NTNU
Det skapende universitet

TKP4580 1 Kjemisk prosessteknologi, fordypningsprosjekt

Kandidat 10018

Oppgaver	Oppgavetype	Vurdering	Status
1 Handing in the specialization report.	Filopplasting	Manuell poengsum	Levert

TKP4580 1 Kjemisk prosessteknologi, fordypningsprosjekt

Emnekode	TKP4580	PDF opprettet	19.12.2016 10:27
Vurderingsform	TKP4580	Opprettet av	Hege Johannessen
Starttidspunkt:	30.11.2016 10:00	Antall sider	54
Sluttidspunkt:	16.12.2016 15:45	Oppgaver inkludert	Ja
Sensurfrist	Ikke satt	Skriv ut automatisk rettede	Ja

Seksjon 1

1 OPPGAVE

Handing in the specialization report.

Here you can upload your project report.
Only .pdf files are accepted.

Mark the file name with name_research group_supervisor!

Click here to upload your file

BESVARELSE

Filopplasting

Filnavn	10163874_cand-12017813_9125253
Filtype	pdf
Filstørrelse	1881.898 KB
Opplastingstid	15.12.2016 22:27:08

Neste side
Besvarelse vedlagt

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for naturvitenskap og teknologi
Institutt for kjemisk prosessteknologi

SPECIALIZATION PROJECT 2016

TKP4580

PROJECT TITLE:

Implementation and simulation of a subsea gravity separator model in Modelica

By

Sarry Haj Yahia

Supervisor for the project: **Prof. Sigurd Skogestad**
Dr. Christoph J. Backi

Date: 16/12/2016

Abstract

In this work we study the concept of subsea separation with emphasis on the gravity separator, we later introduce a mathematical representation of the gravity separator model. The model includes three dynamic state equations describing the levels of water and overall liquid as well as the gas pressures subject to the in- and outflow dynamics. In addition, the model includes algebraic equations to calculate the droplet distributions for each continuous phase in order to investigate the separation between phases. The inflow to the system acts as a disturbance, and PI controllers are added to control the outflows of oil, water and gas. Furthermore, Modelica was studied to introduce the gravity separator model and it was used as the primary simulation tool to obtain results of the model. The simulation results were almost identical for the ones obtained in Matlab, and show very interesting behaviors. The Modelica model are built in a generic way for the purpose of rescaling them for larger and more complex systems. Finally, the Subpro library gravity separator requires further work in order to be able to compare both models.

Preface

This project was written as a part of my M.Sc degree in Chemical Engineering at the Norwegian University of Science and Technology.

I would like to thank my co-supervisor, Dr. Christoph J. Backi, for his endless patience and continuous support and efforts in assisting me to achieve the goals of this project throughout the semester. I also want to thank my supervisor, Prof. Sigurd Skogestad for his valuable input during the work in this project.

Declaration of Compliance

I hereby declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology (NTNU).

Trondheim, December 16th, 2016

Sarry Haj Yahia.

Contents

List of Symbols & Simulation Parameters [11,16]

L	Length of active separation zone	10 m
M_G	Molar mass of the gas	0.01604 kg mol^{-1}
N_S	Number of segments	5
r	Radius of separator	1.65 m
R	Universal gas constant	8.314 kg m^2 (s^2 mol K)$^{-1}$
T	Temperature	328.5 K
V_{Sep}	Volume of active separation zone	85.53 m^3
A_O	Cross sectional area of oil	-
A_W	Cross sectional area of water	-
h_O	Level of oil	-
h_W	Level of water	-
g	Gravitational acceleration	9.8 m s^{-2}
v_h	Horizontal velocity	-
v_v	Vertical velocity	-
$q_{G,in}$	Volumetric inflow of gas	0.456 m^3 s^{-1}
$q_{L,in}$	Volumetric inflow of liquid	0.59, 0.73 m^3 s^{-1}
α	Water cut of the liquid inflow	0.135, 0.475
β	Oil cut of the oil inflow	0.865, 0.525
μ_O	Dynamic viscosity of oil	0.001 kg/ s·m
μ_W	Dynamic viscosity of water	0.0005 kg/ s·m
ρ_O	Density of oil	831.5 kg m^{-3}
ρ_W	Density of water	1030 kg m^{-3}
ρ_G	Density of gas	49.7 kg m^{-3}
ϕ_{WO}	Initial separation factor for water	0.3, 0.4, 0.5
ϕ_{OW}	Initial separation factor for oil	0.3, 0.4, 0.5

1.Introduction

Subsea technology in offshore oil and gas production is a highly specialized field of application with particular demand on engineering and simulation. Here is a brief introduction to subsea oil production, subsea separation system, and simulation tools.

1.1. Subsea Oil Production

Due to huge potential, subsea has been given increasingly more and more attention by the oil industry for the last decades. Successful methods of subsea processes have led to extensive technology development work on subsea separation. Therefore, subsea process developments are becoming important in this field [1]. Subsea operating costs are reduced by avoiding the use of energy to pump the produced water to the surface, and rather pump it back down into the reservoir or release it into the sea. The full scale subsea separation facility increases oil recovery from fields allowing access to additional millions of barrels [2]. When Oil is produced, it unintentionally produces a huge quantity of water. During a lifetime of an oil production facility the water production will increase with decreasing pressure of the reservoir. This creates a problem for platforms that are receiving the flow, since it leads to pressure drop in the pipelines from the well to the platforms. Subsea technology finds a solution by either reinjecting the water back into the reservoir or by releasing the water to the sea [3], [4]. Subsea technology also reduces the requirement for building large platforms.

Having introduced some of the benefits of subsea process, there exist several challenges in this field. The operation of the process plant must be simple, safe and effective. It must be possible to retrieve and replace or repair critical modules quickly and effectively. Maintenance of production is ought to occur independently of the processing plant. There also exist some challenges in power transmission, monitoring of sand and water production, and finally, multi-phase metering in the production pipelines [2].

