.SF.s_H2O_6 * (1 - output.SF.chi_5s) + output.SF.s_mix_7 * ...
                         output.SF.chi_5s; % entropy mix @ state 5s

    % Check if condition isentropic expansion is fulfilled
    F(1) = output.SF.s_mix_v_4 - output.SF.s_mix_5s;
    % Check if temperature as function of partial pressure H2O equals temperature for conditions state 5s
    F(2) = output.SF.T_5s_check - output.SF.T_5s;

end
```

```
%% fCalc_chi_5

% Iteration of quantity and temperature to calculate outlet properties turbine @ state 5
% Frank Niewold
% Released version 1.0, February 2017

function [F,output] = fCalc_chi_5(x,y)

load h_CO2; load P_CO2;load s_CO2;load T_CO2;load H2O_sat_props;

output.SF.chi_5     = x(1);
output.SF.T_5       = x(2);
output.SF.w_CO2_g_2 = y(1);
output.SF.P_5       = y(2);
output.SF.h_mix_5   = y(3);
M_CO2               = y(4);
M_H2O               = y(5);

output.SF.w_CO2_7 = output.SF.w_CO2_g_2/output.SF.chi_5; % mass fraction CO2 saturated gas
output.SF.w_H2O_7 = 1 - output.SF.w_CO2_7; % mass fraction H2O saturated gas
output.SF.n_CO2_7 = (output.SF.w_CO2_7/M_CO2) / (output.SF.w_H2O_7/M_H2O + output.SF.w_CO2_7/M_CO2);
                                                                        % mole fraction CO2 saturated gas
output.SF.n_H2O_7 = (output.SF.w_H2O_7/M_H2O) / (output.SF.w_H2O_7/M_H2O + output.SF.w_CO2_7/M_CO2);
                                                                        % mole fraction H2O saturated gas
output.SF.P_H2O_7 = output.SF.P_5 * output.SF.n_H2O_7; % partial pressure H2O saturated gas
output.SF.P_CO2_7 = output.SF.P_5 * output.SF.n_CO2_7; % partial pressure CO2 saturated gas
output.SF.T_5_check = interp1(H2O_sat_props(:,1), H2O_sat_props(:,2), output.SF.P_H2O_7,'spline');
                                                      % temperature expanded mixture @ state 5s

output.SF.h_H2O_6 = interp1(H2O_sat_props(:,2), H2O_sat_props(:,3), output.SF.T_5,'spline');
                                                                        % enthalpy H2O saturated liquid
output.SF.s_H2O_6 = interp1(H2O_sat_props(:,2), H2O_sat_props(:,5), output.SF.T_5,'spline');
                                                                        % entropy H2O saturated liquid
output.SF.h_H2O_7 = interp1(H2O_sat_props(:,2), H2O_sat_props(:,4), output.SF.T_5,'spline');
                                                                        % enthalpy H2O saturated gas
output.SF.s_H2O_7 = interp1(H2O_sat_props(:,2), H2O_sat_props(:,6), output.SF.T_5,'spline');
                                                                        % entropy H2O saturated gas
output.SF.h_CO2_7 = interp2(P_CO2, T_CO2, h_CO2, output.SF.P_CO2_7, output.SF.T_5,'spline');
                                                                        % enthalpy CO2 saturated gas
output.SF.s_CO2_7 = interp2(P_CO2, T_CO2, s_CO2, output.SF.P_CO2_7, output.SF.T_5,'spline');
                                                                        % entropy CO2 saturated gas
output.SF.h_mix_7 = output.SF.h_H2O_7 * output.SF.w_H2O_7 + output.SF.h_CO2_7 * output.SF.w_CO2_7;
                                                                        % enthalpy mix saturated gas
output.SF.s_mix_7 = output.SF.s_H2O_7 * output.SF.w_H2O_7 + output.SF.s_CO2_7 * output.SF.w_CO2_7;
                                                                        % entropy mix saturated gas
output.SF.h_mix_5_calc = output.SF.h_H2O_6 * (1 - output.SF.chi_5) + output.SF.h_mix_7 * ...
```

```
                        output.SF.chi_5; % enthalpy mix @ state 5
output.SF.s_mix_5 = output.SF.s_H2O_6 * (1 - output.SF.chi_5) + output.SF.s_mix_7 * output.SF.chi_5;
                                                                % entropy mix @ state 5

% Check h_mix_5 (from eta_t) equals h_mix_5_calc (from mixture enthalpy)
F(1) = output.SF.h_mix_5 - output.SF.h_mix_5_calc;
% Check if temperature as function of partial pressure H2O equals temperature for conditions state 5s
F(2) = output.SF.T_5_check - output.SF.T_5;

end
```

```
%% fCalc_T_5

% Iteration of temperature to calculate outlet temperature condenser pump (8)
% Frank Niewold
% Released version 1.0, February 2017

function [F,output] = fCalc_T_8(x,y)

load T_H2O_SC;  load P_H2O_SC;   load rho_H2O_SC;

output.SF.T_8          = x(1); % temperature @ state 8
output.SF.rho_H2O_l_6 = y(1); % density liquid H2O @ state 6
output.SF.P_2          = y(2); % pressure @ state 2 = equal to states(3,4,8,9,14)

output.SF.rho_H2O_l_8 = interp2(P_H2O_SC, T_H2O_SC, rho_H2O_SC, output.SF.P_2, output.SF.T_8,'spline');

F = output.SF.rho_H2O_l_6 - output.SF.rho_H2O_l_8; % Check if incompressible assumption is satisfied

end
```

```
%% fCalc_T_12s

% Iteration of temperature to calculate outlet properties steam ejector/condenser.
% Frank Niewold
% Released version 1.0, February 2017

function [F,output] = fCalc_T_12s(x,y)

load h_CO2; load P_CO2; load s_CO2; load T_CO2; load H2O_sat_prop;
load h_H2O_SH;load s_H2O_SH; load T_H2O_SH; load P_H2O_SH;

output.SF.T_12s      = x(1);    % temperature @ state 12s
output.SF.P_H2O_12   = y(1);    % partial pressure H2O @ state 12
output.SF.P_CO2_12   = y(2);    % partial pressure CO2 @ state 12
output.SF.s_mix_v_11 = y(3);    % entropy gas mixture @ state 11
output.SF.w_CO2_v_11 = y(4);    % mass fraction CO2 in gas @ state 11
output.SF.w_H2O_v_11 = y(5);    % mass fraction H2O in gas @ state 11

    output.SF.h_H2O_v_12s = interp2(P_H2O_SH, T_H2O_SH, h_H2O_SH, output.SF.P_H2O_12, ...
                            output.SF.T_12s,'spline'); % enthalpy H2O in gas @ state 12s
    output.SF.s_H2O_v_12s = interp2(P_H2O_SH, T_H2O_SH, s_H2O_SH, output.SF.P_H2O_12, ...
                            output.SF.T_12s,'spline'); % entropy H2O in gas @ state 12s
    output.SF.h_CO2_v_12s = interp2(P_CO2, T_CO2, h_CO2, output.SF.P_CO2_12, ...
                            output.SF.T_12s,'spline'); % enthalpy CO2 in gas @ state 12s
    output.SF.s_CO2_v_12s = interp2(P_CO2, T_CO2, s_CO2, output.SF.P_CO2_12, ...
                            output.SF.T_12s,'spline'); % entropy CO2 in gas @ state 12s

    output.SF.h_mix_v_12s = output.SF.h_H2O_v_12s * output.SF.w_H2O_v_11 + output.SF.h_CO2_v_12s...
                            * output.SF.w_CO2_v_11; % enthalpy gas mixture @ state 12s
    output.SF.s_mix_v_12s = output.SF.s_H2O_v_12s * output.SF.w_H2O_v_11 + output.SF.s_CO2_v_12s...
                            * output.SF.w_CO2_v_11; % entropy gas mixture @ state 12s

F = output.SF.s_mix_v_12s - output.SF.s_mix_v_11; % Check if isentropic compression is fulfilled.

end
```

```
%% fCalc_T_14

% Iteration of temperature to calculate outlet temperature make-up pump (14)
% Frank Niewold
% Released version 1.0, February 2017

function [F,output] = fCalc_T_14(x,y)

load T_H2O_SC;  load P_H2O_SC;   load rho_H2O_SC;

output.SF.T_14          = x(1);      % temperature @ state 14
output.SF.rho_H2O_l_13  = y(1);      % density liquid H2O @ state 13
output.SF.P_2           = y(2);      % pressure @ state 2 = equal to states(3,4,8,9,14)

output.SF.rho_H2O_l_14 = interp2(P_H2O_SC, T_H2O_SC, rho_H2O_SC, output.SF.P_2, output.SF.T_14,'spline');

F = output.SF.rho_H2O_l_14 - output.SF.rho_H2O_l_13; % Check if incompressible assumption is satisfied

end
```

```
%% fCalc_h_1

% Calculation of outlet temperature evaporator/superheater
% Frank Niewold
% Released version 1.0, February 2017

function [h_1] = fCalc_h_1(x,y)

load h_C5H12_SH; load P_C5H12_SH; load T_C5H12_SH;

h_1 = x(1);
P_1 = y(1);
T_1 = y(2);

T_1_calc = interp2(h_C5H12_SH, P_C5H12_SH, T_C5H12_SH, h_1, P_1);

h_1 = T_1_calc - T_1;

end
```

```
%% fCalc_h_2s

% Calculation of enthalpy at outlet of isentropic expansion
% Frank Niewold
% Released version 1.0, February

function [h_2s] = fCalc_h_2s(x,y)

load h_C5H12_SH; load P_C5H12_SH; load s_C5H12_SH;

h_2s = x(1);
P_2 = y(1);
s_2s = y(2);

s_2s_calc = interp2(h_C5H12_SH, P_C5H12_SH, s_C5H12_SH, h_2s, P_2);

h_2s = s_2s_calc - s_2s;

end
```

```
%% fCalc_geofprops1

% Calculation of geothermal fluid properties
% Frank Niewold
% Released version 1.0, February 2017

function [geofprops] = fCalc_geofprops1(P, T, w_NaCl, w_CO2, output)

    Excel  = actxGetRunningServer('Excel.Application');
    Sheets = Excel.ActiveWorkBook.Sheets;
    sheet2 = get(Sheets, 'Item', 1);
    invoke(sheet2, 'Activate');
    sheet  = Excel.Activesheet;

    % Calculate saturated vapor temperature T_sat_v at particular pressure
    if P < 40
        T_sat_v = interp1(output.VLE.P_sat_v, output.VLE.T_sat_v, P, 'spline');
    else
        T_sat_v = 0;
    end

    if P > 40
        sheet.set('Range', 'C3', P);
```

```
function [geofprops, T_new, w_table, chi_transition] = fCalc_geofprops2(P, T, w_NaCl, w_CO2, ...
                                                        H2O_sat, h, output, h_old, input, data)

    load T_SC; load m_SC; load SC;
    load chi_duan; load m_CO2_duan; load m_NaCl_duan; load P_sat_duan; load TP_duan
    data.T_SC = T_SC;
    data.m_SC = m_SC;
    data.SC = SC;

    % Declaration of initial variables, errors and number of iterations
    chi_transition = 1;
    dh_new         = h_old - h;
    T_new          = T - 0.1;
    dT_new         = T - T_new;
    h_check        = h + 11000;
    n_it           = 1;
    n_dT           = 0;

    % user-defined settings
    error_h_gp2     = input.settings.error_h_gp2;
    n_it_gp2        = input.settings.n_it_gp2;
    n_dT_gp2        = input.settings.n_dT_gp2;
    dT_gp2          = input.settings.dT_gp2;
    dT_VLE_sat_v    = input.settings.dT_VLE_sat_v;

    % Set right Excel sheet
    Excel  = actxGetRunningServer('Excel.Application');
    Sheets = Excel.ActiveWorkBook.Sheets;
    sheet2 = get(Sheets, 'Item', 1);
    invoke(sheet2, 'Activate');
    sheet  = Excel.Activesheet;

    % Calculate saturated vapor temperature T_sat_v at particular pressure
    if P < 40
        T_sat_v = interp1(output.VLE.P_sat_v, output.VLE.T_sat_v, P, 'spline');
    else
        T_sat_v = 0;
    end

    % Find properties P > 40 bar (Region 1, see report documentation)
    if P > 40
        sheet.set('Range', 'C3', P);
        sheet.set('Range', 'C4', T_new);
        sheet.set('Range', 'C8', w_NaCl);
        sheet.set('Range', 'C9', 0);
        sheet.set('Range', 'C10', 0);
        sheet.set('Range', 'C11', w_CO2);
        sheet.set('Range', 'C12', 0);
        sheet.set('Range', 'C13', 0);
        range = sheet.get('Range', 'I4:I19');
        range.Value;
        data_FM = range.Value;

        while (abs(h_check - h) > error_h_gp2) && (n_it < n_it_gp2) && (n_dT < n_dT_gp2)
                                                        % manual programmed iterative procedure
            T = T_new;
            dh_old = dh_new;
            dT_old = dT_new;

            sheet.set('Range', 'C3', P);
            sheet.set('Range', 'C4', T);
            sheet.set('Range', 'C8', w_NaCl);
            sheet.set('Range', 'C9', 0);
            sheet.set('Range', 'C10', 0);
            sheet.set('Range', 'C11', w_CO2);
            sheet.set('Range', 'C12', 0);
            sheet.set('Range', 'C13', 0);
            range = sheet.get('Range', 'I4:I19');
            range.Value;
            data_FM = range.Value;

            h_check          = cell2mat(data_FM(6,1));
            dh_new           = h_check - h;
            dh_step          = dh_old - dh_new;
            dT_new           = (dh_new/dh_step)*dT_old;
            T_new            = T - dT_new;
            chi_data(n_it,1) = cell2mat(data_FM(2,1));
            n_it             = n_it + 1;
```



```
        sheet.set('Range', 'C4', T);
        sheet.set('Range', 'C8', w_NaCl);
        sheet.set('Range', 'C11', w_CO2);
        range = sheet.get('Range', 'I4:I19');
        range.Value;
        data_FM = range.Value;
    elseif  (P < 40) && (T < T_sat_v)
        sheet.set('Range', 'C3', P);
        sheet.set('Range', 'C4', T);
        sheet.set('Range', 'C8', w_NaCl);
        sheet.set('Range', 'C11', w_CO2);
        range = sheet.get('Range', 'I4:I19');
        range.Value;
        data_FM = range.Value;
    else
        T = T_sat_v;
        sheet.set('Range', 'C3', P);
        sheet.set('Range', 'C4', T);
        sheet.set('Range', 'C8', w_NaCl);
        sheet.set('Range', 'C11', w_CO2);
        range = sheet.get('Range', 'I4:I19');
        range.Value;
        data_FM = range.Value;

        sheet2 = get(Sheets, 'Item', 2);
        invoke(sheet2, 'Activate');
        sheet = Excel.Activesheet;
        sheet.set('Range', 'C2', P);
        sheet.set('Range', 'C3', T);
        sheet.set('Range', 'C5', w_CO2);
        range = sheet.get('Range', 'G3:G5');
        range.Value;
        datagas_FM = range.Value;

    end

    Conversion.comma_to_dot (1,1) = strrep(data_FM(7,1), ',', '.');
    Conversion.to_character_liq   = char(Conversion.comma_to_dot(1,1));
    pattern = '\{|}';
    Conversion.to_character_liq   = regexprep(Conversion.to_character_liq, pattern, '');
    Conversion.char_to_value_liq  = str2num(Conversion.to_character_liq);
    Conversion.comma_to_dot (1,2) = strrep(data_FM(12,1), ',', '.');
    Conversion.to_character_gas   = char(Conversion.comma_to_dot(1,2));
    pattern = '\{|}';
    Conversion.to_character_gas   = regexprep(Conversion.to_character_gas, pattern, '');
    Conversion.char_to_value_gas  = str2num(Conversion.to_character_gas);

    geofprops              = zeros(26,1);
    geofprops(1:6,1)       = cell2mat(data_FM(1:6,1));
    if  isempty(Conversion.char_to_value_liq) == 1;
        geofprops(8:18,1)  = zeros;
    else geofprops(8:14,1) = Conversion.char_to_value_liq;
        geofprops(15:18,1) = cell2mat(data_FM(8:11,1));
    end

    if isempty(Conversion.char_to_value_gas) == 1;
        geofprops(19:26,1)  = zeros;
    else geofprops(19:22,1) = Conversion.char_to_value_gas;
        geofprops(23:25,1)  = cell2mat(data_FM(13:15,1));
        geofprops(26,1)     = (0.0042 * T + 1.7621)*10^-5;
    end
    % Calculate and store effective viscosity
    geofprops(7,1) = (geofprops(2,1) * geofprops(26,1)) + ((1 - geofprops(2,1)) * geofprops(18,1));
    geofprops      = geofprops.';
    geofprops(1,1) = geofprops(1,1)/10^5;

    if T == T_sat_v
        geofprops_gas    = cell2mat(datagas_FM(1:3,1));
        geofprops(2,1:3) = geofprops_gas.';
    end

end
```

```
%% fCalc_geofprops2

% Calculation of geothermal fluid properties
% Frank Niewold
% Released version 1.0, February 2017
```

```
if (abs(dT_new) > abs(dT_old)) % Check if calculation converges to a solution.
    n_dT = n_dT + 1;
end

if n_dT == n_dT_gp2 % If calculation did not converge to a solution.
    if h_check > h  % If iterated h > energy balance h, temperature is decreased.
        while h_check > h
            T = T - dT_gp2;
            sheet.set('Range', 'C3', P);
            sheet.set('Range', 'C4', T);
            sheet.set('Range', 'C8', w_NaCl);
            sheet.set('Range', 'C9', 0);
            sheet.set('Range', 'C10', 0);
            sheet.set('Range', 'C11', w_CO2);
            sheet.set('Range', 'C12', 0);
            sheet.set('Range', 'C13', 0);
            range = sheet.get('Range', 'I4:I19');
            range.Value;
            data_FM = range.Value;
            h_check = cell2mat(data_FM(6,1));
        end
    else
        while h_check < h % If iterated h < energy balance h, temperature is increased.
            T = T + dT_gp2;
            sheet.set('Range', 'C3', P);
            sheet.set('Range', 'C4', T);
            sheet.set('Range', 'C8', w_NaCl);
            sheet.set('Range', 'C9', 0);
            sheet.set('Range', 'C10', 0);
            sheet.set('Range', 'C11', w_CO2);
            sheet.set('Range', 'C12', 0);
            sheet.set('Range', 'C13', 0);
            range = sheet.get('Range', 'I4:I19');
            range.Value;
            data_FM = range.Value;
            h_check = cell2mat(data_FM(6,1));
        end
    end
end

if n_it == n_it_gp2 % If no convergence to a solution after n_it_gp2 iterations
    chi_data_min = min(chi_data(:,1));
    chi_data_max = max(chi_data(:,1));
    chi_check    = cell2mat(data_FM(2,1));
    % Check if T/P iterates between P_degas discontinuity of Francke Model
    if chi_data_min < 0.0001 && chi_data_max > 0.001
        while chi_check > 0.001
            T = T + dT_gp2;

            sheet.set('Range', 'C3', P);
            sheet.set('Range', 'C4', T);
            sheet.set('Range', 'C8', w_NaCl);
            sheet.set('Range', 'C9', 0);
            sheet.set('Range', 'C10', 0);
            sheet.set('Range', 'C11', w_CO2);
            sheet.set('Range', 'C12', 0);
            sheet.set('Range', 'C13', 0);
            range = sheet.get('Range', 'I4:I19');
            range.Value;
            data_FM = range.Value;

            chi_check = cell2mat(data_FM(2,1));
        end
    elseif chi_data_min > 0.0001 && chi_data_max > 0.001
        if h_check > h
            while h_check > h
                T = T - dT_gp2;

                sheet.set('Range', 'C3', P);
                sheet.set('Range', 'C4', T);
                sheet.set('Range', 'C8', w_NaCl);
                sheet.set('Range', 'C9', 0);
                sheet.set('Range', 'C10', 0);
                sheet.set('Range', 'C11', w_CO2);
                sheet.set('Range', 'C12', 0);
                sheet.set('Range', 'C13', 0);
                range = sheet.get('Range', 'I4:I19');
                range.Value;
                data_FM = range.Value;

                h_check = cell2mat(data_FM(6,1));
            end
        else
            while h_check < h
                T = T + dT_gp2;

                sheet.set('Range', 'C3', P);
                sheet.set('Range', 'C4', T);
                sheet.set('Range', 'C8', w_NaCl);
                sheet.set('Range', 'C9', 0);
                sheet.set('Range', 'C10', 0);
                sheet.set('Range', 'C11', w_CO2);
                sheet.set('Range', 'C12', 0);
                sheet.set('Range', 'C13', 0);
                range = sheet.get('Range', 'I4:I19');
                range.Value;
                data_FM = range.Value;

                h_check = cell2mat(data_FM(6,1));
            end
        end
    else
        T = T; % If none of the above iterations procedures worked.

        sheet.set('Range', 'C3', P);
        sheet.set('Range', 'C4', T);
        sheet.set('Range', 'C8', w_NaCl);
        sheet.set('Range', 'C9', 0);
        sheet.set('Range', 'C10', 0);
        sheet.set('Range', 'C11', w_CO2);
        sheet.set('Range', 'C12', 0);
        sheet.set('Range', 'C13', 0);
        range = sheet.get('Range', 'I4:I19');
        range.Value;
        data_FM = range.Value;
    end
end

% Find properties (P < 40) && (T < T_sat_v - dT_VLE_sat_v) (Region 2, see report documentation)
elseif  (P < 40) && (T < T_sat_v - dT_VLE_sat_v)

    sheet.set('Range', 'C3', P);
    sheet.set('Range', 'C4', T);
    sheet.set('Range', 'C8', w_NaCl);
    sheet.set('Range', 'C9', 0);
    sheet.set('Range', 'C10', 0);
    sheet.set('Range', 'C11', w_CO2);
    sheet.set('Range', 'C12', 0);
    sheet.set('Range', 'C13', 0);
    range = sheet.get('Range', 'I4:I19');
    range.Value;
    data_FM = range.Value;

    h_check = cell2mat(data_FM(6,1));

    while (h < h_check) && (T < T_sat_v - dT_VLE_sat_v)
        T = T - dT_gp2;
        sheet.set('Range', 'C3', P);
        sheet.set('Range', 'C4', T);
        sheet.set('Range', 'C8', w_NaCl);
        sheet.set('Range', 'C9', 0);
        sheet.set('Range', 'C10', 0);
        sheet.set('Range', 'C11', w_CO2);
        sheet.set('Range', 'C12', 0);
        sheet.set('Range', 'C13', 0);
        range = sheet.get('Range', 'I4:I19');
        range.Value;
        data_FM = range.Value;

        h_check = cell2mat(data_FM(6,1));
        chi_check = cell2mat(data_FM(2,1));

        if iscellstr(data_FM(1,1)) == 1
            h_check = h + 1;
        end
    end
```

```
% Find properties (P < 40) && (T > T_sat_v - dT_VLE_sat_v) (Region 3, see report documentation)
else
    T = T_sat_v - dT_VLE_sat_v; % T is decreased with dT to make sure Francke Model does not crash

    Sheets = Excel.ActiveWorkBook.Sheets;
    sheet2 = get(Sheets, 'Item', 1);
    invoke(sheet2, 'Activate');
    sheet = Excel.Activesheet;
    sheet.set('Range', 'C3', P);
    sheet.set('Range', 'C4', T);
    sheet.set('Range', 'C8', w_NaCl);
    sheet.set('Range', 'C9', 0);
    sheet.set('Range', 'C10', 0);
    sheet.set('Range', 'C11', w_CO2);
    sheet.set('Range', 'C12', 0);
    sheet.set('Range', 'C13', 0);
    range = sheet.get('Range', 'I4:I19');
    range.Value;
    data_FM = range.Value;

    chi_check = cell2mat(data_FM(2,1));
    h_check = cell2mat(data_FM(6,1));

    if iscellstr(data_FM(1,1)) == 1 % Francke Model is below saturated vapor pressure
        T = T_sat_v - dT_VLE_sat_v; % T is decreased with dT, that Francke Model does not crash
    end

     if chi_check == 1 % Francke Model shows discontinuity close to saturated vapor properties.
        dT_old = dT_VLE_sat_v;
        while chi_check == 1 % Change T, until quality chi is below 1 again.
            T_old = T;

            Sheets = Excel.ActiveWorkBook.Sheets;
            sheet2 = get(Sheets, 'Item', 1);
            invoke(sheet2, 'Activate');
            sheet = Excel.Activesheet;
            sheet.set('Range', 'C3', P);
            sheet.set('Range', 'C4', T_old);
            sheet.set('Range', 'C8', w_NaCl);
            sheet.set('Range', 'C9', 0);
            sheet.set('Range', 'C10', 0);
            sheet.set('Range', 'C11', w_CO2);
            sheet.set('Range', 'C12', 0);
            sheet.set('Range', 'C13', 0);
            range = sheet.get('Range', 'I4:I19');
            range.Value;
            data_FM = range.Value;

            chi_check = cell2mat(data_FM(2,1));

            if chi_check < 1
                break
            end

            dT_VLE_sat_v = dT_old + dT_VLE_sat_v;
            T = T_sat_v - dT_VLE_sat_v;
        end
     end

    % interpolate geothermal fluid properties as saturated vapor curve
    geofprops_gas(1,1) = interp1(output.VLE.P_sat_v, output.VLE.rho_sat_v, P, 'spline');
    geofprops_gas(2,1) = interp1(output.VLE.P_sat_v, output.VLE.cp_sat_v, P, 'spline');
    geofprops_gas(3,1) = interp1(output.VLE.P_sat_v, output.VLE.h_sat_v, P, 'spline');
    geofprops_gas(4,1) = interp1(output.VLE.P_sat_v, output.VLE.T_sat_v, P, 'spline');

    geofprops_liq(1,1) = interp1(output.VLE.P_sat_v, output.VLE.rho_sat_l, P, 'spline');
    geofprops_liq(2,1) = interp1(output.VLE.P_sat_v, output.VLE.cp_sat_l, P, 'spline');
    geofprops_liq(3,1) = interp1(output.VLE.P_sat_v, output.VLE.h_sat_l, P, 'spline');
    geofprops_liq(4,1) = interp1(output.VLE.P_sat_v, output.VLE.T_sat_v, P, 'spline');
end

% Make input from Francke Model interpretable for MATLAB
T_new = T;
Conversion.comma_to_dot (1,1) = strrep(data_FM(7,1), ',', '.');
Conversion.to_character_liq   = char(Conversion.comma_to_dot(1,1));
pattern = '\{|}';
Conversion.to_character_liq   = regexprep(Conversion.to_character_liq, pattern, '');
Conversion.char_to_value_liq  = str2num(Conversion.to_character_liq);
Conversion.comma_to_dot (1,2) = strrep(data_FM(12,1), ',', '.');
Conversion.to_character_gas   = char(Conversion.comma_to_dot(1,2));
pattern = '\{|}';
Conversion.to_character_gas   = regexprep(Conversion.to_character_gas, pattern, '');
Conversion.char_to_value_gas  = str2num(Conversion.to_character_gas);

% Obtain properties from Francke Model
geofprops         = zeros(26,1);
geofprops(1:6,1) = cell2mat(data_FM(1:6,1));
if  isempty(Conversion.char_to_value_liq) == 1;
    geofprops(8:18,1)  = zeros;
else geofprops(8:14,1) = Conversion.char_to_value_liq;
    geofprops(15:18,1) = cell2mat(data_FM(8:11,1));
end

if isempty(Conversion.char_to_value_gas) == 1;
    geofprops(19:26,1)  = zeros;
else geofprops(19:22,1) = Conversion.char_to_value_gas;
    geofprops(23:25,1)  = cell2mat(data_FM(13:15,1));
    geofprops(26,1)     = (0.0042 * T + 1.7621)*10^-5;
end

% Calculate and store effective viscosity
geofprops(7,1) = (geofprops(2,1) * geofprops(26,1)) + ((1 - geofprops(2,1)) * geofprops(18,1));
geofprops = geofprops.'; % Transpose matrix
geofprops(1,1) = geofprops(1,1)/10^5; % Pascal to bar

% Calculation of composition
% x_NaCl_liq
w_table(3,2) = geofprops(1,8);
% x_CO2_liq
w_table(3,3) = geofprops(1,11);
% x_CO2_vap
w_table(3,4) = geofprops(1,19);
% x_H2O_liq
w_table(3,5) = geofprops(1,14);
% x_H2O_vap
w_table(3,6) = geofprops(1,22);

% geofprops
% (1,1)     degassing pressure mixture            - P_degas
% (1,2)     quality                               - chi
% (1,3)     void fraction mixture                 - eps_G
% (1,4)     density mixture                       - rho
% (1,5)     specific heat capacity mixture        - c_p
% (1,6)     specific enthalpy mixture             - h
% (1,7)     viscosity mixture                     - mu
% (1,8)     mass fraction NaCl liquid phase       - w_NaCl_l
% (1,9)     mass fraction KCl liquid phase        - N.A.
% (1,10)    mass fraction CaCl2 liquid phase      - N.A.
% (1,11)    mass fraction CO2 liquid phase        - w_CO2_l
% (1,12)    mass fraction N2 liquid phase         - N.A.
% (1,13)    mass fraction CH4 liquid phase        - N.A.
% (1,14)    mass fraction H2O liquid phase        - w_H2O_l
% (1,15)    density liquid phase                  - rho_l
% (1,16)    specific heat capacity liquid phase   - c_p_l
% (1,17)    specific enthalpy liquid phase        - h_l
% (1,18)    viscosity liquid phase                - mu_l
% (1,19)    mass fraction CO2 gas phase           - w_CO2_v
% (1,20)    mass fraction N2 gas phase            - N.A.
% (1,21)    mass fraction CH4 gas phase           - N.A.
% (1,22)    mass fraction H2O gas phase           - w_CO2_g
% (1,23)    density gas phase                     - rho_v
% (1,24)    specific heat capacity gas phase      - c_p_v
% (1,25)    specific enthalpy gas phase           - h_v
% (1,26)    viscosity gas phase                   - mu_v


 if T == T_sat_v - dT_VLE_sat_v
%           if geofprops(1,2) == 0;
    geofprops(1,17) = geofprops_liq(3,1);
    geofprops(1,15) = geofprops_liq(1,1);
    geofprops(1,16) = geofprops_liq(2,1);
%   end
    chi_relative     = (h - geofprops(1,6))/(geofprops_gas(3,1)- geofprops(1,6));
    T_sat_v          = interp1(output.VLE.P_sat_v, output.VLE.T_sat_v, P, 'spline');
    T_new            = T * (1-chi_relative) + (geofprops_gas(4,1) * chi_relative);
    rho_sat_l        = geofprops(1,4);
    chi_transition   = geofprops(1,2);
```

```
geofprops(1,2)   = ((h - geofprops(1,6))/(geofprops_gas(3,1) - geofprops(1,6)))*(1 - ...
                   geofprops(1,2)) + geofprops(1,2);
v_spec           = (1/geofprops(1,4)) * (1-chi_relative) + ((1/geofprops_gas(1,1)) * ...
                    chi_relative);
geofprops(1,4)   = 1/v_spec;
geofprops(1,5)   = (geofprops(1,5)) * (1 - chi_relative) + ((geofprops_gas(2,1)) * ...
                   chi_relative);
geofprops(1,7)   = (geofprops(1,7)) * (1 - chi_relative) + ((geofprops(1,26)) * chi_relative);
geofprops(1,3)   = (rho_sat_l - geofprops(1,4))/(rho_sat_l - geofprops_gas(1,1))*...
                   (1-geofprops(1,3)) + geofprops(1,3);
geofprops(2,1:4) = geofprops_gas.';
geofprops(1,23)  = geofprops(2,1); %rho_vap
geofprops(1,26)  = (0.0042 * T + 1.7621)*10^-5; %mu_vap

% load m_SC.mat; load T_SC.mat; load SC.mat;
% Calculate mass fraction in liquid and vapor phase with separation coefficient
w_table = zeros(1);
% w_CO2_v
SC           = interp2(data.T_SC, data.m_SC, data.SC, T_new, input.general.m_NaCl);
SC           = 10^SC;
w_CO2_rel    = SC / (((1/geofprops(1,2)) - 1) + SC);
w_CO2_v      = (w_CO2_rel * w_CO2)/geofprops(1,2);
w_table(3,4) = w_CO2_v;
% w_H2O_v
w_H2O_v      = 1 - w_CO2_v;
w_table(3,6) = w_H2O_v;
% w_NaCl_l
w_NaCl_l     = w_NaCl / (1 - (geofprops(1,2) * (1 - w_NaCl)));
w_table(3,2) = w_NaCl_l;
% w_CO2_l
w_CO2_l      = (w_CO2 - (w_CO2_v * geofprops(1,2)))/(1-geofprops(1,2));
w_table(3,3) = w_CO2_l;
% w_H2O_l
w_H2O_l      = 1 - w_CO2_l - w_NaCl_l;
w_table(3,5) = w_H2O_l;

    end

end
```

```
%% fCalc_geofprops3

% Calculation of geothermal fluid properties between P_degas of Duan and
% Sun (2003) and Francke (2014)
% Frank Niewold
% Released version 1.0, February 2017

function [T_new, rho_m, c_p_m, mu_m, eps_G, rho_g, rho_l, mu_g, mu_l] = fCalc_geofprops3(P, T, ~, ...
                                                    w_NaCl_l, w_CO2_g, h, chi, h_old, input, T_old,l,T_int)

    Excel = actxGetRunningServer('Excel.Application');
    Sheets = Excel.ActiveWorkBook.Sheets;

    % Declaration of initial variables, errors and number of iterations
    T_initial = T;
    dh_new  =   h_old - h;
    T_new   =   T - 0.2;
    dT_new  =   T - T_new;
    h_m     =   h + 2;
    n_it    =   1;
    n_dT    =   0;

    error_h_gp3 = input.settings.error_h_gp3;
    n_it_gp3    = input.settings.n_it_gp3;
    n_dT_gp3    = input.settings.n_dT_gp3;
    dT_gp3      = input.settings.dT_gp3;

    % Start iterative procedure
    while (abs(h_m - h) > error_h_gp3) && (n_it < n_it_gp3) && (n_dT < n_dT_gp3)
        T      = T_new;
        dh_old = dh_new;
        dT_old = dT_new;

        % gas phase properties
        sheet2 = get(Sheets, 'Item', 2);
        invoke(sheet2, 'Activate');
        sheet  = Excel.Activesheet;
        sheet.set('Range', 'C2', P);
        sheet.set('Range', 'C3', T);
        sheet.set('Range', 'C5', w_CO2_g);
        sheet.set('Range', 'C6', 0);
        sheet.set('Range', 'C7', 0);
        range = sheet.get('Range', 'G3:G5');
        range.Value;
        datagas_FM = range.Value;
        h_g     = cell2mat(datagas_FM(3,1));

        % liquid phase properties
        sheet2 = get(Sheets, 'Item', 3);
        invoke(sheet2, 'Activate');
        sheet  = Excel.Activesheet;
        sheet.set('Range', 'C3', P);
        sheet.set('Range', 'C4', T);
        sheet.set('Range', 'C6', w_NaCl_l);
        sheet.set('Range', 'C7', 0);
        sheet.set('Range', 'C8', 0);
        range = sheet.get('Range', 'G4:G7');
        range.Value;
        data_FM  = range.Value;
        h_l      = cell2mat(data_FM(3,1));

        % mixture properties
        h_m              = h_g * chi + h_l * (1 - chi);
        dh_new           = h_m - h;
        dh_step          = dh_old - dh_new;
        dT_new           = (dh_new/dh_step)*dT_old;
        T_new            = T - dT_new;
        chi_data(n_it,1) = cell2mat(data_FM(2,1));
        n_it             = n_it + 1;

        if (abs(dT_new) > abs(dT_old)) % Check if calculation converges to a solution
            n_dT = n_dT + 1;
        end

        if n_dT == 2 % If calculation did not converge to a solution
            if h_m > h % If iterated mixture h > h from energy balance, temperature is decreased.
                while h_m > h
                    T = T - dT_gp3;