1.2. Subsea Separation

To be able to achieve or build a good separation system, it is essential to understand what is being separated, the properties of the components, and phenomena that occur between phases and in the continuous phase. Therefore, crude oil properties and emulsions are discussed in this section.

1.2.1. Crude oil properties

Although the separation of two liquids, appears to be a simple process, it is crucial to understand the properties and characteristics of the components that are desired to be separated, in order to achieve high efficiencies. In this process, water is separated from crude oil. Crude oil consists of huge amounts of hydrocarbon molecule groups which could possibly affect the separation process. Oil's components are divided according to their polarizability and polarity in an analysis method called SARA [5]. The term SARA stands for:

- *Saturates*: non-polar hydrocarbons without double bonds.
- *Aromatics*: benzene rings and derivatives.
- *Resins*: polar group containing heteroatoms like Oxygen, Nitrogen and sulfur.

- *Asphaltenes*: molecules containing polar, acidic, and basic groups. In addition, they are polycyclic and tend to form stacked aggregates.

1.2.2. Emulsions

When one or more liquids are dispersed in another immiscible liquid, an emulsion is created. Emulsions contain both a dispersed (droplet form) and a continuous (where droplets are present) phase, with the boundary between the phases called interface. In the case of oil water separation an emulsion could be oil in water (left picture in figure 1), water in oil (center picture in figure 1), as well as water in oil in water (right picture in figure 1), and vice versa. Therefore, it is really important to understand this phenomena, given that when separation of phases occurs, not all water will directly enter its respective continuous phase, rather a fraction of it will disperse in the oil phase. The same holds for incoming oil, a fraction of the oil will be dispersed in the bulk water phase. Emulsions are stabilized by natural emulsifiers or by adding emulsifying agents. Emulsifying agents have both hydrophilic part that have strong affinity to water particles, and hydrophobic parts that have strong affinity to oil. These agents are concentrated at the oil/water interface to provide a protective barrier around the dispersed droplets. In addition to this barrier, emulsifiers stabilize the emulsion by reducing the interfacial tension of the system. Emulsifying agents can be classified according to their chemical structure, or mechanism of action [6]. They also include surface active agents which are known as surfactants. Surfactants are compounds that lower the interfacial tension between two liquids or a liquid and a solid.

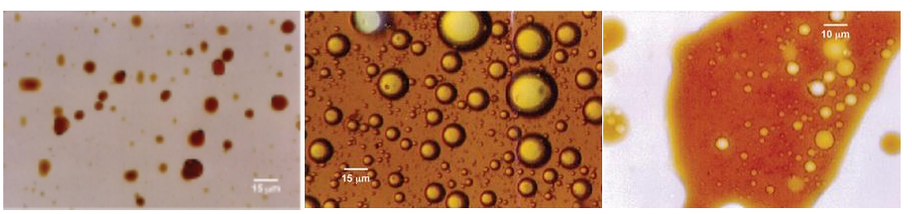

Figure 1: Starting from the left, the first picture displays an oil in water emulsion. Second picture, shows water in oil emulsion. Finally, the third picture indicates that emulsions could occur several times, in this case, it is a water in oil in water emulsion [4].

1.2.3. Methods and phenomena for increasing separation efficiency

Foam inhibitors show promising results in this field. The addition of foam inhibitors could increase the separation process efficiency, since it decreases the foam created when liquid and gas are mixed as they enter the separator. As foam is formed on the interface between oil and gas, it increases difficulties in liquid level measurements since the sensors might transmit wrong measurements, which can result in faulty control.

Emulsion inhibitors (Demulsifiers/ Emulsion breakers), which are commonly used in crude oil processing, are used to separate emulsions of oil in water or water in oil. Demulsifiers work to destroy the interfacial film that raises the stability of an emulsion. Once this film is destroyed it is easier for droplets to aggregate and coalesce. The first step in demulsification is the flocculation of droplets also called aggregation. The droplets clump together to form aggregates. If the emulsifier film is week the droplets will proceed to coalesce. Coalescence is the process by which two or more droplets, bubbles or particles merge during contact to form a single daughter droplet, bubble or particle. It is an irreversible process that leads to a decrease in the number of water droplets [13]. There are several factors that affect demulsification,

flocculation, and coalescence. If the amount of dispersed droplets in a continuous phase is high, it will increase the flocculation rate. Increase in temperature also leads to high flocculation since the thermal energy of droplets is increased which leads to higher collisions. Higher rate of flocculation increases the collision frequency, thereby increases the probability for coalescence. In addition, higher temperature reduces viscosity which increases again the collision frequency.

1.2.4. Subsea Separators

Separators are used to separate oil or gas from water or other undesired substances, such that the water can be reinjected into the reservoir in order to enhance oil recovery and boost production. This work is focused in particular on 3 phase (gas-oil-water) subsea gravity separators. It is beneficial to separate these phases early in the processing at least in subsea applications to obtain as pure single phase streams as possible. This is achieved by implementing and utilizing several separation techniques. Commonly, separation of streams begins with bulk separation of liquid and gas using e.g. cyclones, followed by the 3 phase separation using a gravity separator. After separating the streams, further purification of oil water and gas can be achieved using hydrocyclones and membrane technologies for separation.

Cyclone

Cyclone separators are used to separate dispersed phase from a continuous phase based on centrifugal forces. There are several types of cyclones namely Conical Liquid Hydrocyclones, Liquid-Liquid Cylindrical Cyclones, and Gas Liquid Cylindrical Cyclones [4]. The differences between these mainly lies in the design of the unit. Cyclone separators have one inlet and two outlets (one at the top and one at the bottom). As the mixture flows in through the inlet of the cyclone it creates a large swirling flow that forces lighter components to be allocated to the center while heavier and larger components migrate to the walls of the cyclone. In the case of oil water separation oil will migrate to the center and will flow out through the overflow, while water will go out through the underflow [4].