                    % gas phase properties
                    sheet2 = get(Sheets, 'Item', 2);
                    invoke(sheet2, 'Activate');
                    sheet  = Excel.Activesheet;
                    sheet.set('Range', 'C2', P);
                    sheet.set('Range', 'C3', T);
                    sheet.set('Range', 'C5', w_CO2_g);
                    sheet.set('Range', 'C6', 0);
                    sheet.set('Range', 'C7', 0);
                    range = sheet.get('Range', 'G3:G5');
                    range.Value;
                    datagas_FM = range.Value;
                    h_g        = cell2mat(datagas_FM(3,1));

                    % liquid phase properties
                    sheet2 = get(Sheets, 'Item', 3);
                    invoke(sheet2, 'Activate');
                    sheet  = Excel.Activesheet;
                    sheet.set('Range', 'C3', P);
                    sheet.set('Range', 'C4', T);
                    sheet.set('Range', 'C6', w_NaCl_l);
                    sheet.set('Range', 'C7', 0);
                    sheet.set('Range', 'C8', 0);
                    range  = sheet.get('Range', 'G4:G7');
                    range.Value;
                    data_FM = range.Value;
                    h_l     = cell2mat(data_FM(3,1));

                    % mixture properties
                    h_m = h_g * chi + h_l * (1-chi);
                end
            else
                while h_m < h % If iterated mixture h < energy balance h, temperature is increased.
                    T = T + dT_gp3;

                    % gas phase properties
                    sheet2 = get(Sheets, 'Item', 2);
                    invoke(sheet2, 'Activate');
```

```
            sheet  = Excel.Activesheet;
            sheet.set('Range', 'C2', P);
            sheet.set('Range', 'C3', T);
            sheet.set('Range', 'C5', w_CO2_g);
            sheet.set('Range', 'C6', 0);
            sheet.set('Range', 'C7', 0);
            range  = sheet.get('Range', 'G3:G5');
            range.Value;
            datagas_FM = range.Value;
            h_g        = cell2mat(datagas_FM(3,1));

            % liquid phase properties
            sheet2 = get(Sheets, 'Item', 3);
            invoke(sheet2, 'Activate');
            sheet  = Excel.Activesheet;
            sheet.set('Range', 'C3', P);
            sheet.set('Range', 'C4', T);
            sheet.set('Range', 'C6', w_NaCl_l);
            sheet.set('Range', 'C7', 0);
            sheet.set('Range', 'C8', 0);
            range  = sheet.get('Range', 'G4:G7');
            range.Value;
            data_FM = range.Value;
            h_l     = cell2mat(data_FM(3,1));

            % mixture properties
            h_m = h_g * chi + h_l * (1-chi);
        end
    end
end
if n_it == n_it_gp3 % If calculation did not converge to a solution after n_it_gp3 iterations
    if h_m > h % If iterated mixture h > h from energy balance, temperature is decreased.
        while h_m > h
            T = T - dT_gp3;
            sheet2 = get(Sheets, 'Item', 2);
            invoke(sheet2, 'Activate');
            sheet = Excel.Activesheet;
            sheet.set('Range', 'C2', P);
            sheet.set('Range', 'C3', T);
            sheet.set('Range', 'C5', w_CO2_g);
            sheet.set('Range', 'C6', 0);
            sheet.set('Range', 'C7', 0);
            range = sheet.get('Range', 'G3:G5');
            range.Value;
            datagas_FM = range.Value;
            h_g = cell2mat(datagas_FM(3,1));

            sheet2 = get(Sheets, 'Item', 3);
            invoke(sheet2, 'Activate');
            sheet = Excel.Activesheet;
            sheet.set('Range', 'C3', P);
            sheet.set('Range', 'C4', T);
            sheet.set('Range', 'C6', w_NaCl_l);
            sheet.set('Range', 'C7', 0);
            sheet.set('Range', 'C8', 0);
            range = sheet.get('Range', 'G4:G7');
            range.Value;
            data_FM = range.Value;
            h_l = cell2mat(data_FM(3,1));

            h_m = h_g * chi + h_l * (1-chi);
        end
    else
        while h_m < h % If iterated mixture h < energy balance h, temperature is increased.
            T      = T + dT_gp3;
            sheet2 = get(Sheets, 'Item', 2);
            invoke(sheet2, 'Activate');
            sheet  = Excel.Activesheet;
            sheet.set('Range', 'C2', P);
            sheet.set('Range', 'C3', T);
            sheet.set('Range', 'C5', w_CO2_g);
            sheet.set('Range', 'C6', 0);
            sheet.set('Range', 'C7', 0);
            range  = sheet.get('Range', 'G3:G5');
            range.Value;
            datagas_FM = range.Value;
            h_g        = cell2mat(datagas_FM(3,1));

            sheet2 = get(Sheets, 'Item', 3);
            invoke(sheet2, 'Activate');
            sheet  = Excel.Activesheet;
            sheet.set('Range', 'C3', P);
            sheet.set('Range', 'C4', T);
            sheet.set('Range', 'C6', w_NaCl_l);
            sheet.set('Range', 'C7', 0);
            sheet.set('Range', 'C8', 0);
            range  = sheet.get('Range', 'G4:G7');
            range.Value;
            data_FM = range.Value;
            h_l     = cell2mat(data_FM(3,1));

            h_m = h_g * chi + h_l * (1-chi);
        end
    end
else
    T = T; % If none of the above iterations procedures worked.

    % gas phase properties
    sheet2 = get(Sheets, 'Item', 2);
    invoke(sheet2, 'Activate');
    sheet  = Excel.Activesheet;
    sheet.set('Range', 'C2', P);
    sheet.set('Range', 'C3', T);
    sheet.set('Range', 'C5', w_CO2_g);
    sheet.set('Range', 'C6', 0);
    sheet.set('Range', 'C7', 0);
    range  = sheet.get('Range', 'G3:G5');
    range.Value;
    datagas_FM = range.Value;
    h_g        = cell2mat(datagas_FM(3,1));

    % liquid phase properties
    sheet2 = get(Sheets, 'Item', 3);
    invoke(sheet2, 'Activate');
    sheet  = Excel.Activesheet;
    sheet.set('Range', 'C3', P);
    sheet.set('Range', 'C4', T);
    sheet.set('Range', 'C6', w_NaCl_l);
    sheet.set('Range', 'C7', 0);
    sheet.set('Range', 'C8', 0);
    range  = sheet.get('Range', 'G4:G7');
    range.Value;
    data_FM = range.Value;
    h_l     = cell2mat(data_FM(3,1));

    % mixture properties
    h_m = h_g * chi + h_l * (1-chi);
end

rho_g = cell2mat(datagas_FM(1,1));   % density gas phase [kg/m3]
c_p_g = cell2mat(datagas_FM(2,1));   % heat capacity gas phase [J/kg/K]
mu_g  = (0.0042 * T + 1.7621)*10^-5; % viscosity gas phase [Pa*s]

rho_l = cell2mat(data_FM(1,1)); % density liquid phase [kg/m3]
c_p_l = cell2mat(data_FM(2,1)); % heat capacity liquid phase [J/kg/K]
mu_l  = cell2mat(data_FM(4,1)); % viscosity liquid phase [Pa*s]

T_new = T;                                          % Calculated temperature [C]
rho_m = 1/((1/rho_g) * chi + (1/rho_l) * (1-chi)); % density mixture [kg/m3]
c_p_m = c_p_g * chi + c_p_l * (1-chi);              % heat capacity mixture [J/kg/K]
mu_m  = mu_g * chi + mu_l * (1-chi);                % viscosity mixture [Pa*s]
eps_G = (chi/rho_g)/((chi/rho_g)+((1-chi)/rho_l)); % void fraction homogeneous flow [-]

if T_initial - T_new < T_old - T_initial
    if l < 10
        T_new = T_initial - (T_old - T_initial);
    else
        T_new = T_int;
    end
    % gas phase properties
    sheet2 = get(Sheets, 'Item', 2);
    invoke(sheet2, 'Activate');
    sheet  = Excel.Activesheet;
    sheet.set('Range', 'C2', P);
    sheet.set('Range', 'C3', T_new);
    sheet.set('Range', 'C5', w_CO2_g);
    sheet.set('Range', 'C6', 0);
```

```
        sheet.set('Range', 'C7', 0);
        range  = sheet.get('Range', 'G3:G5');
        range.Value;
        datagas_FM = range.Value;
        h_g        = cell2mat(datagas_FM(3,1));

        % liquid phase properties
        sheet2 = get(Sheets, 'Item', 3);
        invoke(sheet2, 'Activate');
        sheet  = Excel.Activesheet;
        sheet.set('Range', 'C3', P);
        sheet.set('Range', 'C4', T_new);
        sheet.set('Range', 'C6', w_NaCl_l);
        sheet.set('Range', 'C7', 0);
        sheet.set('Range', 'C8', 0);
        range  = sheet.get('Range', 'G4:G7');
        range.Value;
        data_FM = range.Value;
        h_l     = cell2mat(data_FM(3,1));

        % mixture properties
        h_m = h_g * chi + h_l * (1-chi);

        rho_g = cell2mat(datagas_FM(1,1));   % density gas phase [kg/m3]
        c_p_g = cell2mat(datagas_FM(2,1));   % heat capacity gas phase [J/kg/K]
        mu_g  = (0.0042 * T + 1.7621)*10^-5; % viscosity gas phase [Pa*s]

        rho_l = cell2mat(data_FM(1,1)); % density liquid phase [kg/m3]
        c_p_l = cell2mat(data_FM(2,1)); % heat capacity liquid phase [J/kg/K]
        mu_l  = cell2mat(data_FM(4,1)); % viscosity liquid phase [Pa*s]

        rho_m = 1/((1/rho_g) * chi + (1/rho_l) * (1-chi)); % density mixture [kg/m3]
        c_p_m = c_p_g * chi + c_p_l * (1-chi);             % heat capacity mixture [J/kg/K]
        mu_m  = mu_g * chi + mu_l * (1-chi);               % viscosity mixture [Pa*s]
        eps_G = (chi/rho_g)/((chi/rho_g)+((1-chi)/rho_l)); % void fraction homogeneous flow [-]
    end
end
```

```
%% fCalc_geofprops4

% Calculation of geothermal fluid properties
% Frank Niewold
% Released version 1.0, February 2017

function [geofprops] = fCalc_geofprops4(P, T, w_NaCl, w_CO2)

    Excel  = actxGetRunningServer('Excel.Application');
    Sheets = Excel.ActiveWorkBook.Sheets;
    sheet2 = get(Sheets, 'Item', 1);
    invoke(sheet2, 'Activate');
    sheet  = Excel.Activesheet;
    sheet.set('Range', 'C3', P);
    sheet.set('Range', 'C4', T);
    sheet.set('Range', 'C8', w_NaCl);
    sheet.set('Range', 'C11', w_CO2);
    range  = sheet.get('Range', 'D8:D13');
    range2 = sheet.get('Range', 'I4:I18');
    range.Value;
    range2.Value;
    data_FM = range.Value;
    data2_FM = range2.Value;

    geofprops(1,1) = cell2mat(data_FM(1,1));
    geofprops(2,1) = cell2mat(data_FM(4,1));
    geofprops(1:6,2) = cell2mat(data2_FM(1:6,1));
    geofprops(8:11,2) = cell2mat(data2_FM(8:11,1));

end
```

```
%% fCalc_eps_G

% Calculation of the void fraction in the wellbore
% Frank Niewold
% Released version 1.0, February 2017

function [eps_G, FP, u_gu, C_0] = fCalc_eps_G(T, rho_l, rho_g, mu_l, mu_g, l_E, D_i, eps_pipe, ...
                                  u_sg, u_sl, g, chi, DF_model)

options = optimset('Display','off');

% General variables
T      = T + 273.15;                                                % temperature [K]
sigma  = 0.2358 * ((1 - (T/647.096))^1.256) * (1 - 0.625 * (1 - (T/647.096)));
                                                                    % surface tension water [kg/m2]
mu_kin = mu_l/rho_l;                                                % kinematic viscosity [m2/s]
mu_m   = chi * mu_g + (1-chi) * mu_l;                               % viscosity of mixture [Pa*s]
u_m    = u_sl + u_sg;                                               % mixture velocity [m/s]

if DF_model == 1; % Homogeneous model
    % None of the drift-flux models are applied

%% Rouhani & Axelsson (1970)
elseif DF_model == 2;

    C_0    = 1.1;                                                   % Distribution parameter
    %C_0    = 1.0 + 0.2*(1-chi);                                     % Distribution parameter
    u_gu   = 1.18 *(g *sigma*(rho_l-rho_g)/(rho_l^2))^0.25;         % drift-flux velocity [m/s]
    eps_G  = u_sg / (C_0 * u_m + u_gu);                             % void fraction [-]
%% Hasan & Kabir (2010)
elseif DF_model == 3;

    % Distribution parameters
    C_0b   = 1.2;
    C_0s   = 1.2;
    C_0c   = 1.15;
    C_0cdb = 1.15;
    C_0a   = 1.0;

    u_b    = 1.53 * (g*(rho_l-rho_g)*sigma/(rho_l^2))^0.25; % small bubble rise velocity [m/s]
    u_T    = 0.35 * (g*D_i*(rho_l-rho_g)/rho_l)^0.5;        % Taylor bubble rise velocity [m/s]
    u_m    = u_sl + u_sg;                                   % mixture velocity [m/s]

    % Check bubble pattern
    C_0    = C_0b;                    % distribution parameter [-]
    u_gu   = u_b;                     % drift-flux velocity [m/s]
    eps_G = u_sg / (C_0*u_m + u_gu); % void fraction [-]

    rho_m = eps_G * rho_g + (1-eps_G)* rho_l;        % density mixture [kg/m3]
    Re_m  = rho_m * u_m * D_i / mu_m;                % Reynolds number mixture [-]
    f_DW  = 0.25/(log10((eps_pipe/D_i/3.7065)-((5.0452/Re_m)*log10(((1/2.8257)*...
            (eps_pipe/D_i)^1.1098)+(5.8506/Re_m^0.8981))))^2);
                                                     % friction factor [-] (Hasan and Kabir, 2010)
    %f_DW = 0.046*Re_m^-0.2;                         % friction factor (Taitel et al., 1980)
    u_ms = ((0.725 + 4.15*(u_sl/u_m)^0.5)/((((f_DW/2/D_i)^0.4)*((rho_l/sigma)^0.6)*...
           (((0.4*sigma)/(g*(rho_l-rho_g)))^0.5)*2)))^(1/1.2);
                    % minimum mixture velocity for dispersed bubble flow (Hasan and Kabir, 2010)
    u_ms_taitel = 4 *(D_i^0.429*(sigma/rho_l)^0.089/mu_kin^0.072)*((g*(rho_l-rho_g)/rho_l)^0.446);
                    % minimum mixture velocity for dispersed bubble flow (Taitel et al., 1980)
    u_gc = 3.1 * (g*sigma*(rho_l-rho_g)/(rho_g)^2)^0.25;
                                        % superficial gas velocity for transition to annular flow
    if eps_G < 0.25 && (u_sg < u_gc)             % bubble or dispersed bubble flow
        if (u_m > u_ms) && (u_sg < u_gc)         % dispersed bubble flow
            C_0    = C_0cdb;                     % distribution parameter
            %C_0    =  1 + 0.2*(1-chi);           % Rouhani & Axelsson (1970)
            eps_G = u_sg / (C_0 * (u_m) + u_gu);  % void fraction [-]
            FP     = 'dispersed bubble';
        elseif (u_sg < u_gc)                     % bubble flow
            FP     = 'bubble';
        end
    else % dispersed bubble, slug, churn or annular flow
        % iterative procedure
        u_gb = u_sg - 0.1;                       % initial value
        x0   = u_gb;                             % iteration variable
        y0   = [u_b, u_sg, u_T, C_0b, C_0s, u_sl]; % iteration constants
        f    = @(x0)fCalc_u_gb(x0,y0);
        u_gb = fsolve(f,x0,options);             % transition from bubble to slug flow
        C_0  = C_0b*(1-exp(-0.1*u_gb/(u_sg-u_gb))) + C_0s*(exp(-0.1*u_gb/(u_sg-u_gb)));
                                        % distribution parameter transtion from bubble to slug flow
        u_gu = u_b * (1-exp(-0.1*u_gb/(u_sg-u_gb))) + u_T *(exp(-0.1*u_gb/(u_sg-u_gb)));
                                        % drift-flux velocity transition from bubble to slug flow
        %C_0_B =  1 + 0.2*(1-chi); % Rouhani & Axelsson (1970)
        %C_0 = min(C_0_A,C_0_B);
        %C_0 = 1.1;
        eps_G = u_sg / (C_0 * u_m + u_gu);       % void fraction slug flow [-]
        rho_m = eps_G * rho_g + (1-eps_G)* rho_l;  % density mixture [kg/m3]
        Re_m  = rho_m * u_m * D_i / mu_m;        % Reynolds number mixture [-]
```

```
        f_DW  = 0.25/(log10((eps_pipe/D_i/3.7065)-((5.0452/Re_m)*log10(((1/2.8257)*...
                (eps_pipe/D_i)^1.1098)+(5.8506/Re_m^0.8981))))^2);
                                                % friction factor [-] (Hasan and Kabir, 2010)
        % f_DW = 0.046*Re_m^-0.2;               % friction factor (Taitel et al., 1980)
        u_ms   = ((0.725 + 4.15*(u_sl/u_m)^0.5)/((((f_DW/2/D_i)^0.4)*((rho_l/sigma)^0.6)*...
                 (((0.4*sigma)/(g*(rho_l-rho_g)))^0.5)*2)))^(1/1.2);
                      % minimum mixture velocity for dispersed bubble flow (Hasan and Kabir, 2010)
        u_ms_taitel = 4 * (D_i^0.429*(sigma/rho_l)^0.089/mu_kin^0.072)*((g*...
                      (rho_l-rho_g)/rho_l)^0.446);
                        % minimum mixture velocity for dispersed bubble flow (Taitel et al., 1980)

        %u_gc = 3.1 * (g*sigma*(rho_l-rho_g)/(rho_g)^2)^0.25;
        u_gc = 2.0 * (g*sigma*(rho_l-rho_g)/(rho_g)^2)^0.25;
                                        % superficial gas velocity for transition to annular flow
        % u_mc = ((0.725 + 4.15*(u_sl/u_m)^0.5)/((((f_DW/2/D_i)^0.4)*...
        %        ((rho_l/sigma)^0.6)*(((0.4*sigma)/(g*(rho_l-rho_g)))^0.5)*2)))^(1/1.2);
                % mixture velocity for transition from slug to churn flow (Hasan and Kabir, 2010)
        u_mc = (l_E/D_i/40.6 - 0.22) * (g*D_i)^0.5;
                % mixture velocity for transition from slug to churn flow (Taitel et al, 1980)
        if (u_m > u_mc) && (u_sg > 1.08*u_sl) && (u_sg < u_gc)% churn flow
            C_0   = C_0s*(1-exp(-0.1*u_mc/(u_m-u_mc))) + C_0c*(exp(-0.1*u_mc/(u_m-u_mc)));
                                % distribution parameter for transition from slug to churn flow
            % iterative procedure
            u_gb  = u_sg - 0.1;                 % intial value
            x0    = u_gb;                       % iteration variable
            y0    = [u_b, u_sg, u_T, C_0, u_sl, 1]; % iteration constants
            f     = @(x0)fCalc_u_gb(x0,y0);
            u_gb  = fsolve(f,x0,options);
            u_gu  = u_b * (1-exp(-0.1*u_gb/(u_sg-u_gb))) + u_T *(exp(-0.1*u_gb/(u_sg-u_gb)));
                                        % drift-flux velocity for transition from slug to churn
            eps_G = u_sg / (C_0*u_m+u_gu); % void fraction for transition from slug to churn
            FP = 'churn';
        elseif (u_m > u_mc) && (u_m < u_ms) && (u_sg < u_gc) % churn flow
            C_0   = C_0s*(1-exp(-0.1*u_mc/(u_m-u_mc))) + C_0c*(exp(-0.1*u_mc/(u_m-u_mc)));
                                % distribution parameter for transition from slug to churn flow
            % iterative procedure
            u_gb  = u_sg - 0.1;                 % intial value
            x0    = u_gb;                       % iteration variable
            y0    = [u_b, u_sg, u_T, C_0, u_sl, 1]; % iteration constants
            f     = @(x0)fCalc_u_gb(x0,y0);
            u_g  = fsolve(f,x0,options);
            u_gu  = u_b * (1-exp(-0.1*u_gb/(u_sg-u_gb))) + u_T *(exp(-0.1*u_gb/(u_sg-u_gb)));
                                        % drift-flux velocity for transition from slug to churn
            eps_G = u_sg / (C_0*u_m+u_gu); % void fraction for transition from slug to churn
            FP = 'churn';
        elseif (u_m > u_ms) && (u_sg < 1.08*u_sl) && (u_sg < u_gc)% dispersed bubble flow
            C_0 = C_0cdb;
            % C_0 =  1 + 0.2*(1-chi);
            % C_0 = 1.1;
            u_gu = u_b;
            eps_G = u_sg / (C_0 * u_m + u_gu);
            FP = 'dispersed bubble';
        elseif (u_sg < u_gc) % slug flow
            FP = 'slug';
        elseif (u_sg > u_gc) || eps_G >= 0.7
            C_0 = C_0c*(1-exp(-0.1*u_gc/(u_sg-u_gc))) + C_0a*(exp(-0.1*u_gc/(u_sg-u_gc)));
          % distribution parameter for transition from churn to annular flow (Hasan & Kabir, 2010)
            %C_0_B = 1 + 0.2*(1-chi); % Rouhani & Axelsson (1970)
            %C_0 = C_0_B;
            %C_0 = 1.1;
            %u_gu = u_T * (1-exp(-1*(u_gc)/(u_sg-(u_gc))));
            eps_G = u_sg / (C_0 * u_m); % void fraction annular flow
            FP = 'annular';
        end
%         if  eps_G >= 0.7
%             C_0 = C_0c*(1-exp(-0.1*u_gc/(u_sg-u_gc))) + C_0a*(exp(-0.1*u_gc/(u_sg-u_gc)));
%             eps_G = u_sg / (C_0 * u_m); % void fraction annular flow
%             FP = 'annular';
%         end
    end
    %% Dix (1971)
    elseif DF_model == 4;

        C_0    = (u_sg/u_m) * (1 + ((u_sl/u_sg)^((rho_g/rho_l)^0.1))); % Distribution parameter
        u_gu   = 2.9 *(g *sigma*(rho_l-rho_g)/(rho_l^2))^0.25;          % drift-flux velocity [m/s]
        eps_G  = u_sg / (C_0 * u_m + u_gu);                             % void fraction [-]

    %% Nicklin (1965)
    elseif DF_model == 5
        C_0    = 1.2;                                                   % Distribution parameter
        u_gu   = 0.35 * (g * D_i)^0.5;                                  % drift-flux velocity [m/s]
        eps_G  = u_sg / (C_0 * u_m + u_gu);                             % void fraction [-]

    %% Toshiba (1989)
    elseif DF_model == 6

        C_0    = 1.08;                                                  % Distribution parameter
        u_gu   = 0.45;                                                  % drift-flux velocity [m/s]
        eps_G  = u_sg / (C_0 * u_m + u_gu);                             % void fraction [-]

    end

    %% check flow pattern for drift-flux models other than Hasan and Kabir (2010)
    if DF_model == 2 || DF_model > 3

        % Check bubble pattern
        rho_m = eps_G * rho_g + (1-eps_G) * rho_l;  % density mixture [kg/m3]
        Re_m  = rho_m * u_m * D_i / mu_m;           % Reynolds number mixture [-]
        f_DW  = 0.25/(log10((eps_pipe/D_i/3.7065)-((5.0452/Re_m)*log10(((1/2.8257)*...
                (eps_pipe/D_i)^1.1098)+(5.8506/Re_m^0.8981))))^2);
                                                % friction factor [-] (Hasan and Kabir, 2010)
        % f_DW = 0.046*Re_m^-0.2;               % friction factor (Taitel et al., 1980)
        u_ms  = ((0.725 + 4.15*(u_sl/u_m)^0.5)/((((f_DW/2/D_i)^0.4)*((rho_l/sigma)^0.6)*...
                (((0.4*sigma)/(g*(rho_l-rho_g)))^0.5)*2)))^(1/1.2);
                      % minimum mixture velocity for dispersed bubble flow (Hasan and Kabir, 2010)
        % u_ms = 4 * (D_i^0.429*(sigma/rho_l)^0.089/mu_kin^0.072)*((g*(rho_l-rho_g)/rho_l)^0.446);
                      % minimum mixture velocity for dispersed bubble flow (Taitel et al., 1980)
        u_gc  = 3.1 * (g*sigma*(rho_l-rho_g)/(rho_g)^2)^0.25;
                                        % superficial gas velocity for transition to annular flow
        if eps_G < 0.25 && (u_sg < u_gc)     % bubble or dispersed flow
            if (u_m > u_ms) && (u_sg < u_gc)
                FP = 'dispersed bubble';
            elseif (u_sg < u_gc)
                FP = 'bubble';
            end
        else % dispersed bubble, slug, churn or annular flow
            % u_mc = ((0.725 + 4.15*(u_sl/u_m)^0.5)/((((f_DW/2/D_i)^0.4)*((rho_l/sigma)^0.6)*...
            %        (((0.4*sigma)/(g*(rho_l-rho_g)))^0.5)*2)))^(1/1.2);
                    % mixture velocity for transition from slug to churn flow (Hasan and Kabir, 2010)
            u_mc = (l_E/D_i/40.6 - 0.22) * (g * D_i)^0.5;
                    % mixture velocity for transition from slug to churn flow (Taitel et al, 1980)
            if (u_m > u_mc) && (u_sg > 1.08*u_sl) && (u_sg < u_gc)
                FP = 'churn';
            elseif (u_m > u_mc) && (u_m < u_ms) && (u_sg < u_gc) % churn
                FP = 'churn';
            elseif (u_m > u_ms) && (u_sg < 1.08*u_sl) && (u_sg < u_gc) % dispersed bubble
                FP = 'dispersed bubble';
            elseif (u_sg < u_gc)
                FP = 'slug';
            elseif (u_sg > u_gc)
                FP = 'annular';
            else
                FP = 'annular';
            end
        end
    end

end
```

```
%% fCalc_u_gb

% Calculation of velocity neededd for transition from bubbly to slug flow
% Frank Niewold
% Released version 1.0, February 2017

function [F] = fCalc_u_gb(x,y)

    %% transition from slug to churn flow
    if y(6) == 1

        u_gb_it = x(1);
        u_b     = y(1);
        u_sg    = y(2);
        u_T     = y(3);
        C_0     = y(4);
        u_sl    = y(5);
```

```
        u_gu = u_b * (1-exp(-0.1*u_gb_it/(u_sg-u_gb_it))) + u_T *(exp(-0.1*u_gb_it/(u_sg-u_gb_it)));
        u_gb = (C_0*u_sl+u_gu) / (4-C_0);
        F    = u_gb - u_gb_it;

    %% transition from bubble to slug flow
    else

        u_gb_it = x(1);
        u_b     = y(1);
        u_sg    = y(2);
        u_T     = y(3);
        C_0b    = y(4);
        C_0s    = y(5);
        u_sl    = y(6);

        u_gu = u_b * (1-exp(-0.1*u_gb_it/(u_sg-u_gb_it))) + u_T *(exp(-0.1*u_gb_it/(u_sg-u_gb_it)));
        C_0  = C_0b*(1-exp(-0.1*u_gb_it/(u_sg-u_gb_it))) + C_0s*(exp(-0.1*u_gb_it/(u_sg-u_gb_it)));
        u_gb = (C_0*u_sl+u_gu) / (4-C_0);
        F    = u_gb - u_gb_it;
    end
end
```

```
%% fCalc_u_ms

% Calculation of the minimum mixture velocity
% Frank Niewold
% Released version 1.0, February 2017

function [F] = fCalc_u_ms(x,y)
    %% script drift-flux_model
    if y(13) == 1

        u_sl     = x(1);
        f_DW     = y(1);
        D_i      = y(2);
        rho_l    = y(3);
        sigma    = y(4);
        g        = y(5);
        rho_g    = y(6);
        u_sg     = y(7);
        mu_g     = y(8);
        mu_l     = y(9);
        C_0b     = y(10);
        u_b      = y(11);
        eps_pipe = y(12);

        u_m   = u_sl + u_sg;
        eps_G = u_sg / (C_0b*u_m + u_b);
        chi   = (u_sg*pi*(D_i/2)^2*rho_g) / ((u_sg*pi*(D_i/2)^2*rho_g) + (u_sl*pi*(D_i/2)^2*rho_l));
        mu_m  = chi * mu_g + (1-chi) * mu_l;
        rho_m = eps_G * rho_g + (1-eps_G)* rho_l;
        Re_m  = rho_m * u_m * D_i / mu_l;

        % Hasan & Kabir (2010)
        f_DW = 0.25/(log10((eps_pipe/D_i/3.7065)-((5.0452/Re_m)*log10(((1/2.8257)*(eps_pipe/D_i)...
               ^1.1098)+(5.8506/Re_m^0.8981))))^2);
        u_ms = ((0.725 + 4.15*(u_sl/u_m)^0.5)/((((f_DW/2/D_i)^0.4)*((rho_l/sigma)^0.6)*...
               (((0.4*sigma)/(g*(rho_l-rho_g)))^0.5)*2)))^(1/1.2);
        F    = u_m - u_ms;

    %% function fCalc_eps_G
    else

        u_ms_it = x(1);
        u_sl    = y(1);
        f_DW    = y(2);
        D_i     = y(3);
        rho_l   = y(4);
        sigma   = y(5);
        g       = y(6);
        rho_g   = y(7);

        u_ms = ((0.725 + 4.15*(u_sl/u_ms_it)^0.5)/((((f_DW/2/D_i)^0.4)*((rho_l/sigma)^0.6)*...
               (((0.4*sigma)/(g*(rho_l-rho_g)))^0.5)*2)))^(1/1.2);
        F    = u_ms - u_ms_it;

    end
```

```
end
```

```
%% fCalc_u_mc

% Calculation of superficial gas velocity corresponding to eps_G = 0.25
% Frank Niewold
% Released version 1.0, December 2016

function [F] = fCalc_u_mc(x,y)
    if y(5) == 1
        u_sg = x(1);
        l    = y(1);
        g    = y(2);
        D_i  = y(3);
        u_sl = y(4);

        u_m  = u_sg + u_sl;
        u_mc = (1/40.6 - 0.22) * (g*D_i)^0.5; % Taitel et al. (1980)
        F    = u_m - u_mc;

    else

        u_gb_it = x(1);
        u_sg    = x(2);
        u_b     = y(1);
        u_T     = y(2);
        C_0b    = y(3);
        C_0s    = y(4);
        u_sl    = y(5);
        D_i     = y(6);
        rho_g   = y(7);
        rho_l   = y(8);
        mu_l    = y(9);
        eps_pipe= y(10);
        g       = y(11);
        sigma   = y(12);
        mu_g    = y(13);

        u_gu  = u_b * (1-exp(-0.1*u_gb_it/(u_sg-u_gb_it))) + u_T *(exp(-0.1*u_gb_it/(u_sg-u_gb_it)));
        C_0   = C_0b*(1-exp(-0.1*u_gb_it/(u_sg-u_gb_it))) + C_0s*(exp(-0.1*u_gb_it/(u_sg-u_gb_it)));
        u_gb  = (C_0*u_sl+u_gu) / (4-C_0);
        u_m   = u_sg + u_sl;
        % C_0_B =  1 + 0.2*(1-x); % Rouhani & Axelsson (1970)
        % C_0(i,j) = min(C_0_A,C_0_B);
        eps_G = u_sg / (C_0*(u_m) + u_gu);
        rho_m = eps_G * rho_g + (1-eps_G)* rho_l;
        chi   = (u_sg*pi*(D_i/2)^2*rho_g) / ((u_sg*pi*(D_i/2)^2*rho_g) + (u_sl*pi*(D_i/2)^2*rho_l));
        mu_m  = chi * mu_g + (1-chi) * mu_l;
        Re_m  = rho_m * u_m * D_i / mu_m;
        f_DW  = 0.25/(log10((eps_pipe/D_i/3.7065)-((5.0452/Re_m)*log10(((1/2.8257)*...
                (eps_pipe/D_i)^1.1098)+(5.8506/Re_m^0.8981))))^2);
        u_mc  = ((0.725 + 4.15*(u_sl/u_m)^0.5)/((((f_DW/2/D_i)^0.4)*((rho_l/sigma)^...
                0.6)*(((0.4*sigma)/(g*(rho_l-rho_g)))^0.5)*2)))^(1/1.2); % Hasan and Kabir (2010)

        F(1)  = u_gb_it - u_gb;
        F(2)  = u_m - u_mc;