Swirl

Swirl element is a new generation of compact separators which can be inserted into well stream processing pipework on production platforms. It is a better and cheaper system for a primary liquid-liquid separation. A swirl separation unit typically consists of the swirl element, the cylindrical pipe, and an outflow section. The basic principle involves the use of static swirl elements, which induce cyclonic flows to press liquids (water & oil)/water out towards the pipe walls while driving the gas/oil inwards to create a gas/oil core. The two factors that contribute to the separation are the differences in density between oil and water, and the centrifugal force [9].

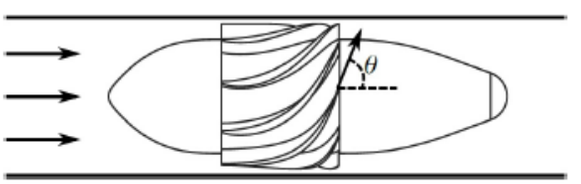

Figure 2: A swirl element used for oil and water separation inside a pipe. The multi/two-phase flow enters from the left. Phases will then separate due to centrifugal and density differences [4].

An important point to mention is that although these separators are easy to install and lead towards lower investment cost, this separator is still not in use. This compact in line separator is yet to be used in subsea mainly because of the lack in experience and research in the field.

Gravity Separator

The idea behind a gravity separator, is to allow particles in the tank to settle, sediment, or cream, given their density differences. In a liquid-liquid separation process, particles dispersed in the continuous phase will have two options. If their density exceeds that of the continuous phase they will sediment and go to the bottom of the tank. However, if their relative density is lower than that for the continuous phase, the particles will rise to the top. The most common liquid-liquid separators are horizontal. Usually a multiphase flow consisting of oil, water, and gas enters the separator, where the phases will initially separate gas at the top, water on the bottom, and oil in between. At the end of the separator there is a weir that serves as a wall for creating two outflows, where oil will flow over the weir to exist through an oil outlet. In the subsea industry it is necessary to have separators that are able to withstand extreme environments and high pressures. The separator is ought to be compact to be able to install, maintain, and retrieve it in a simple manner. In addition, it will reduce size, weight and cost.

Figure 3: Tordis gravity separator , on the left graphical representation of Tordis on sea bed. On the right, actual picture of Tordis gravity separator taken in June 2007. The separator ensures that residual oil droplets are discharged into separate subsea wells instead of the sea.

Since 2001 there were several subsea separator installations under operators such as Statoil, Petrobras, Shell, and Total [8]. Here are some Statoil subsea installations.

- Statoil Troll C: with its 115 subsea wells Troll is the world's largest subsea development. Troll pilot separator started production in 2001. It was the first subsea separation system to be installed on the sea bed at a depth of 340 meters and 3.5 kilometers from the platform. Water is separated from oil and gas using a gravity separator system. The water is then injected back into the reservoir, while the separated oil and gas are sent up to the platform. It is noteworthy to mention that the Troll pilot separator had an extra feature which reduced emulsions and increased initial oil-water-gas separation, that is it had a cyclonic inlet device that slows down the incoming flow and also decreases the amount of emulsions. This inlet feature leads to increased separation of gas, which is and then transported through a separate gas line [4,8,10].

- Statoil Tordis: this oil field lies in the Tampen area of the Norwegian North Sea, and came on stream in 1994. However, in 2007 Statoil, with the help of the FMC/CDS technology suppliers, have installed a 3 phase horizontal separator with a cyclonic inlet. This separator

has the capacity to handle water cuts of 30%-40%. It is 17 meters long, with a diameter of 2.11 meters, and liquid retention time of 3 minutes. In this field, sand was present in large amounts. Therefore, it was required to build a sand removal system after the separator. It is noteworthy to mention that all gas was separated initially through the cyclonic inlet, in addition to that, this separator did not include any weir to separate outflows, rather it disposed oil from the top and water with sand from the bottom.

1.3. Literature Review

This chapter reviews the existing methodologies in the research area of 3 phase gravity separation, dynamics, control and simulations. Previous work in this area lacks a dynamic, control-oriented approach in gravity separator modeling. [14] focuses on the three phase separator, where each phase's dynamics are modeled. This paper, extends a model based on the American Petroleum Institute (API) design criteria, to address the process dynamics in addition to its statics. The simulation results in [14] proved the sophistication of the model in spite of its simplicity. However, [14] demonstrated challenging task of modeling and controlling oil and gas facilities, and that more work needs to be done. [6] develops a CFD model for three-phase separators using a two fluid model extended to tree phases. [6] concluded that there is a strong dependency of drop diameter on entrainment. This is an expected observation, since increasing droplet's size will increase the relative velocity of the droplet. The model in [11] contributes towards developing a more advanced, yet simple and control-oriented model for subsea applications. The idea is to develop a model that delivers good and reliable results in the matter of product purities.

2. Gravity separator Modeling

This model is based primarily on the model proposed by Backi and Skogestad in their paper *A simple dynamic gravity separator model for separation efficiency evaluation incorporating level and pressure control* [11]. The well stream from a petroleum reservoir is a multi-phase flow with hydrocarbons, water and gas. As mentioned before it is very important and beneficial to separate these phases early in the processing stages in order to decrease impurities of the streams as much as possible. This model concentrates on a gravity separator. The driving forces of the separation are mainly differences in densities of the media and gravitational forces. The purpose of this model is to control the levels of liquid and water, as well as the gas pressure. Finally, to calculate the amount of oil in water and water in oil as the flows exit the separator unit. In this model the principle of *Stokes' law* is used to describe the velocities of the rising dispersed oil droplets and settling dispersed water droplets in the water and oil phases, respectively:

$$v = \frac{gD^2(\rho_d - \rho_c)}{18\mu_c}. \quad (1)$$

In the latter equation g denotes the gravitational acceleration, D is the droplet diameter, $\rho_{d,c}$ are the densities of dispersed and continuous phases, respectively, and μ_c is the dynamic viscosity of the continuous phase. Although the dynamics of a gravity separator might seem simple to come up with, such separation could be very complex to deal with. There are several phenomena that add extra complexity to such models. These include:

1) *Sedimentation, which is the settling of a droplet.*
2) *Creaming, which describes the rising of a droplet.*
3) *Coalescence, the creation of larger droplets, due to collision of smaller droplets,*
4) *Breakage, describes the creation of smaller droplets, due to the bursting of a bigger droplet.*
5) *Emulsion layer presence on the interface between oil and water phases, which hinders particles to enter their respective phases as they rise or settle.*
6) *Foaming presence, which is the accumulation of droplets between the oil and gas phases, which could result in erroneous level measurements, and control.*

This model adopts a dynamic, and more control-oriented approach to model the gravity separator.

2.1. Mathematical model

In this simple mathematical model, the only phenomena considered and adopted are sedimentation and creaming of oil and water particles. Other aspects, such as coalescence and breakage, emulsion layers and foaming, are not considered. Therefore, particles are assumed to leave directly into their bulk phases once they reach the interface. This assumption is valid when a highly effective demulsifier is dosed into the separator at the inlet. Here are some additional simplifications to the model:

1) *Gas is immediately separated, and no gas emulsions are present.*
2) *No liquid dispersed in the gas phase and vice versa.*
3) *Plug flow regime, with an average horizontal velocity for each phase including droplets.*
4) *Static distribution of droplets throughout the volume, due to the absence of coalescence and droplet breakage.*
5) *Droplets are assumed spherical.*
6) *The vertical velocity of the droplet is determined by Stokes's law (1) without any correction factor. The fact that coalescence is not taken into consideration allows this compensation.*
7) *The water and liquid levels instantly level out with respect to changes for in- and outflows.*
8) *Constant densities.*
9) *Ideal behavior and Isothermal compression for the gas phase.*

In this model, the feed to the separator is the main disturbance, and the three outflows for gas, oil and water, are the manipulated variables. In the inlet part the liquid enters with a water cut α and an oil cut $\beta=1-\alpha$, however, not all the water will directly enter the water phase immediately, but a fraction of it will disperse in the oil. The same holds for the incoming oil. Thus, it is necessary to define the following fractions. First the fraction of inflowing directly entering its respective bulk phase, in this case, the fraction of oil directly entering the oil ϕ_{oo}. Then, water directly entering the bulk water phase ϕ_{ww}. From these fractions, the fractions that are being dispersed in the other phase can be calculated where $\phi_{ow}=1-\phi_{oo}$ (fraction of oil initially dispersed in water), and $\phi_{wo}=1-\phi_{ww}$ (fraction of water initially dispersed in oil).

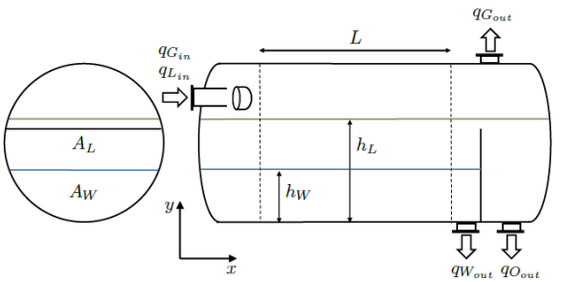

Figure 4: simplified schematic representation of a gravity separator, including a cross sectional view on the left. On the right, liquid inflow, gas inflow, and a dish head to achieve rough separation of gas and liquid. Further, gas, oil, water controlled levels and the manipulated outflows. In addition, the active separation zone with length L. [11]

2.1.1. Dynamics

A. Liquid level rate of change equations

The rate of change in the separator volume can be calculated from the volumetric in- and outflows:

$$\frac{dV_L}{dt} = q_{L,in} - q_{L,out}. \tag{2}$$

In the latter equation, the volumetric inflow of liquid is a given parameter which is subject to some disturbances. However, the liquid outflow is not, rather it is split into two, oil and water outflow, which are manipulated variables. This gives the following:

$$\frac{dV_L}{dt} = q_{L,in} - q_{Oil,out} - q_{Water,out} \tag{3}$$

The next step is to find the expression for the liquid cross sectional area that is shown in Figure 4. This area is denoted by A_L and is the area of a circular segment with liquid height h_L. It is calculated according to the following equation with r denoting the radius of the separator:

$$A_L = \frac{r^2}{2}\left[2arccos\left(\frac{r-h_L}{r}\right) - sin\left(2arccos\left(\frac{r-h_L}{r}\right)\right)\right]. \tag{4}$$

To find the dependency of the latter equation on h_L with respect to time, equation (4) is derived with respect to time and yields

$$\frac{dA_L}{dt} = \frac{dh_L}{dt}\left[r^2\left(\frac{1-\cos\left(2arccos\left(\frac{r-h_L}{r}\right)\right)}{\sqrt{h_L(2r-h_L)}}\right)\right]. \tag{5}$$

Given that $V_L = A_L L$, the change of the area with respect to time is

$$\frac{dA_L}{dt} = \frac{dV_L}{dt} L \tag{6}$$

Substituting (5) and (3) in (6) and solving it for $\frac{dh_L}{dt}$,the following differential equation is obtained:

$$\boldsymbol{\frac{dh_L}{dt} = \frac{dV_L}{dt} \frac{\sqrt{h_L(2r-h_L)}}{r^2 L\left(1-\cos\left(2arccos\left(\frac{r-h_L}{r}\right)\right)\right)}}. \tag{7}$$