    end
end
```

C

MODELING COMPONENTS

C.1. Model Input - MS Excel Interface

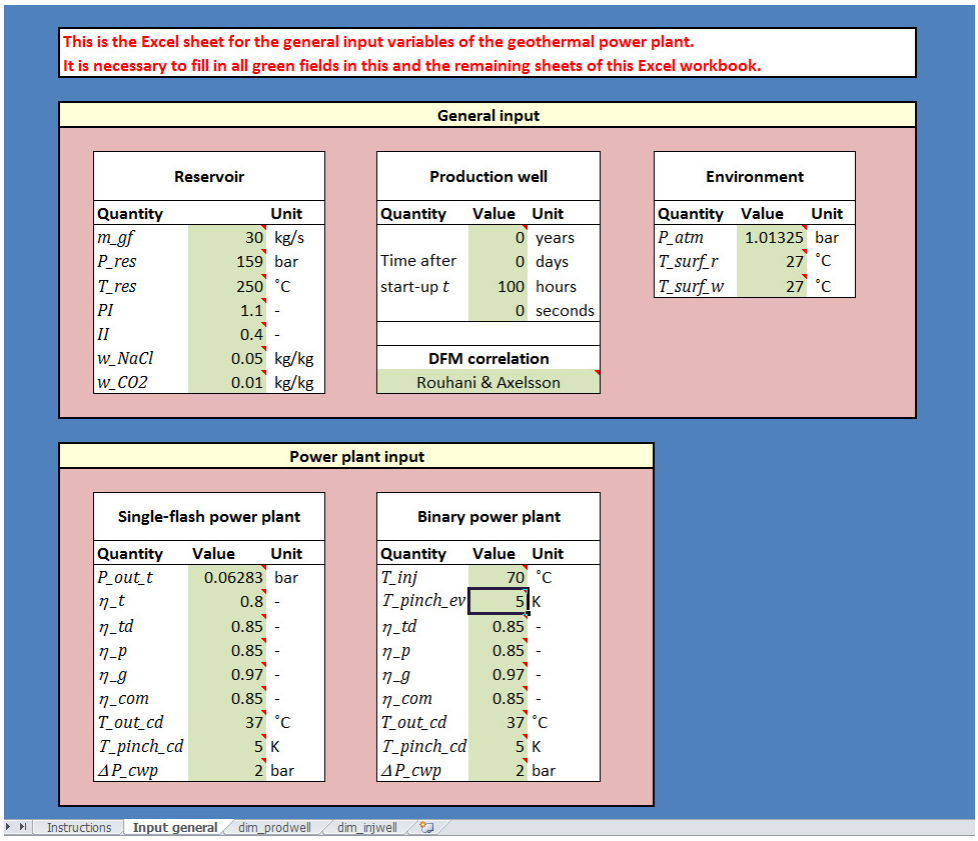


Figure C.1: MS Excel interface for user-defined model input. Sheet Input_general. In this sheet the general input variables for the sub models are entered.

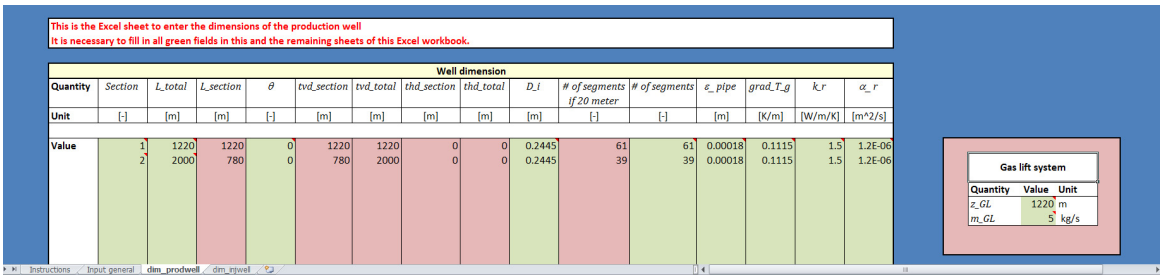
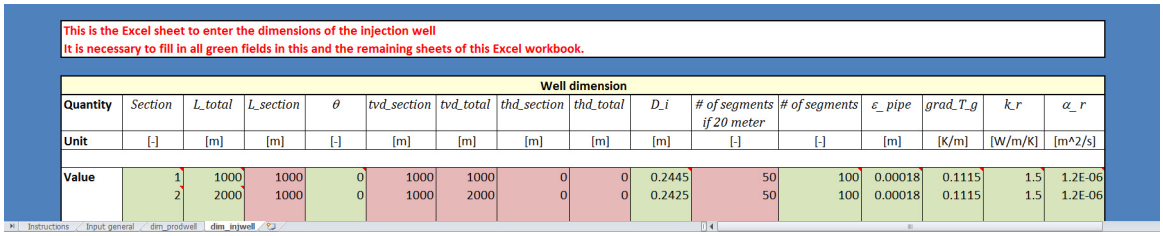


Figure C.2: MS Excel interface for user-defined model input. Sheet dim_prodwell_SF. In this sheet the dimensions for the production well for a single-flash power plant are entered.



This is the Excel sheet to enter the dimensions of the injection well
It is necessary to fill in all green fields in this and the remaining sheets of this Excel workbook.

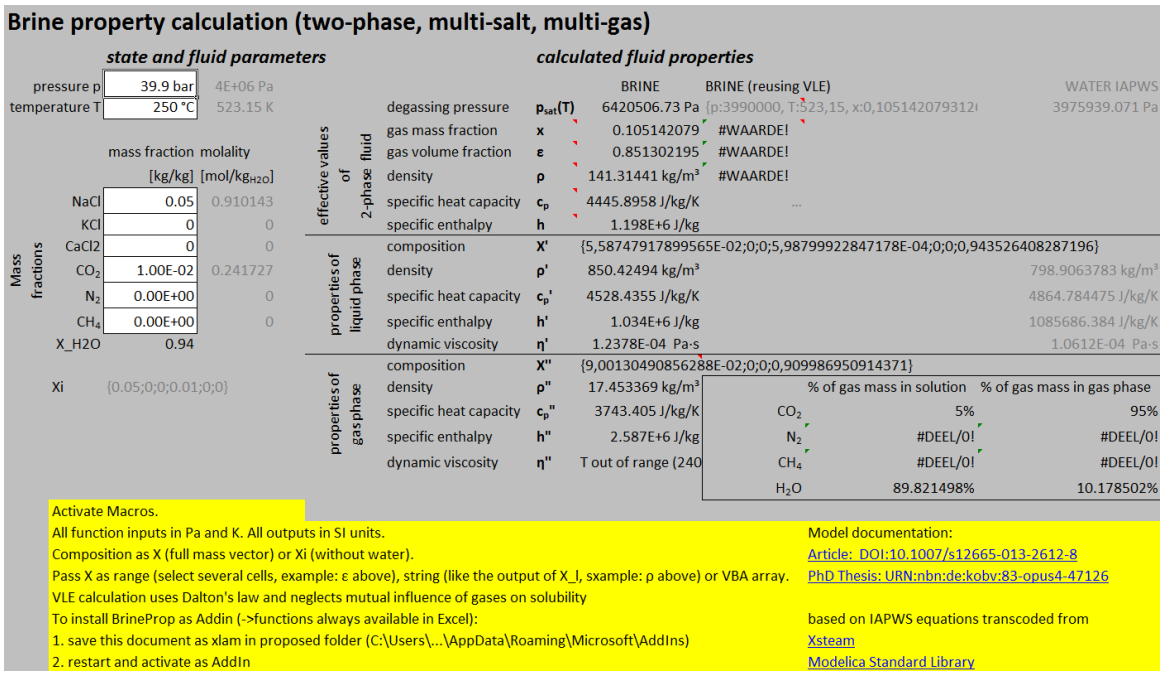
Well dimension

Quantity	Section	L_total	L_section	θ	tvd_section	tvd_total	thd_section	thd_total	D_i	# of segments if 20 meter	# of segments	ε_pipe	grad_T_g	k_r	α_r
Unit	[-]	[m]	[m]	[-]	[m]	[m]	[m]	[m]	[m]	[-]	[-]	[m]	[K/m]	[W/m/K]	[m^2/s]
Value	1	1000	1000	0	1000	1000	0	0	0.2445	50	100	0.00018	0.1115	1.5	1.2E-06
	2	2000	1000	0	1000	2000	0	0	0.2425	50	100	0.00018	0.1115	1.5	1.2E-06

Instructions | Input general | dim_prodwell | dim_injwell

Figure C.3: MS Excel interface for user-defined model input. Sheet dim_injwell. In this sheet the dimension for the injection well are entered.

C.2. Interface GFP Excel Model



Brine property calculation (two-phase, multi-salt, multi-gas)

state and fluid parameters

pressure p	39.9 bar	4E+06 Pa
temperature T	250 °C	523.15 K

Mass fractions	mass fraction [kg/kg]	molality [mol/kg$_{H2O}$]
NaCl	0.05	0.910143
KCl	0	0
CaCl2	0	0
CO_2	1.00E-02	0.241727
N_2	0.00E+00	0
CH_4	0.00E+00	0
X_H2O	0.94	

Xi {0.05;0;0;0.01;0;0}

calculated fluid properties

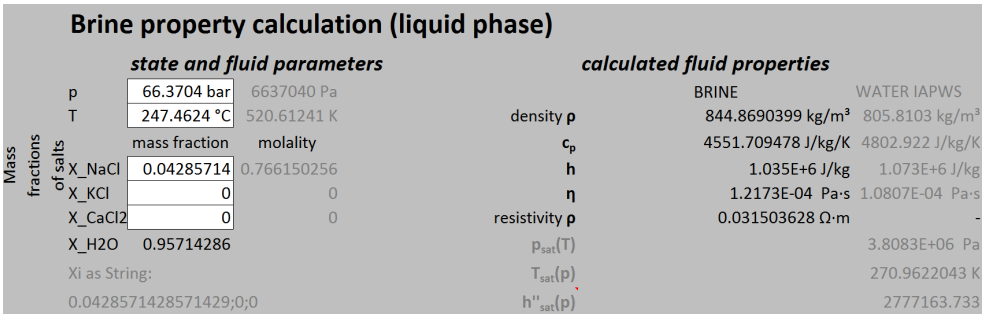
			BRINE	BRINE (reusing VLE)	WATER IAPWS
effective values of 2-phase fluid	degassing pressure	$p_{sat}(T)$	6420506.73 Pa	{p:3990000, T:523,15, x:0,105142079312	3975939.071 Pa
	gas mass fraction	x	0.105142079	#WAARDE!	
	gas volume fraction	ε	0.851302195	#WAARDE!	
	density	ρ	141.31441 kg/m³	#WAARDE!	
	specific heat capacity	c_p	4445.8958 J/kg/K	...	
	specific enthalpy	h	1.198E+6 J/kg		
properties of liquid phase	composition	X'	{5,58747917899565E-02;0;0;5,98799922847178E-04;0;0;0,943526408287196}		
	density	ρ'	850.42494 kg/m³		798.9063783 kg/m³
	specific heat capacity	c_p'	4528.4355 J/kg/K		4864.784475 J/kg/K
	specific enthalpy	h'	1.034E+6 J/kg		1085686.384 J/kg/K
	dynamic viscosity	η'	1.2378E-04 Pa·s		1.0612E-04 Pa·s
properties of gas phase	composition	X''	{9,00130490856288E-02;0;0;0,909986950914371}		
	density	ρ''	17.453369 kg/m³	% of gas mass in solution	% of gas mass in gas phase
	specific heat capacity	c_p''	3743.405 J/kg/K	CO_2 5%	95%
	specific enthalpy	h''	2.587E+6 J/kg	N_2 #DEEL/0!	#DEEL/0!
	dynamic viscosity	η''	T out of range (240	CH_4 #DEEL/0!	#DEEL/0!
				H_2O 89.821498%	10.178502%

Activate Macros.
All function inputs in Pa and K. All outputs in SI units.
Composition as X (full mass vector) or Xi (without water).
Pass X as range (select several cells, example: ε above), string (like the output of X_l, example: ρ above) or VBA array.
VLE calculation uses Dalton's law and neglects mutual influence of gases on solubility
To install BrineProp as Addin (->functions always available in Excel):
1. save this document as xlam in proposed folder (C:\Users\...\AppData\Roaming\Microsoft\AddIns)
2. restart and activate as AddIn

Model documentation:
Article: DOI:10.1007/s12665-013-2612-8
PhD Thesis: URN:nbn:de:kobv:83-opus4-47126

based on IAPWS equations transcoded from
Xsteam
Modelica Standard Library

Figure C.4. MS Excel interface of the thermodynamic model for two-phase flow from Heineken (2016).



Brine property calculation (liquid phase)

state and fluid parameters

p	66.3704 bar	6637040 Pa
T	247.4624 °C	520.61241 K

Mass fractions of salts	mass fraction	molality
X_NaCl	0.04285714	0.766150256
X_KCl	0	0
X_CaCl2	0	0
X_H2O	0.95714286	

Xi as String:
0.0428571428571429;0;0

calculated fluid properties

	BRINE	WATER IAPWS
density ρ	844.8690399 kg/m³	805.8103 kg/m³
c_p	4551.709478 J/kg/K	4802.922 J/kg/K
h	1.035E+6 J/kg	1.073E+6 J/kg
η	1.2173E-04 Pa·s	1.0807E-04 Pa·s
resistivity ρ	0.031503628 Ω·m	-
$p_{sat}(T)$		3.8083E+06 Pa
$T_{sat}(p)$		270.9622043 K
$h''_{sat}(p)$		2777163.733

Figure C.5: MS Excel interface of the thermodynamic model for the gas phase from Heineken (2016).

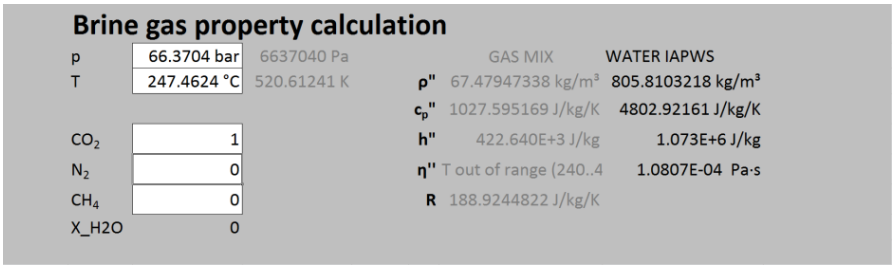


Figure C.6: MS Excel interface of the thermodynamic model for the liquid phase from Heineken (2016).

C.3. Degassing Pressures of Duan and Sun (2003)

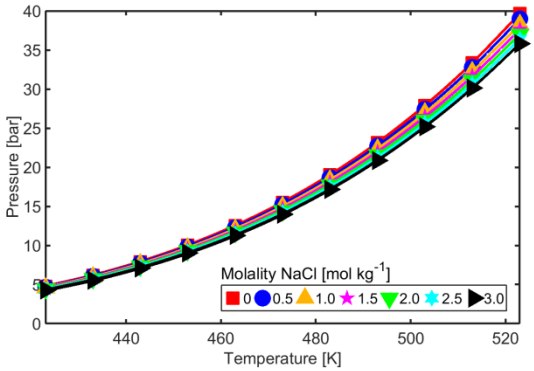


Figure C.7: Degassing pressures as a function of temperature for various NaCl molalities and $m_{CO2} = 0$ mol kg^{-1} (Duan and Sun, 2003).

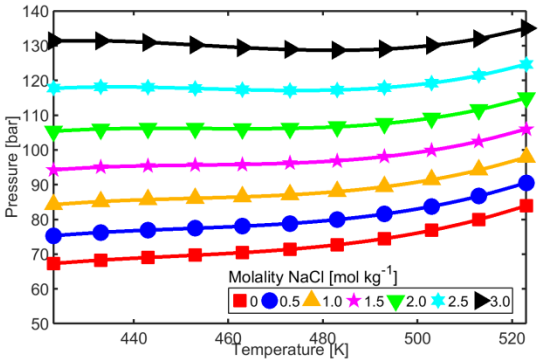


Figure C.8: Degassing pressures as a function of temperature for various NaCl molalities and $m_{CO2} = 0.5$ mol kg^{-1} (Duan and Sun, 2003).

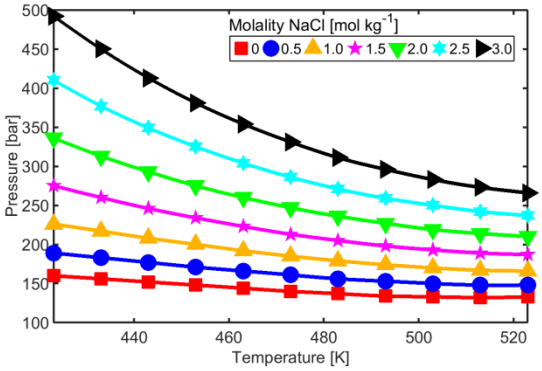


Figure C.9: Degassing pressures as function of temperature for various NaCl molalities and $m_{CO2} = 1.0$ mol kg^{-1} (Duan and Sun, 2003).

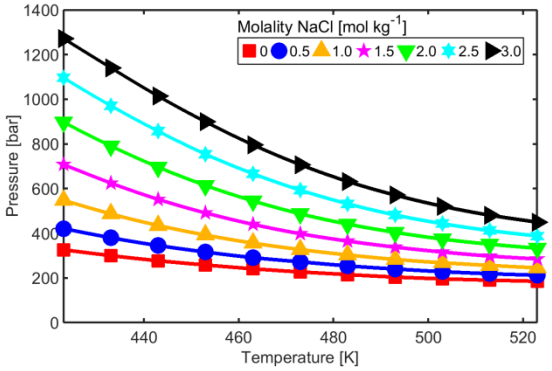


Figure C.10: Degassing pressures as function of temperature for various NaCl molalities and $m_{CO2} = 1.5$ mol kg^{-1} (Duan and Sun, 2003).

D

ADDITIONAL CALCULATIONS

D.1. Single-Flash Power Plant Model

The numbers in the equations in the present section correspond to the numbers in Figure 2.21 and Figure 2.24.

Cyclone separator:

The calculation starts with the cyclone separator (CS) with input from the top of the production well model. It invokes *fCalc_geofprops2* to calculate the relevant gas and liquid properties of state 3 and 4. Isenthalpic flashing is assumed (eq. (2.53)).

The GFP Excel model is not able to calculate geothermal fluid properties below a pressure of 1 bar. The pressure at the outlet of the steam turbine can be below 1 bar. Therefore, after the CS the single-flash power plant model calculates the geothermal fluid properties from implemented data tables, covering $(P, T, h, s)_{CO2}$ and $(P, T, h, s)_{H2O,sat}$ obtained from FluidProp (Span and Wagner, 1996; Wagner and Pruss, 2002). At the inlet of the steam turbine (state 4) the enthalpy $h_{mix,4}$ and entropy $s_{mix,4}$ of the H_2O – CO_2 gas mixture is calculated. This involves determining the mole fractions $x_{i,4}$, eqs. (A.58) and (A.59), and partial pressures $P_{i,4}$, eq. (2.73), of the components, where i indicates the component (CO_2, H_2O). Then h and s of the single components are interpolated from the data tables. Finally, $h_{mix,4}$ and $s_{mix,4}$ are calculated by eqs. (2.71) and (2.72), respectively.

$$x_{H2O,4} = \frac{w_{H2O,4}/M_{H2O}}{w_{H2O,4}/M_{H2O} + w_{CO2,4}/M_{CO2}} \tag{A.58}$$

$$x_{CO2,4} = \frac{w_{CO2,4}/M_{CO2}}{w_{H2O,4}/M_{H2O} + w_{CO2,4}/M_{CO2}} \tag{A.59}$$

Steam turbine:

Next, in order to calculate the outlet properties of the steam turbine, isentropic expansion is assumed initially. User-defined model input is required, given in Figure 3.1 . A function *fCalc_chi_5s* is developed, which calculates χ_{5s} and thereby the other fluid properties which are a function of P, T and χ. This involves an iterative procedure in which χ_{5s} and T_{5s} are iterated until eq. (2.74) for isentropic expansion and eq. (A.60), in which the temperature as a function of the partial pressure of H_2O equals the iterated T_{5s}, are solved. The $s_{mix,5s}$ in eq. (2.74) is calculated by eq. (2.76). The $(h, s)_{6,7}$ for CO_2 and H_2O, necessary for this iteration, are interpolated from the implemented data tables. States 6 and 7 correspond to the liquid and vapor saturation of H_2O, respectively.

$$T(P_{H2O,7}) = T_{5s,calc} \tag{A.60}$$

The procedure of calculating χ_5 shows much resemblance with the calculation of χ_{5s}. Again an iterative procedure is involved in which now χ_5 and T_5 are iterated, until $h_{mix,5}$ from eq. (2.77) equals $h_{mix,5}$ from eq. (A.61) and $T_{5,calc}$ equals the temperature belonging to $P_{H2O,7}$ at state 5 (eq. (A.62)). However, a nested loop is implemented in this case. Because eq. (2.77) contains η_t, which is a function of χ_5 itself. Finally, $\dot{W}_t$ and $\dot{W}_e$ are calculated by eq. (2.55) and eq. (2.57), respectively.

$$h_{mix,5} = (1-\chi)h_{H2O}(T) + \chi \sum_{i=1}^{n} w_i h_i(P_i, T) \tag{A.61}$$

$$T(P_{H2O,7}) = T_{5,calc} \tag{A.62}$$

Condenser and SE/C:

The temperature at which the gas is extracted from the condenser is user-defined in the model input as T_{11}. Then the partial pressure $P_{H2O,11}$ is calculated by eq. (2.79) and the partial pressure $P_{CO2,11}$ is calculated by eq. (A.63), which is derived from eq. (2.80). These partial pressures can be related to the mole fractions according to Dalton's Law. Subsequently, the mass fractions are calculated with the molar masses and $\dot{m}_{11}$ is calculated by solving the mass balance for CO_2 given by eq. (2.81). The $h_{mix,11}$ and $s_{mix,11}$ is then obtained from eq. (2.71) and eq. (2.72) respectively, but then adjusted to state 11, where the single component properties are interpolated from the data tables.

$$P_{11,CO2} = P_5 + P_{11,H2O} \tag{A.63}$$

The SE/C has been modeled as two SE/C in series. According to the HEI the normal range of suction pressures for a two stage SE/C operating at atmospheric discharge pressures is between 0.002 – 0.135 bar (Coker and Coker, 2010). The outlet pressure of the steam turbine in this study is approximately 0.74 bar. The calculation procedure of the SE/C has been outlined in Section 2.4.4.3. The pressure ratio of both stages is equal.

Condenser pump:

The ρ_6 is interpolated from data tables. Then the power required by the pump is calculated by eq. (A.64) derived from eq. (2.60) and (A.65).

$$\dot{W}_{cp} = \frac{(1/\rho_6)[P_8 - P_6]}{\eta_p}\dot{m}_6 \tag{A.64}$$

$$P_8 = P_3 \tag{A.65}$$

Next, a provisional power is calculated, which is the $\dot{W}_{ip}$ added to the $\dot{W}_{net}$. The $\dot{W}_{ip}$ cannot be calculated at this stage of the simulation, because it needs the input from the injection well model. Therefore, at first it is checked if the provisional power of the current iteration is higher than the provisional power of the previous iteration. If so, the flash pressure P_2 is reduced with ΔP and the procedure proceeds with the next iteration. ΔP is a user-defined variable, the default value is 0.1 bar.

Injection pump:

Before the injection pump the streams 3, 8 and 14 are joined. Using the composition and the mass flow of the fluid at state 3, the composition of the fluid at state 9 can be calculated. The mass flow pumped by the make-up pump equals $\dot{m}_{11}$. Data tables $(P, T, \rho, c_p)_{H2O,SC}$ for subcooled H_2O are implemented to interpolate $\rho_{13}(P_{atm}, T_{surf\_w})$ at state 13 and c_p at state 8 and state 14. The T_8 and T_{14} are iterated until the assumption of incompressible liquid is met. The $c_{p,3}$ is calculated by invoking the GFP Excel model. Finally, the T_9 is given by eq. (A.66). At this stage, the single-flash power plant model simulation terminates. The output $(T, w)_9$ is exported to the injection well model. For the calculation procedure of the injection well model is referred to Section 3.7. To finalize the simulation the single-flash power plant model imports output $(P, \rho)_{10}$ from the injection well model. The $\dot{W}_{ip}$ is calculated by eq. (A.67). In this particular case, the injection pump is assumed to operate isothermally. This has been chosen, because that iteration would involve the total injection well model. The calculation of the injection well model is a time-

consuming process, so iteration is rather avoided. In case of the hypothetical case (Chapter 5), the temperature increase was in the range of 1 – 2 ˚C. The error induced is negligible, which justifies the assumption. The $\dot{W}_{net}, \eta_{th}$ and η_u are obtained by using eqs. (2.61) – (2.64).

$$T_9 = \frac{\dot{m}_3 c_{p,3} T_3 + \dot{m}_8 c_{p,8} T_8 + \dot{m}_{14} c_{p,14} T_{14}}{\dot{m}_3 c_{p,3} + \dot{m}_8 c_{p,8} + \dot{m}_{14} c_{p,14}} \tag{A.66}$$

$$\dot{W}_{ip} = \frac{(1/\rho_{10})[P_{10} - P_9]}{\eta_p} \dot{m}_9 \tag{A.67}$$

E

Model Validation & Sensitivity Analysis

E.1. Mean Error and Standard Deviation Mean Error

The equations in the present section were obtained from Ambastha and Gudmundsson (1986a). The error of a calculated single data point has been defined as the difference between the calculated value and the measured value, given in eqs. (E.1) and (E.2) for pressure and temperature, respectively. The i represents the number of a specific data point.

$$er_{i,P} = |P_{calc} - P_{meas}| \tag{E.1}$$

$$er_{i,T} = |T_{calc} - T_{meas}| \tag{E.2}$$

Then the mean error of the calculated pressure and temperature profiles is calculated by eq. (E.3). The n represents the number of evaluated data points.

$$\overline{er} = \frac{\sum_{i=1}^{n} er_i}{n} \tag{E.3}$$

The standard deviation of the mean error is calculated by eq. (E.4).

$$\sigma_{er} = \left(\frac{\sum_{i=1}^{n} (er_i - \overline{er})^2}{n-1} \right)^{1/2} \tag{E.4}$$

E.2. Drift-Flux Model Hasan et al. (2010)

Figure E.1 and Figure E.2 show the pressure-, density-, vapor quality- and void fraction profiles, and the flow patterns of the East-Mesa 6-1 well calculated with the present model using the drift-flux model of Hasan et al. (2010). Subsequently, Figure E.3, Figure E.4 and Figure E.5 present the pressure-, density- and vapor quality profiles, and the flow patterns of the East-Mesa 6-1 well according to Chadha and Malin (1993). It can be seen that flash depth and bubble flow region in the present model and in Chadha and Malin (1993) show a resemblance. However, shallower in the well totally different flow patterns have been observed. This does not necessarily means that one of two models is wrong. The density in Figure E.4 shows a sharp decrease at the slug 1/slug 2 transition. This particular concept of different slug patterns has not been encountered in other literature. Additionally, the characteristics of slug 2- and transition flow pattern in Chadha and Malin (1993) lies closer to annular flow than slug flow in Hasan et al. (2010), which shows it is more a difference in terminology. Figure E.4 also shows by the sharp transitions in Chadha and Malin (1993) smoothening between flow patterns has not been taken into account. Also, the pressure profile (Figure E.3) and vapor quality profile (Figure E.5) show a sharp transition especially between slug 1/slug 2 transition, most likely caused by the density change. In the present model these transitions are relatively smoother, resulting in a smooth pressure profile (Figure E.1) and smooth quality profile (Figure E.2).

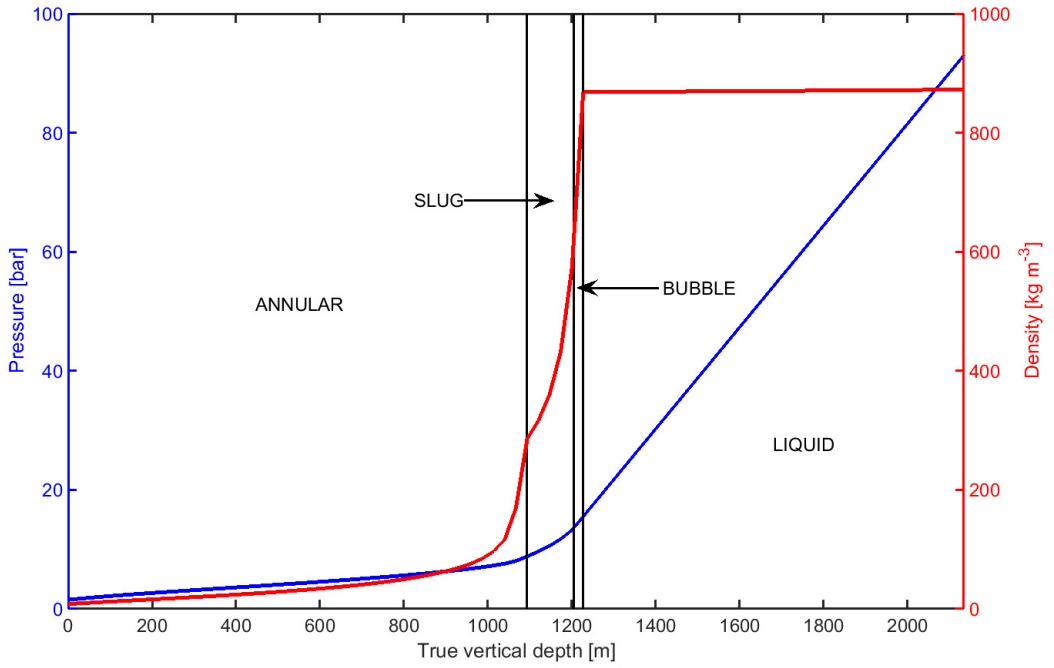


Figure E.1: Pressure profile (left y-axis) and density profile (right y-axis) of the East-Mesa 6-1 well, described in Section 4.1. Three flow patterns were observed: bubble, slug and annular.

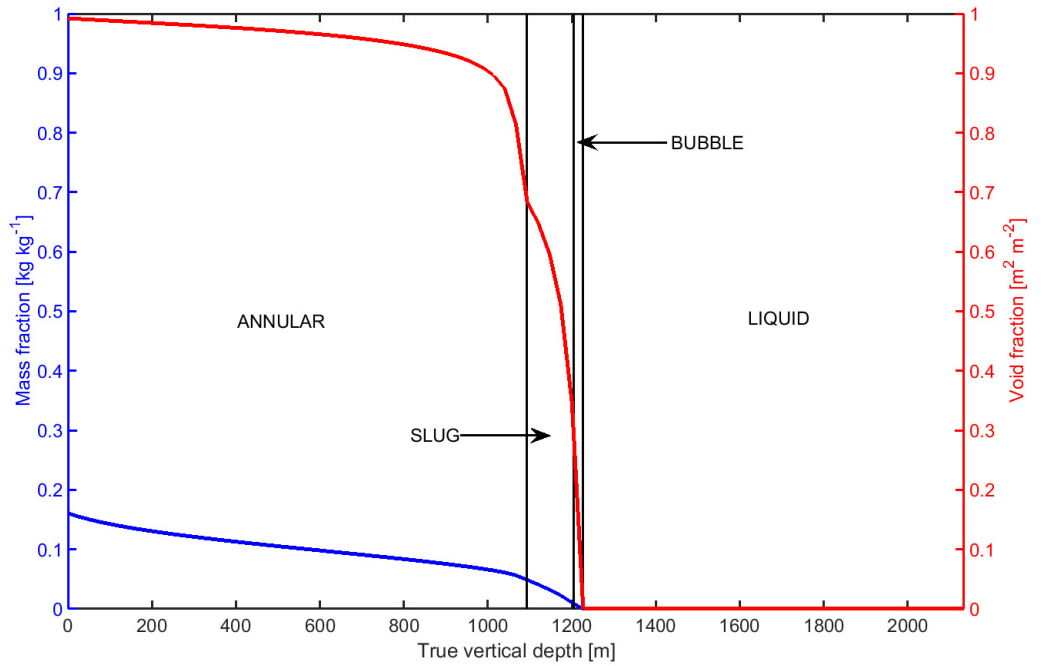


Figure E.2: Vapor quality (left y-axis) and void fraction (right y-axis) of the East-Mesa 6-1 well, described in Section 4.1. Three flow patterns were observed: bubble, slug and annular.

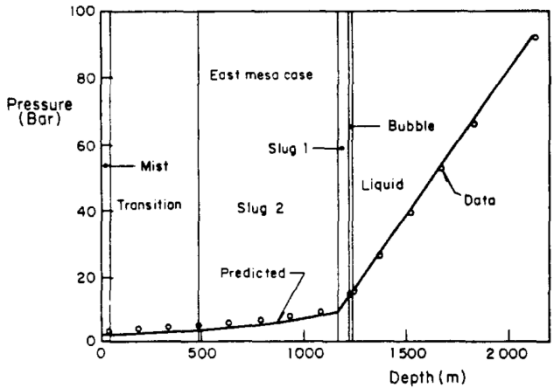


Figure E.3: Pressure profile of the East-Mesa 6-1 well (Chadha and Malin, 1993).

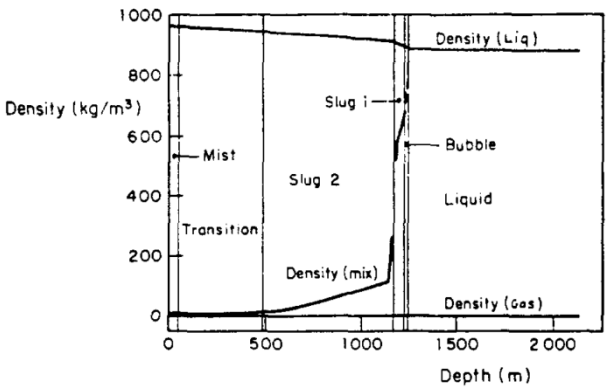


Figure E.4: Density profile of the East-Mesa 6-1 well (Chadha and Malin, 1993).

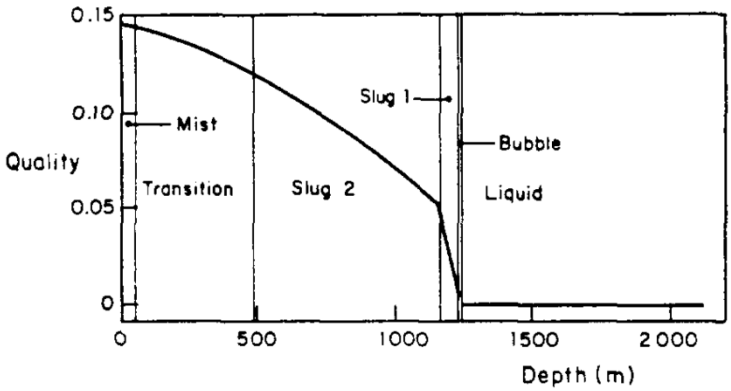


Figure E.5: Vapor quality profile of the East-Mesa 6-1 well (Chadha and Malin, 1993).

E.3. Validation Single-Flash Power Plant

E.3.1. Validation of Thermal Efficiency

Table E.1: Model input parameters for the validation of the single-flash power plant model.

Quantity	Value
Turbine outlet pressure, bar	Variable
Initial wet turbine efficiency	0.8
Dry turbine efficiency	0.85
Pump efficiency	0.85
Generator efficiency	0.97
Condenser outlet temperature, ˚C	37
Compressor efficiency	0.85
Pressure build-up cooling water pump, bar	2
Pinchpoint temperature condenser, K	5

Table E.2: Technical specifications of five single-flash power plants used for the single-flash power plant model validation. The fluid properties at the inlet of the power plant correspond to the wellhead fluid properties obtained from the production well model validation performed in Section 4.1.

Power plant/production well	East-Mesa 6-1	Ngawha 11	NWS-1 Sabalan-2	W2	W3
Power plant type	Single-flash	Single-flash	Single-flash	Single-flash	Single-flash
Mass flow rate steam, kg s^{-1}	12.9	6.60	30	32.8	50
NCG mass fraction steam, wt%	0	6.53	2.47	15.8	51.2
Turbine:					
Inlet pressure, bar	1.52	3.08	4.31	3.01	6.30
Inlet temperature, ˚C	111.8	134.2	146.4	133.4	160.6
Steam mass flow rate, kg s^{-1}	2.07	1.24	3.92	6.22	11.7
Exhaust pressure, bar	0.063	0.100	0.090	0.130	0.300
Condenser					
Cooling water (CW) flow, kg s^{-1}	214.6	112.8	377.8	498.7	499.9
Inlet temperature CW, ˚C	27.0	27.0	27.0	27.0	27.0
Outlet temperature CW, ˚C	32.0	32.0	32.0	32.0	32.0
Steam flow, kg s^{-1}	2.07	1.24	3.92	6.22	11.7
Steam enthalpy inlet condenser, kJ kg^{-1}	2318.9	2187.8	2234.0	2032.6	1369.0
Steam enthalpy outlet condenser, kJ kg^{-1}	155.0	287.7	220.8	357.8	474.5
Heat flow, MW	4.49	2.36	7.89	10.4	10.4
NCG system					
Steam ejector	no	no	yes	no	no
Stages	N/A	N/A	2	N/A	N/A
Motive steam flow, kg s^{-1}	N/A	N/A	0.92	N/A	N/A
Centrifugal compressor	no	yes	no	yes	yes
Plant performance					
Gross power, MW	0.78	0.49	1.78	2.21	3.39
Condenser pump power, MW	3.6E-4	3.9E-4	2.3E-3	1.7E-3	3.6E-3
CW pump power, MW	0.05	0.03	0.09	0.12	0.12
Centrifugal compressor power, MW	N/A	0.05	N/A	0.34	0.66
Production pump, MW [1]	N/A	N/A	N/A	N/A	N/A
Injection pump, MW [2]	N/A	N/A	N/A	N/A	N/A
Net power, MW	0.70	0.39	1.64	1.68	2.50
Thermal efficiency	6.42	6.18	5.64	5.30	5.21
Utilization efficiency [3]	32.8	28.9	26.1	24.6	25.9
SSC, kg s^{-1}/MW	2.94	3.18	2.39	3.70	4.68

[1] It is assumed that single-flash power plant do not require a production pump in the production well.

[2] The injection pump was neglected in this calculation, because it is a function of reservoir characteristics (PI, II). This is not particularly related to power plant performance.

[3] Based on a dead-state at 1.01 bar and 25 ˚C.

E.3.2. Validation of SE/C

Table E.3: Technical specifications of the Cerro Prieto I geothermal power plant (units 1 – 4) (Ocampo-Díaz et al., 2005; DiPippo, 2012). Data are averaged per unit. Technical specifications of the present model simulation. Green values were input, red values were output.

Technical data	Cerro Prieto I	Model simulation
Rating, MW	37.5	37.78
Mass flow rate steam, kg s^{-1}	85.93	85.93
NCG mass fraction steam, wt%	?	1.308
Turbine:		
Inlet pressure, bar	6.2	6.2
Inlet temperature, ˚C	160 (sat.)	160
Steam mass flow rate, kg s^{-1}	79.25	79.86
Exhaust pressure, bar	0.1185	0.1185
Condenser		
Cooling water (CW) flow, kg s^{-1}	2974	2974
Inlet temperature CW, ˚C	32.0	32.0
Outlet temperature CW, ˚C	45.3	45.3
Steam flow, kg s^{-1}		79.86
Steam enthalpy inlet condenser, kJ kg^{-1}		2256.5
Steam enthalpy outlet condenser, kJ kg^{-1}		185.33
Heat flow, MW	165.4	165.4
NCG extraction system		
Steam ejector	yes	yes
Stages	2	2
Steam flow, kg s^{-1}	6.68	6.07
Plant performance		
SSC, kg s^{-1}/MW	2.11	2.11

E.4. Validation Binary Cycle Power Plant

Table E.4: Model input parameters for the validation of the binary cycle power plant model. The injection temperature at the wellhead of the injection well was varied to validate the binary cycle power output as a function of injection temperature (Section 4.5.2).

Quantity	Value
Geothermal fluid	
Mass flow rate, kg s^{-1}	221
Pressure, bar	10.03 (sat.)
Temperature, ˚C	180 (sat.)
NaCl concentration, wt%	0
CO_2 concentration, wt%	0
Binary cycle power plant	
Injection temperature wellhead, ˚C	Variable
Pinchpoint temperature preheater/evaporator, K	5
Efficiency turbine dry	1
Efficiency pump	1
Efficiency generator	1
Temperature condenser, ˚C	40
Working fluid	Isopentane

E.5. Power Plant Model Sensitivity Analysis

Table E.5: Model input parameters for the full model of the single-flash power plant model. Geothermal fluid properties were adopted from the output of the simulation of the NWS-1 Sabalan-2 production well, described in Chapter 4.

Quantity	Value
Pressure turbine outlet, bar	0.0738
Initial efficiency turbine wet	0.8
Efficiency turbine dry	0.85
Efficiency pump	0.85
Efficiency generator	0.97
Temperature condenser outlet, ˚C	37
Pinchpoint temperature condenser, K	5

Table E.6: Model input parameters for the full model of the binary cycle power plant model. Geothermal fluid properties were adopted from the output of the simulation of the NWS-1 Sabalan-2 production well, described in Chapter 4.

Quantity	Value
Injection temperature wellhead, ˚C	70
Pinchpoint temperature evaporator/preheater, K	5
Efficiency turbine dry	0.85
Efficiency pump	0.8
Efficiency generator	0.97
Temperature condenser, ˚C	37
Working fluid	Isopentane

F

MODEL INPUT PARAMETERS SIMULATIONS

F.1. Model Input Parameters - Results

Table F.1: General model input parameters for the simulations performed in Chapter 5. The values in dark green were varied. The pressure at the turbine outlet was optimized to obtain the maximum power output for the single-flash power plant. In one scenario the injection temperature of the binary cycle power plant is equal to the injection temperature of the single-flash power plant with a SE/C (SF-1).

Quantity	Value
Reservoir	
Mass flow rate, kg s^{-1}	30
Pressure, bar	159
Temperature, ˚C	250
NaCl mass fraction, kg kg^{-1}	0.025, 0.050
CO_2 mass fraction, kg kg^{-1}	0, 0.005, 0.01, 0.015, 0.020, 0.025, 0.034
Production well	
Start-up time, h	100
Drift-flux model	Rouhani and Axelsson
Environment	
Pressure atmosphere, bar	1.01325
Temperature surface water, ˚C	27
Temperature rock earth's surface, ˚C	27
Single-flash power plant	
Pressure turbine outlet, bar	$P_{out,t,OPT}$
Initial efficiency turbine wet	0.80
Efficiency turbine dry	0.85
Efficiency pump	0.85
Efficiency generator	0.97
Efficiency compressor	0.85
Temperature condenser outlet, ˚C	37
Pinchpoint temperature condenser, K	5
Pressure change cooling water pump, bar	2
NCG extraction system	SE/C, centrifugal compressor
Binary cycle power plant	
Injection temperature wellhead, ˚C	$T_{inj,BC} = T_{inj,SF-1}$, 70
Pinchpoint temperature evaporator/preheater, K	5
Efficiency turbine dry	0.85

Table F.1 (Continued)

Quantity	Value
Efficiency pump	0.85
Efficiency generator	0.97
Efficiency compressor	0.85
Temperature condenser, ˚C	37
Pinchpoint temperature condenser, K	5
Pressure change cooling water pump, bar	2
Working fluid	Isopentane

Table F.2: Self-flowing production well, production well with gas lift and injection well model input parameters for the simulations performed in Chapter 5. The values in dark green were varied. The depth of the gas lift valve is equal to the flash depth of the self-flowing production well with corresponding model input parameters.

Quantity	Value
Production well / Injection well	
Length, m	2000
Inclination angle	0
Inner diameter, m	0.245
Number of segments	100
Segment length, m	20
Pipe roughness, m	1.8E-04
Geothermal gradient, K m^{-1}	0.1115
Thermal conductivity rock, W m^{-1} K^{-1}	1.5
Thermal diffusivity, m^2 s^{-1}	1.2E-06
Production well – gas lift	
Depth gas lift valve, m	$z_{GL} = z_{flash,SF-1}$
Mass flow rate gas, kg s^{-1}	0.5, 1.0

BIBLIOGRAPHY

Adams, J.J. and Bachu, S. (2002) 'Equations of state for basin geofluids: Algorithm review and intercomparison for brines', *Geofluids*, 2(4), pp. 257–271.

Akbar, S., Fathianpour, N. and Al Khoury, R. (2016) 'A finite element model for high enthalpy two-phase flow in geothermal wellbores', *Renewable Energy*, 94, pp. 223–236.

Aksoy, N. (2007) 'Optimization of downhole pump setting depths in liquid-dominated geothermal systems: A case study on the Balcova-Narlidere field, Turkey', *Geothermics*, 36(5), pp. 436–458.

Ambastha, A. K., & Gudmundsson, J. S. (1986a) *Collection and Evaluation of Flowing Pressure and Temperature Data fkom Geothermal Wells, (SGP-TR-100)*, Stanford Univ., Stanford, CA (USA), August 1986.

Ambastha, A. K., & Gudmundsson, J. S. (1986b) 'Pressure profiles in two-phase geothermal wells: comparison of field data and model calculations', *Proceedings 11th workshop on geothermal engineering,* Stanford, CA (USA), 21-23 January 1986.

Baldwin, J., Slatter, A., Jespersen, R. and Conaway, C.F. (2000) *Computer-assisted reservoir - Management*. United States: PennWell Books.

Barelli, A., Corsi, R., Del Pizzo, G. and Scali, C. (1982) 'A two-phase flow model for geothermal wells in the presence of non-condensable gas', *Geothermics*, 11(3), pp. 175–191.

Battistelli, A., Calore, C. and Pruess, K. (1997) 'The simulator TOUGH2/EWASG for modelling geothermal reservoirs with brines and non-condensible gas', *Geothermics*, 26(4), pp. 437–464.

Batzle, M. and Wang, Z. (1992) 'Seismic properties of pore fluids', *GEOPHYSICS*, 57(11), pp. 1396–1408.