B. Water level rate of change
By adopting the same procedure as for the liquid level dynamics, the rate of change of water level can be calculated as

$$\frac{dV_W}{dt} = q_{W,in} - q_{W,out}. \tag{8}$$

These quantities can be found by utilizing the cuts and fractions defined before, which yields:

$$\frac{dV_W}{dt} = q_{L,in}(\alpha\phi_{ww} + \beta\phi_{ow}) - q_{W,out}. \tag{9}$$

The respective area of the water is then

$$A_W = \frac{r^2}{2}\left[2arccos\left(\frac{r-h_W}{r}\right) - sin\left(2arccos\left(\frac{r-h_W}{r}\right)\right)\right], \tag{10}$$

with its time derivative

$$\frac{dA_W}{dt} = \frac{dh_W}{dt}\left[r^2\left(\frac{1-\cos\left(2arccos\left(\frac{r-h_W}{r}\right)\right)}{\sqrt{h_W(2r-h_W)}}\right)\right]. \tag{11}$$

Combining equations (9) and (11) into

$$\frac{dA_W}{dt} = \frac{dV_W}{dt} L \tag{12}$$

leads to the dynamic differential equation of the water level inside the separator

$$\boldsymbol{\frac{dh_W}{dt} = \frac{dV_W}{dt} \frac{\sqrt{h_W(2r-h_W)}}{r^2 L\left(1-\cos\left(2arccos\left(\frac{r-h_W}{r}\right)\right)\right)}}. \tag{13}$$

B. Pressure rate of change
Using the assumption of the ideal gas behavior, the ideal gas relation can be used to obtain a deferential representation of the rate of change of pressure with respect to time.

$$pV_G = n_G RT \tag{14}$$

The derivative of this equation with respect to time is

$$\frac{dp}{dt}V_G + p\frac{dV_G}{dt} = RT\frac{dn_G}{dt}. \quad (15)$$

The isothermal compression assumption disregards change in temperature with time. Further, the change in the gas volume with respect to time will be the negative value of the total change in liquid volume with respect to time, since it is a closed system and the pressure on top must remain constant. Hence we get:

$$\frac{dV_G}{dt} = -\frac{dV_L}{dt} = -\left(q_{L,in} - q_{L,out}\right). \quad (16)$$

To find the molar gas change with respect to time, the molecular weight (M_G) and density (ρ_G) of entering gas are used to convert the volumetric in- and outflows into the correct units, this yields:

$$\frac{dn_G}{dt} = \frac{\rho_G}{M_G}\left(q_{G,in} - q_{G,out}\right). \quad (17)$$

Finally, the dynamic representation of the gas in the gravity separator is obtained by combining equations (15), (16) and (17).

$$\frac{dp}{dt}V_G = RT\frac{\rho_G}{M_G}\left(q_{G,in} - q_{G,out}\right) + p\left(q_{L,in} - q_{L,out}\right). \quad (18)$$

2.1.2. Steady state particle calculations

A. Spatial discretization

In order to obtain more accurate results for droplet balances, a discretized model is employed. The separator it therefore divided into N_S volumetric segments along the x axis (Figure 5). There by, equations (7) and (13) are calculated for each volumetric segment, and it is assumed that the average value of their sum, is the corresponding level for each phase. An additional noteworthy assumption is that there exist no delays throughout the level calculations for each phase between each segment. Meaning, that the inflow and outflow are divided by the number of segments N_S equally and added or subtracted from each volumetric segment without any holdup.

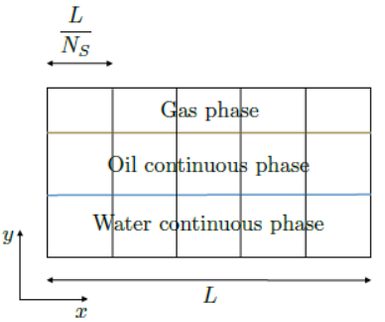

Figure 5: Discretization of the active separation zone. [11]

B. Simplified droplet balances

In this section 10 classes of droplets are taken into consideration. These classes have a diameter of c_i={50,100,150,…,500}µm, i= 1,2,..., 10. The first step in the calculation process is to find how long a droplet will remain in a segment in the positive horizontal x axis direction, namely horizontal residence time t_h.

$$t_h = \frac{L}{N_S}\frac{A_{w,o}}{q_{w,o}}, \tag{19}$$

In the latter expression $\frac{L}{N_S}$ represents the length of the active separation zone for each segment, as indicated in Figure 5 . $A_{w,o}$ indicates the respective cross-sectional area of water and oil, and $q_{w,o}$ is the volumetric inflow for each phase. The second step is to find the vertical residence time t_v for each droplet class. t_v is found by taking the distance a particle will travel (in this case the level) in the y direction, and dividing it by the vertical velocity v_v from *Stokes' law* (1):

$$t_v = \frac{h_{w,o}}{v_v} = \frac{18\mu_c}{g{c_i}^2(\rho_d - \rho_c)}. \tag{20}$$

The idea here is to compare t_v with t_h . If a certain particle class has a larger vertical residence time than its horizontal, then it is expected that not all the particles of this class will leave to their respective phase, rather some will remain in the outlet. However, if it was the opposite case, meaning t_v is less than t_h, then given that these values are calculated for worst case scenarios (largest distance), it is expected that all particles in this class will leave to their respective bulk phase. It is important to mention that once a particle reaches the interface, it is assumed that its volume will be subtracted from the segment it is leaving and added to the segment it is entering. In other words, once a particle reaches the interface it instantly enters the approached segment.

This model will investigate the behavior of each class of particles based on their residence time in both horizontal and vertical directions, and their position relative to the oil-water interface. By measuring the distance, a particular particle has traveled during its presence in a specific segment, the number of particle that have left the segment can be easily calculated from the distance traveled to level ratio. Furthermore, the total volume can be calculated from the basic

volume of the sphere equation. Finally, a volumetric flow is obtained be dividing the respective volume by the residence time, which in this case is the horizontal residence time. This volumetric flow exiting a certain segment is also entering another, thus can be added or subtracted from equations (2) and (8).