Baumann, K. (1921) 'Some recent developments in large steam turbine practice', *Journal of the Institution of Electrical Engineers*, 59(302), pp. 565–623.

Bertani, R. (2016) 'Geothermal power generation in the world 2010–2014 update report', *Geothermics*, 60, pp. 31–43.

Bischoff, J.L. and Pitzer, K.S. (1989) 'Liquid-vapor relations for the system NaCl-H 2 O; summary of the p-t-x surface from 300 degrees to 500 degrees C', *American Journal of Science*, 289(3), pp. 217–248.

Bromley, C. and Bignall, G. (2016) *Ngawha geothermal field: geology, geophysics, conceptual model, geochemical monitoring trends and environmental issues.* JOGMEC-GNS Workshop, Tokyo, 2 June 2016. Available at: http://geothermal.jogmec.go.jp/ (Accessed: 9 December 2016).

Carroll, J.J., Slupsky, J.D. and Mather, A.E. (1991) 'The Solubility of carbon dioxide in water at low pressure', *Journal of Physical and Chemical Reference Data*, 20(6), p. 1201.

Chadha, P.K., Malin, M.R. and Palacio-Perez, A. (1993) 'Modelling of two-phase flow inside geothermal wells', *Applied Mathematical Modelling*, 17(5), pp. 236–245.

Champel, B. (2006) 'Discrepancies in brine density databases at geothermal conditions', *Geothermics*, 35(5-6), pp. 600–606.

Chen, J.C. (1966) 'Correlation for boiling heat transfer to saturated fluids in Convective flow', *Industrial & Engineering Chemistry Process Design and Development*, 5(3), pp. 322–329.

Chen, N.H. (1979) 'An explicit equation for friction factor in pipe', *Industrial & Engineering Chemistry Fundamentals*, 18(3), pp. 296–297.

Cholet, H. (2008) *Well production practical handbook*. Paris, France: Technip.

Chou, I.M. (1987) 'Phase relations in the system NaCl-KCl-H2O. III: Solubilities of halite in vapor-saturated liquids above 445°C and redetermination of phase equilibrium properties in the system NaCl-H2O to 1000°C and 1500 bars', *Geochimica et Cosmochimica Acta*, 51(7), pp. 1965–1975.

Clark, N. and Flemmer, R. (1986) 'The effect of varying gas voidage distributions on average holdup in vertical bubble flow', *International Journal of Multiphase Flow*, 12(2), pp. 299–302.

Clegg, J.D., Bucaram, S.M. and Hein, N.W. (1993) 'Recommendations and comparisons for selecting artificial-lift methods', *Journal of Petroleum Technology*, 45(12), pp. 1128–1167.

Clyde Pumps Ltd (2008) *Hydraulic submersible pump system* [Brochure]. Available at: http://www.clydepumps.com (Accessed: 26 May 2016).

Coddington, P. and Macian, R. (2002) 'A study of the performance of void fraction correlations used in the context of drift-flux two-phase flow models', *Nuclear Engineering and Design*, 215(3), pp. 199–216.

Coker, K.A. and Coker, P.K.A. (2010) *Ludwig's applied process design for chemical and petrochemical plants: Volume 2: Distillation, packed towers, petroleum fractionation, gas processing and dehydration*. 4th edn. Oxford, UK: Gulf Professional Publishing.

Conaway, C.F., Baldwin, J., Slatter, A. and Jespersen, R. (2000) *The petroleum industry: A nontechnical guide*. Tulsa, OK: PennWell Pub. Co.

Dake, L.P. (1978) *Fundamentals of reservoir engineering*. Amsterdam: Elsevier Scientific Pub. Co.

DiPippo, R. (2012) *Geothermal power plants: Principles, applications, case studies and environmental impact, Third edition*. 3rd edn. Amsterdam: Butterworth-Heinemann.

Dittman, G. L. (1977) *Calculation of brine properties*. Lawrence Livermore Laboratory. Available at: http://www.osti.gov/ (Accessed: 3 June 2016).

Dix, G. E. (1971) 'Vapor void fractions for forced convection with subcooled boiling at low flow rates', University of California, Berkeley.

Driesner, T. (2007) 'The system H2O–NaCl. Part II: Correlations for molar volume, enthalpy, and isobaric heat capacity from 0 to 1000°C, 1 to 5000bar, and 0 to 1 XNaCl', *Geochimica et Cosmochimica Acta*, 71(20), pp. 4902–4919.

Duan, Z. and Sun, R. (2003) 'An improved model calculating CO2 solubility in pure water and aqueous NaCl solutions from 273 to 533 K and from 0 to 2000 bar', *Chemical Geology*, 193(3-4), pp. 257–271.

Duan, Z., Hu, J., Li, D. and Mao, S. (2008) 'Densities of the CO 2 –H 2 O and CO 2 –H 2 O–NaCl systems up to 647 K and 100 MPa', *Energy & Fuels*, 22(3), pp. 1666–1674.

Duan, Z., Hu, J., Zhu, C. and Chou, I.-M. (2007) 'PVTx properties of the CO2–H2O and CO2–H2O–NaCl systems below 647 K: Assessment of experimental data and thermodynamic models', *Chemical Geology*, 238(3-4), pp. 249–267.

Duan, Z., Møller, N. and Weare, J.H. (1992) 'An equation of state for the CH4-CO2-H2O system: I. Pure systems from 0 to 1000°C and 0 to 8000 bar', *Geochimica et Cosmochimica Acta*, 56(7), pp. 2605–2617.

Duan, Z., Sun, R., Zhu, C. and Chou, I.-M. (2006) 'An improved model for the calculation of CO2 solubility in aqueous solutions containing Na+, K+, ca2+, mg2+, Cl−, and SO42−', *Marine Chemistry*, 98(2-4), pp. 131–139.

El-Dessouky, H., Ettouney, H., Alatiqi, I. and Al-Nuwaibit, G. (2002) 'Evaluation of steam jet ejectors', *Chemical Engineering and Processing: Process Intensification*, 41(6), pp. 551–561.

Eppelbaum, L., Kutasov, I. and Pilchin, A. (2014) *Applied geothermics*. Berlin, Heidelberg: Springer Berlin Heidelberg.

European Geothermal Energy Council (EGEC) (2012) *Strategic research priorities for geothermal electricity - Technology platform on geothermal electricity (TP-Geoelec)*. Available at: http://www.egec.org/ (Accessed: 25 May 2016).

Flowserve (2011) *Pumps for geothermal power generation* - Production well - Heat transfer fluid - Condensate extraction - Circulating water - Re-injection well [Brochure]. Available at: https://www.flowserve.com/ (Accessed: 25 May 2016).

Forster, H.K. and Zuber, N. (1955) 'Dynamics of vapor bubbles and boiling heat transfer', *AIChE Journal*, 1(4), pp. 531–535.

Francke, H. (2014) *Thermo-hydraulic model of the two-phase flow in the brine circuit of a geothermal power plant.* Berlin, Technische Universität Berlin, PhD thesis.

Francke, H. and Thorade, M. (2010) 'Density and viscosity of brine: An overview from a process engineers perspective', *Chemie der Erde - Geochemistry*, 70, pp. 23–32.

Francke, H., Kraume, M. and Saadat, A. (2013) 'Thermal–hydraulic measurements and modelling of the brine circuit in a geothermal well', *Environmental Earth Sciences*, 70(8), pp. 3481–3495.

Frick, S., Regenspurg, S., Kranz, S., Milsch, H., Saadat, A., Francke, H., Brandt, W. and Huenges, E. (2011) 'Geochemical and process engineering challenges for geothermal power generation', *Chemie Ingenieur Technik*, 83(12), pp. 2093–2104.

Frick, S., Saadat, A., Surana, T., Ezer Siahaan, E., Kupfermann, G.A., Erbas, K., Huenges, E. and Gani, M.A. (2015) 'Geothermal binary power plant for Lahendong, Indonesia: A German-Indonesian collaboration project', *Proceedings World Geothermal Congress 2015*, Melbourne, 19-25 April 2015.

Garcia-Gutierrez, A., Espinosa-Paredes, G. and Hernandez-Ramirez, I. (2002) 'Study on the flow production characteristics of deep geothermal wells', *Geothermics*, 31(2), pp. 141–167.

Gates, J.A., Tillett, D.M., White, D.E. and Wood, R.H. (1987) 'Apparent molar heat capacities of aqueous NaCl solutions from 0.05 to 3.0 mol·kg−1, 350 to 600 K, and 2 to 18 MPa', *The Journal of Chemical Thermodynamics*, 19(2), pp. 131–146.

Geremew, H. (2012) *A study of thermodynamic modelling and gas extraction system design for Aluto Langano geothermal power plant II in Ethiopia.* UNU-GTP, Iceland, Report 10, pp. 99-136.

Glassley, W. E. (2014) *Geothermal energy: renewable energy and the environment.* CRC Press.

Godbole, P.V., Tang, C.C. and Ghajar, A.J. (2011) 'Comparison of void fraction correlations for different flow patterns in upward vertical Two-Phase flow', *Heat Transfer Engineering*, 32(10), pp. 843–860.

Gokcen, G. and Yildirim, N. (2008) 'Effect of Non-Condensable gases on geothermal power plant performance. Case study: Kizildere geothermal power plant-turkey', *International Journal of Exergy*, 5(5/6), p. 684.

Gomberg, S. (2016) *Benefits of renewable energy use*. Available at: http://www.ucsusa.org/ (Accessed: 9 May 2016).

Gnielinski, V. (2009) 'Heat transfer coefficients for turbulent flow in concentric Annular ducts', *Heat Transfer Engineering*, 30(6), pp. 431–436.

Gnielinski, V. (1976) 'New equations for heat and mass-transfer in turbulent pipe and channel flow', *International chemical engineering*, 16(2), pp. 359-368.

Grant, M.A. and Bixley, P.F. (2011) *Geothermal reservoir engineering*. 2nd edn. United States: Elsevier Science Publishing Co.

Guet, S. C. L. (2004) *Bubble size effect on the gas-lift technique*. Delft, Delft University of Technology, PhD thesis.

Harmathy, T.Z. (1960) 'Velocity of large drops and bubbles in media of infinite or restricted extent',*AIChE Journal*, 6(2), pp. 281–288.

Harrison, R., Mortimer, N. and Smarason, O. (1990) *Geothermal heating: Handbook of engineering economics*. 1st edition edn. Exeter: Pergamon Press.

Harvey, A.H. and Prausnitz, J.M. (1989) 'Thermodynamics of high-pressure aqueous systems containing gases and salts', *AIChE Journal*, 35(4), pp. 635–644.

Hasan, A.R., Kabir, C.S. and Sayarpour, M. (2010) 'Simplified two-phase flow modeling in wellbores', *Journal of Petroleum Science and Engineering*, 72(1-2), pp. 42–49.

Hecker, C. (2016) *GEOCAP project - what is GEOCAP?* Available at: https://www.geocap.nl/ (Accessed: 10 May 2016).

Heineken (2016) *Heineken/BrineProp*. Available at: https://github.com/Heineken/BrineProp/releases (Accessed: 20 October 2016).

Hibiki, T. and Ishii, M. (2003) 'One-dimensional drift–flux model for two-phase flow in a large diameter pipe', *International Journal of Heat and Mass Transfer*, 46(10), pp. 1773–1790.

Hills, J.H. (1976) 'The operation of a bubble column at high throughputs', *The Chemical Engineering Journal*, 12(2), pp. 89–99.

Hosgor, F. B., Tureyen, O. I., Satman, A., & Cinar, M. (2015) 'Effects of Carbon Dioxide Dissolved in Geothermal Water on Reservoir Production Performance', *Proceedings World Geothermal Congress 2015,* Melbourne, Australia.

Huenges, E. and Ledru, P. (eds.) (2010) *Geothermal energy systems: Exploration, development, and utilization*. Germany: Wiley-VCH Verlag GmbH.

IF Technology (2012) *Electricity production from deep geothermal wells in the Netherlands*. IF Technology, Arnhem.

IPS (1998) IPS-E-PR-745 Engineering standard for process design of vacuum equipment (vacuum pumps and steam jet ejectors).

Ishii, M. (1977) *One-dimensional drift-flux model and constitutive equations for relative motion between phases in various two-phase flow regimes (No. ANL-77-47).* Argonne National Lab., Ill.(USA).

Kataoka, I. and Ishii, M. (1987) 'Drift flux model for large diameter pipe and new correlation for pool void fraction', *International Journal of Heat and Mass Transfer*, 30(9), pp. 1927–1939.

Kelessidis, V.C., Karydakis, G.I. and Andritsos, N. (2007) 'Method for selecting casing diameters in wells producing low-enthalpy geothermal waters containing dissolved carbon dioxide', *Geothermics*, 36(3), pp. 243–264.

Kestin, J., Khalifa, H.E. and Correia, R.J. (1981) 'Tables of the dynamic and kinematic viscosity of aqueous KCl solutions in the temperature range 25–150 °C and the pressure range 0.1–35 MPa', *Journal of Physical and Chemical Reference Data*, 10(1), p. 57.

Khalifa, H. E. and Michaelides, E. (1978) *Effect of non-condensable gases on the performance of geothermal steam power systems, (No. COO-4051-36).* Brown Univ., Providence, RI (USA), Dept. of Engineering.

Li, Y.-K. and Nghiem, L.X. (1986) 'Phase equilibria of oil, gas and water/brine mixtures from a cubic equation of state and henry's law', *The Canadian Journal of Chemical Engineering*, 64(3), pp. 486–496.

Lienau, P.J., Lunis, B.C. and Lund, J.W. (1991) *Geothermal direct use engineering and design guidebook.* United States: Oregon Institute of Technology, Geo-Heat Center.

Mahon, T., Harvey, C., & Crosby, D. (2000) 'The chemistry of geothermal fluids in Indonesia and their relationship to water and vapour dominated systems', *Proceedings of the World Geothermal Congress 2000*, Kyushu–Tohoku, Japan, pp. 1389-1394.

Mills, A.F. (1998) *Basic heat and mass transfer*. 2nd edn. New York, NY, United States: Prentice Hall.

Moghaddam, A. R. (2006) 'A conceptual design of a geothermal combined cycle and comparison with a single-flash power plant for well NWS-4, Sabalan, Iran'. Report 18 in: *Geothermal Training in Iceland 2006*, UNU-GTP, Iceland, pp. 391-428.

Nasruddin, Idrus Alhamid, M., Daud, Y., Surachman, A., Sugiyono, A., Aditya, H.B. and Mahlia, T.M.I. (2016) 'Potential of geothermal energy for electricity generation in Indonesia: A review', *Renewable and Sustainable Energy Reviews*, 53, pp. 733–740.

New Mexico Tech (2005) *Introduction to artificial lift*. Available at: http://www.nmt.edu/ (Accessed: 27 May 2016).

Nicklin, D. J. (1961) *Two phase flow in vertical tubes*. PhD Thesis.

Ocampo-Díaz, J. D. D., Valdez-Salaz, B., Shorr, M., Sauceda, M., & Rosas-González, N. (2005) 'Review of corrosion and scaling problems in Cerro Prieto geothermal field over 31 years of commercial operations', *Proceedings of World Geothermal Congress,* Antalya, Turkey, 24 – 29 April 2005.

Olasolo, P., Juárez, M. C., Morales, M. P., & Liarte, I. A. (2016) 'Enhanced geothermal systems (EGS): A review', *Renewable and Sustainable Energy Reviews*, *56*, pp. 133-144.

Palliser, C. and McKibbin, R. (1998a) 'A model for deep geothermal brines, I: TpX state-space description', *Transport in porous media*, *33*(1-2), pp. 65-80.

Palliser, C. and McKibbin, R. (1998b) 'A model for deep geothermal brines, III: Thermodynamic properties–enthalpy and viscosity', *Transport in Porous Media*, *33*(1-2), pp. 155-171.

Parada, A. F. M. (2013) 'Geothermal binary cycle power plant principles, operation and'. Report 20 in: *Geothermal Training in Iceland 2013*, UNU-GTP, Iceland, pp. 443-476.

Phillips, S. L. (1981) *A technical databook for geothermal energy utilization.* Lawrence Berkeley National Laboratory. Available at: http://escholarship.org/uc/item/5wg167jq Accessed: 3 June 2016).

Pitzer, K.S. (1973) 'Thermodynamics of electrolytes. I. Theoretical basis and general equations', *The Journal of Physical Chemistry*, 77(2), pp. 268–277.

Pitzer, K.S., Peiper, J.C. and Busey, R.H. (1984) 'Thermodynamic properties of aqueous sodium chloride solutions', *Journal of Physical and Chemical Reference Data*, 13(1), p. 1.

Potter, R. W., Babcock, R. S., & Brown, D. L. (1977) 'New method for determining the solubility of salts in aqueous solutions at elevated temperatures', *Journal of Research of the U.S. Geological Survey*, *5*(3).

Pruess, K. (2002). *Mathematical modeling of fluid flow and heat transfer in geothermal systems: an introduction in five lectures.* Orkustofnun.

Renpu, W. (2011) *Advanced well completion engineering*. 3rd edn. United States: Gulf Professional Publishing.

Rivera Diaz, A., Kaya, E. and Zarrouk, S.J. (2016) 'Reinjection in geothermal fields − A worldwide review update', *Renewable and Sustainable Energy Reviews*, 53, pp. 105–162.

Rouhani, S.Z. and Axelsson, E. (1970) 'Calculation of void volume fraction in the subcooled and quality boiling regions', *International Journal of Heat and Mass Transfer*, 13(2), pp. 383–393.

Ryley, D.J. (1980) 'The mass discharge of a geofluid from a geothermal reservoir — well system with flashing flow in the bore', *Geothermics*, 9(3-4), pp. 221–235.

SANDIA (2008) *Enhanced geothermal systems (EGS) well construction technology evaluation report.* Available at: https://www1.eere.energy.gov/ (Accessed: 27 May 2016).

Schlegel, J., Hibiki, T. and Ishii, M. (2010) 'Development of a comprehensive set of drift-flux constitutive models for pipes of various hydraulic diameters', *Progress in Nuclear Energy*, 52(7), pp. 666–677.

Shen, X., Matsui, R., Mishima, K. and Nakamura, H. (2010) 'Distribution parameter and drift velocity for two-phase flow in a large diameter pipe', *Nuclear Engineering and Design*, 240(12), pp. 3991–4000.

Sheppard, D. S. (1987) 'Geochemistry and the Exploration of the Ngawha Geothermal System, New Zealand', *Proceedings 12th workshop on geothermal engineering,* Stanford, CA (USA), 20-22 January 1987.

Shi, H., Holmes, J.A., Durlofsky, L.J., Aziz, K., Diaz, L., Alkaya, B. and Oddie, G. (2005) 'Drift-flux modeling of Two-Phase flow in Wellbores', *SPE Journal*, 10(01), pp. 24–33.

Sinnott, R.K. and Towler, G. (2009) *Chemical engineering design: SI edition*. 5th edn. Oxford, United Kingdom: Butterworth-Heinemann.

Span, R. and Wagner, W. (1996) 'A new equation of state for carbon dioxide covering the fluid region from the triple-point temperature to 1100 K at pressures up to 800 MPa', *Journal of Physical and Chemical Reference Data*, 25(6), pp. 1509–1596.

Swamee, P.K., Jain, A.K. (1976) 'Explicit equations for pipe-flow problems', *Journal of the Hydraulics Division,* 102(5), pp. 657–664.

Thome, J.R. (2010) *Engineering data book 3*. Elvetia: Wolverine Tube Inc.

Taitel, Y., Bornea, D. and Dukler, A.E. (1980) 'Modelling flow pattern transitions for steady upward gas-liquid flow in vertical tubes', *AIChE Journal*, 26(3), pp. 345–354.

Van der Hoorn, K., Rombaut, B., Maaijwee, C., Gankema, M. and Smeets, J. (2012) *Geothermal energy at Hoogeveen - Feasibility study*. Available at: http://soilpedia.nl/ (Accessed: 10 May 2016).

Van Everdingen, A.F. (1953) 'The skin effect and its influence on the productive capacity of a well', *Journal of Petroleum Technology*, 5(6), pp. 171–176.

VDI (2010) *VDI Heat Atlas*. Edited by VDI-Gesellschaft Verfahrenstechnik und Chemieingenieurwesen. 2nd edn. Germany: Springer-Verlag Berlin and Heidelberg GmbH & Co. K.

Wagner, W. and Pruß, A. (2002) 'The IAPWS formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use', *Journal of Physical and Chemical Reference Data*, *31*(2), pp. 387-535.

Wang, X., Liu, X. and Zhang, C. (2013) 'Performance analysis of organic Rankine cycle with preliminary design of radial turbo Expander for binary-cycle geothermal plants', *Journal of Engineering for Gas Turbines and Power*, 135(11), p. 111402.

Wallis, G.B. (1969) *One-Dimensional Two-Phase flow: The First complete account of John Paul Jones' greatest battle*. United States: McGraw Hill Higher Education.

Wisman, R. (1975) 'Analytical pressure drop correlation for adiabatic vertical two-phase flow', *Applied Scientific Research*, 30(5), pp. 367–380.

Woldesemayat, M.A. and Ghajar, A.J. (2007) 'Comparison of void fraction correlations for different flow patterns in horizontal and upward inclined pipes', *International Journal of Multiphase Flow*, 33(4), pp. 347–370.

Xie, X., Bloomfield, K.K., Mines, G.L. and Shook, G.M. (2005) *Design considerations for artificial lifting of enhanced geothermal system fluids.* Report No. INL/EXT-05-00533, Idaho, USA: Idaho National Laboratory.

Yuniarto, Soesilo, T. E. B. and Heviati, E. (2015) 'Geothermal Power Plant Emissions in Indonesia', *Proceedings World Geothermal Congress 2015*, Melbourne, Australia.

Yusufova, V.D., Pepinov, R.I., Nikolaev, V.A. and Guseinov, G.M. (1975) 'Thermal conductivity of aqueous solutions of NaCl', *Journal of Engineering Physics*, 29(4), pp. 1225–1229.

Zarrouk, S.J. and Moon, H. (2014) 'Efficiency of geothermal power plants: A worldwide review',*Geothermics*, 51, pp. 142–153.

Zhenhao Duan Research Group (2006) *The Duan group - models - H2O-CO2-NaCl*. Available at: http://models.kl-edi.ac.cn/models/h2o_co2_nacl/ (Accessed: 20 October 2016).

Zuber, N. and Findlay, J.A. (1965) 'Average volumetric concentration in two-phase flow systems', *Journal of Heat Transfer*, 87(4), p. 453.

Zuo, Y.-X. and Guo, T.-M. (1991) 'Extension of the Patel—Teja equation of state to the prediction of the solubility of natural gas in formation water', *Chemical Engineering Science*, 46(12), pp. 3251–3258.

NOMENCLATURE

List of Symbols

Roman symbol	Description	Unit
a	diameter ratio	-
A	area	m^2
c	isothermal compressibility	Pa^{-1}
c_p	isobaric heat capacity	$J\ kg^{-1}\ K^{-1}$
C	circumference well interior	m
C_0	flow distribution parameter	-
C_f	skin friction coefficient	-
D	diameter	m
e	specific exergy	$J\ kg^{-1}$
er	error	-
$er\%$	percentage error	-
$\dot{E}$	maximum theoretical power	W
E	energy	$J\ kg^{-1}$
f	Darcy friction factor	-
F	friction force	N
F_a	correlation factor for annular ducts in Gnielinski (2009)	-
F_θ	well-deviation factor	-
g	gravitational acceleration (9.81)	$m\ s^{-2}$
G	mass flux	$kg\ m^{-2}\ s^{-1}$
h	specific enthalpy	$J\ kg^{-1}$
h_c	convective heat transfer coefficient	$W\ m^{-2}\ K^{-1}$
II	injectivity index	$kg\ s^{-1}\ Pa^{-1}$
J	mass flux	$kg\ m^{-2}\ s^{-1}$
k	thermal conductivity	$W\ m^{-1}\ K^{-1}$
k_1	correlation constant for annular ducts in Gnielinski (2009)	-
K	permeability	m^2
L	length	m
$\dot{m}$	mass flow rate	$kg\ s^{-1}$
m	molality	$mol\ kg^{-1}$
M	molar mass	$kg\ mol^{-1}$
P	pressure	MPa
P_i	partial pressure of gas i	Pa
PI	productivity index	$kg\ s^{-1}\ Pa^{-1}$
Pr	Prandtl number	-
q	volumetric flow rate	$m^3\ s^{-1}$
$\dot{Q}$	heat flow rate	W
r	radius	m
Re	Reynolds number	-
s	specific entropy	$J\ kg^{-1}\ K^{-1}$
S	skin factor	-
t	time	s
T	temperature	^oC
T_h^*	scaled temperature for enthalpy correlation	^oC
tvd	true vertical depth	m
u	velocity	$m\ s^{-1}$
U	overall heat transfer coefficient	$W\ m^{-2}\ K^{-1}$
v	specific volume	$m^3\ kg^{-1}$
V	volume	m^3
w	mass fraction	$kg\ kg^{-1}$, wt%

$\dot{W}$	rate of work, power	W
x	mole fraction	mol mol^{-1}
X_{tt}	Lockhart-Martinelli parameter	-
z	elevation	m

Greek symbol	Description	Unit
α	thermal diffusivity	m^2 s^{-1}
γ	Euler's constant (1.78)	-
ε	absolute pipe roughness	m
ε_g	cross-sectional void fraction	m^2 m^{-2}
η	efficiency	-
θ	inclination angle well	-
μ	dynamic viscosity	Pa s
ρ	density	kg m^{-3}
σ	surface tension	kg m^{-2}
τ	shear stress	N m^{-2}
Φ	porosity	m^3 m^{-3}
χ	quality, gas mass fraction	kg kg^{-1}

Subscript	Description
∞	bubble-rise
∞b	small bubble-rise
∞T	Taylor bubble-rise
0	dead-state, ambient
$0a$	fully developed annular flow
$0b$	fully developed bubble flow
$0c$	fully developed churn flow
$0s$	fully developed slug flow
a	annular
ao	annulus outer
b	brine
BC	binary cycle
bh	bottom hole
c	critical
cd	condenser
$comp$	compressor
cp	condenser pump
CS	cross-sectional
cw	cooling water
cwp	cooling water pump
e	electrical
ev	evaporator
E	entrance
f	frictional
fc	forced-convective
FP	flash point
g	gas, geothermal, generator
gb	superficial gas (transition from bubble to slug flow)
gc	superficial gas (transition from churn to annular flow)
gf	geothermal fluid
GL	gas lift
gu	drift-flux (gas velocity relative to mixture velocity)
h	hydraulic
hp	high-pressure
hs	hydrostatic
i	inner, component, segment number

in	input
inj	injection
ip	injection pump
k	kinetic
l	liquid
lp	low-pressure
m	mixture
mc	slug to churn transition
mix	mixture
mp	make-up pump
ms	minimum mixture
nb	nucleate boiling
net	net
o	outer
OPT	optimized
p	pump
$pinch$	pinchpoint
pot	potential
r	rock
R	reservoir
res	reservoir
s	isentropic
sat	two-phase saturated
SAT	three-phase saturated
SE/C	steam ejector/condenser
SF	single-flash
sg	superficial gas
$skin$	skin at the well-face
sl	superficial liquid
sol	solution
$start-up$	start-up
$surf$	surface
t	turbine
td	turbine dry
th	thermal
tw	turbine wet
u	utilization
v	vapor
w	water
W	well
Wc	well casing
Wi	well inner
Wo	well outer
$wall$	wall
wf	working fluid
wh	wellhead

List of Abbreviations

ASR	air to steam ratio
BC	binary cycle
BCV	ball check valve
BT	binary turbine
C	condenser
$CaCO_3$	calcium carbonate/calcite
CO_2	carbon dioxide
COM	component object model

COMP	compressor
CP	condensate pump
CR	compression ratio
CS	cyclone separator
CSV	control and stop valves
CT	cooling tower
CWP	cooling water pump
DAE	dry air equivalent
DFM	drift-flux model
E	evaporator
EGS	enhanced geothermal systems
EOS	equation of state
ESP	electrical submersible pump
ER	expansion ratio
F	flasher
FF	final filter
FP	flow pattern
G	generator
GEOCAP	Geothermal Capacity Building Programme – Indonesia-Netherlands
GFP	Visual Basic Excel model developed by Francke et al. (2013)
GFP Excel Model	geothermal fluid property model developed in MS Excel (Heineken, 2016)
GFP MATLAB Model	geothermal fluid property model developed in MATLAB for this study
GHG	greenhouse gas
GL	gas lift
GLV	gas lift valve
hp	high-pressure
H_2O	water
HEI	Heat Exchange Institute
HTP	hydraulic turbine pump
I/O	input/output
IP	injection pump
IW	injection well
lp	low-pressure
LRVP	liquid ring vacuum pumps
LSP	line shaft pump
M	make up water
MR	moisture remover
NaCl	sodium chloride
NCG	non-condensable gases
ORC	organic Rankine cycle
P	pump
PCP	progressing cavity pump
PH	pre-heater
PR	particulate remover
PW	production well
S	silencer
SC	steam consumption
SE/C	steam ejector/condenser
SF	single-flash
SiO_2	silica
SP	steam piping
SR	sand remover
SRP	sucker rod pump
ST	steam turbine
T/G	turbine/generator
TCF	temperature correction factor
TDS	total dissolved solids
TV	throttling valve
VBA	Visual Basic for Applications
VLE	vapor-liquid equilibrium

WER	weight entrainment ratio
wf	working fluid
WP	water (brine) piping
WV	wellhead valve