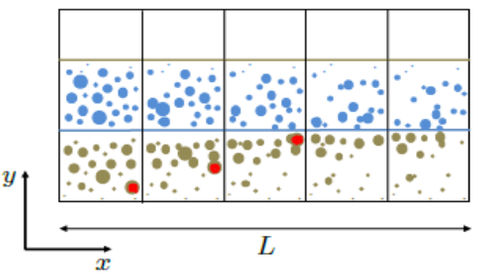

Figure 6: The evolution of particles over the active separation zone. Particles enter from the left and leave from the right. Blue particles are water dispersed in oil, which are sedimenting throughout their travel. Brownish particles are oil dispersed in water, rising to the oil phase. The red marked particles indicate the change of same particle position over each segment. [11]

Because of the changing water and liquid levels overall the process, a factor $F_{O,W}$ is introduced. $F_{O,W}$ will keep the volume-ratio between dispersed droplets and continuous phase constant for changing levels. The factor F will change the initial distribution of water in oil and oil in water. Hence, if levels are rising F increases and the opposite is correct. Introducing the latter, will allow further investigation of how changes in levels affect efficiencies for steady state calculations. Here is how this factor is calculated:

$$F_w = \alpha\phi_{wo}\,\frac{L}{N_S}A_o\,\frac{1}{\delta\cdot\psi'} \tag{21}$$

$$F_o = \beta\phi_{ow}\,\frac{L}{N_S}A_W\,\frac{1}{\delta\cdot\psi'} \tag{22}$$

$$\delta = F_i[1\ \ 5\ \ 10\ \ 50\ \ 100\ \ 100\ \ 50\ \ 10\ \ 5\ \ 1]10^8 \tag{23}$$

Equation (23) describes the initial distribution δ of the particle classes. Where Ψ is the volume of the particle classes.

2.2. Controller Design

PI controller

The PI control scheme is named after its two correcting terms, whose sum constitutes the manipulated variable (MV). The proportional, and integral terms are summed to calculate the output of the PI controller. Thus, the PI algorithm is described as:

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau \,, \tag{24}$$

Where, u(t) is defined as the controller input, K_p is the proportional gain, K_i is the integral gain, e(t) is the error between measured and reference value, t is the time, τ is an integration variable. This equation can be represented in the Laplace domain as:

$$L(s) = K_p + K_i/s, \tag{25}$$

where s is the complex frequency, and $K_i = K_p/T$, here T is the integral time in seconds.

SIMC tunings

In practice, for plants with a large time constant, one has to wait a long time for the process to settle. At this time, one might approximate it as an integrating process. Therefore, the ration of the gain and time constant is more important than their individual values. The figure below shows how to obtain these values.

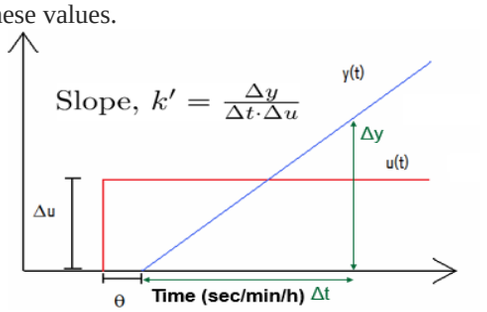

Figure 7: Step response for integrating process, here u is the input, y output, k' is the ratio between gain and time response, and θ is the time delay [12].

After finding these informations, one can obtain the controller's parameters according to the SIMC rules:

$$K_c = \frac{1}{k'} \cdot \frac{1}{\tau_c + \theta} \tag{26}$$

$$\tau_I = \frac{4}{k' K_c} \tag{27}$$

In equation (26), τ_c denotes the controller time constant, this value is determined based on the desired performance, for example smooth or tight control. Furthermore, K_c is the controller gain, and τ_I is integral time constant.

Smooth level control

In this method the slowest possible control is desired, because it is subject to acceptable disturbance rejection. In general, smooth control should be present for processes where setpoint tracking is not required, in this case level control. Now the desired closed loop response time τ_c should be larger than the delay (in this case the delay is zero, thus a good estimate for τ_c is required) to achieve slower control, smoother input functions, less sensitivity to measurement noise and better robustness. However, to smoothen disturbance in the flow, the minimum value for K_c should be equal to the ratio of expected flow change (input disturbance) Δq_d [m^3/s] and to the largest allowed variation in level ΔV_{max} [m^3]. Further, if the operator insists on integral action, the minimum value for τ_I should be obtained by working backwards from equation (28) to (27).

$$K_c \geq \frac{|\Delta q_d|}{|\Delta V_{max}|} \tag{28}$$

3. Modelica software

Modelica is the primary modeling language used in this project. The main objective of this work was to study Modelica, and to incorporate the gravity separator model into Modelica. This section gives a brief introduction to the modeling tool and its features. Furthermore, it presents some of the classes built to describe the mathematical model of the gravity separator, and shows some of the work done by Statoil engineers over gravity separator modeling using Modelica from the Subpro Library.

3.1. Introduction to Modelica

Modelica is a programming language that allows specification of mathematical models of complex natural or man-made systems, for example, for the purpose of computer simulation of dynamic systems where behavior evolves as a function of time. Modelica is also an object oriented equation-based programming language, oriented toward computational applications with high complexity requiring high performance [17]. The concept of object orientation in modelica is viewed as a structuring concept which is used to manage the complexity of large system descriptions. Modelica models are essentially declarative mathematical descriptions where dynamic systems are expressed in a declarative way through equations. Hence, rather than giving a detailed stepwise algorithm on how to reach a desired goal, modelica is inspired by mathematics, where it is common to state or declare what holds. This will make the code easier to change without introducing errors [18]. Here are the four most important features of Modelica:

1. *Modelica is mostly based on equations instead of assignment statements. This gives acausal modeling that gives better reuse of classes, since equations don't specify a certain data flow direction.*
2. *Modelica has multidomain modeling capability.*
3. *Modelica is an object-oriented language with a general class concept that unifies classes, generics and general subtyping into a single language construct.*
4. *Modelica is an architectural description language, it offers graphical modeling experience.*

3.2. Modelica classes and the gravity separator

Modelica programs are built from classes, also called models. From a class definition, it is possible to create any number of objects that are known as instances of that class. Modelica classes contain elements, such as variable declarations and equation sections containing equations. The equations will specify the behavior of instances of that class. The class concept is the key to reuse of modeling knowledge in Modelica. In many situations it is advantageous to be able to express generic patterns for models or programs. Instead of writing many similar pieces of code with essentially the same structure, this can be avoided by directly expressing the general structure of the problem and providing the special cases as parameters. To illustrate the latter, consider the following class, *LevelSep*.

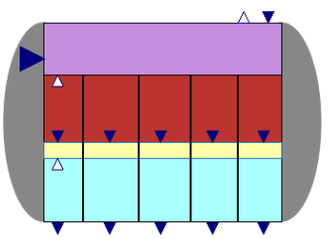

Figure 8: Graphical representation of the class *LevelSep*, gravity separator model in Modelica, with 3 phases gas, oil and water from top to bottom. Yellow layer exists merely for graphical simplicity when connecting signals.

The model *LevelSep* takes in in- and outflows and based on these values it will describe the dynamic states of the gravity separator. The model's three main variables are the liquid level, water level and pressure. When this model is inherited it requires some parameter specifications, these parameters include the production rates, phases cuts, particle classes, number of segments, etc.

Figures 9 and 10 show generic classes which perform steady state calculations of oil and water droplets in their respective bulk phases. In order to have a descritized model, the calculations are modeled for each segment. Since there is a similar pattern in the calculations for each segment in the separator, the same classes *particleSepOil* and *particleSepWater* are inhereted in *OOutOfW* and *WOutOfO*, which create 5 instances of *particleSepOil* and *particleSepWater* to perform the calculations for each segment. The variables and parameters of *OOutOfW* and *WOutOfO* are specified in *LevelSep*, thus, *OOutOfW* and *WOutOfO* must be connected to *LevelSep*. In return *OOutOfW* and *WOutOfO* will provide *LevelSep* with new variables, which are the volumetric in- and outflows between phases.

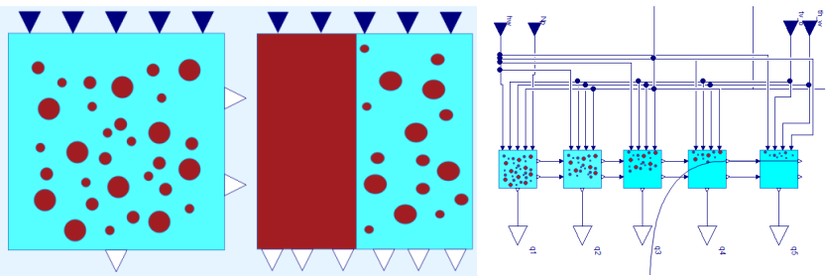

Figure 9: Steady state calculations for oil particles in water. Starting from the left we have the following classes: *particleSepOil*, *OOutOfW*, and graphical modeling of *OOutOfW* which shows it inherits *particleSepOil* 5 times.

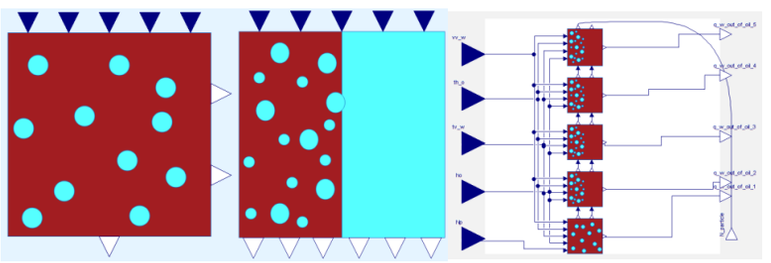

Figure 10: Steady state calculations for water particles in oil. Starting from the left we have the following classes: *particleSepWater*, *WOutOfO*, and graphical modeling of *WOutOfO* which shows that this class merely inherits *particleSepOil* 5 times and connects.

As can be seen from Figures 9, 10 and later 12, Modelica offers graphical modeling and instead of typing all the various instances and connectors (blue and white triangles) it allows programmers to save enormous time and merely drag and drop classes and connect them graphically. Hence, the architectural description language feature.

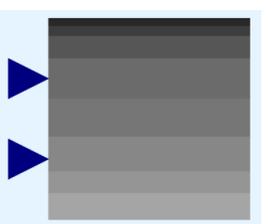

Figure 11: Efficiency calculator, this block takes in the initial volume of dispersed particles at the inlet and final volume of dispersed particles at the outlet and returns efficiencies.

The whole model is connected in *GravitySepSimMO*, this is considered the main class and the topmost layer. *GravitySepSimMO* creates instances of the models presented above or from the Modelica library and connects them according to the mathematical model description. In this model, the initial in- and outflows, steps, disturbances and tuning parameters are specified and then simulated to obtain results for further analysis.

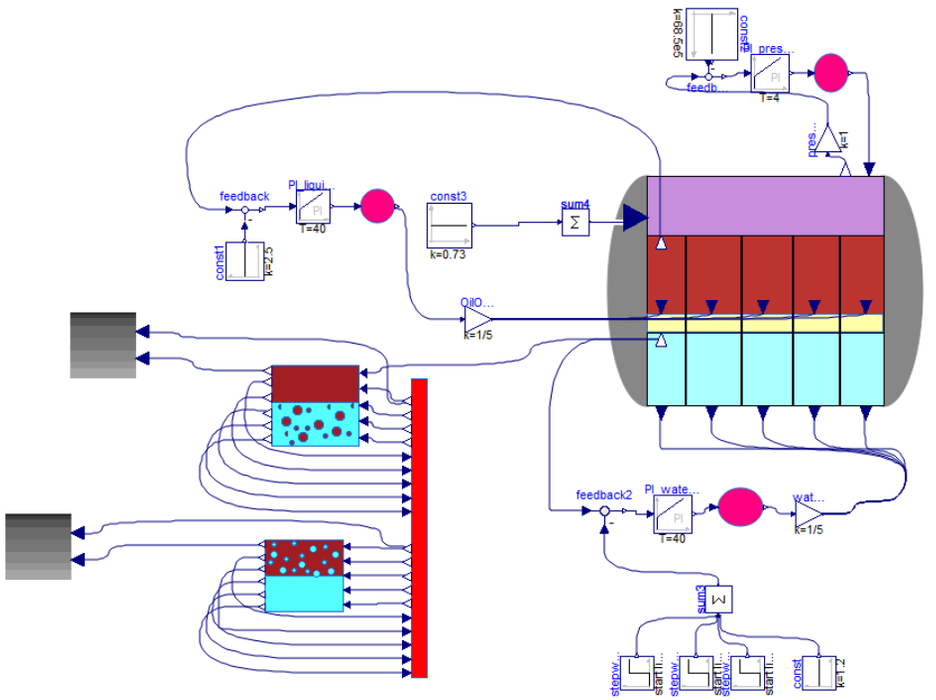

Figure 12: The main class of the entire model *GravitySepSimMO*. *GravitySepSimMO* inherits and creates several instances of PI controllers, gravity separator dynamics and state calculations, and efficiency blocks.

3.3. Subpro library

The Subpro Library in Modelica developed by Statoil offers many general models.

1. *Medium models with multiphase models for gas, oil and water.*
2. *Equipment models, such as separators, compressors, pumps, and valves.*
3. *Control models including PI(D) controllers, and event tables*.

Figure 13 shows a 3 phase gravity separator from the Subpro library. The physical measure of the separator drum is considered for two cases when the level of oil after the weir is low or when the oil level is normal as shown in Figure 13. In the implementation of the model, it is necessary to calculate the liquid level as a function of the liquid volume. In the latter implementation it is assumed that when reducing the amount of oil, the oil level on the feed side of the weir stays at the level of the weir while oil level is further reduced on the outlet side. Further, the model assumes that the phases are at steady state, hence it is built merely to obtain dynamic results without any steady state calculation. Figure 14 shows how the model was implemented for simulation. There the oil, gas and water feeds are entering separately. This model is incomparable with the mathematical model in this project, since the Statoil separator is designed merely to show how the levels of oil and water changes relative to the weir.

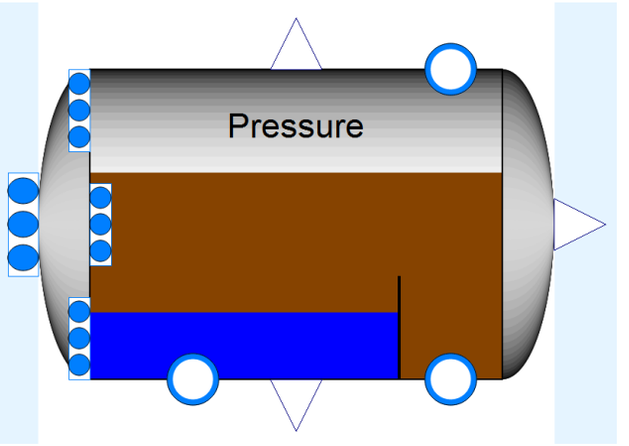

Figure 13: Dynamic gravity separator from the Subpro library

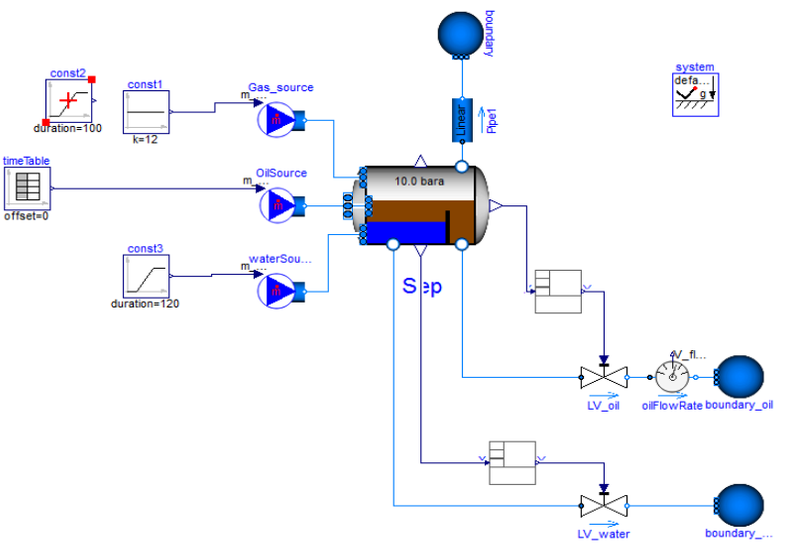

Figure 14: Simulation of the dynamic gravity separator model, given separate feeds and not mixed oil water and gas.

4. Simulations

The model introduced in Section 2 is a differential algebraic equation system, and is solved and simulated using Modelica as was briefly introduced in Section 3.2. Simulation parameters are mostly taken from [16]. These parameters include initial and future production rates, and in addition geometrical parameters. This section begins by finding suitable tuning parameters, followed by our investigation of the states and stability of the system with initial and future production rates, while simultaneously investigating steady state results. Finally, some simulations using the Subpro separator are performed and discussed. The same plots are generated with same production rates as in [11] for the sake of comparison and discussion later.

4.1. Controller design and tuning

The controller designs are obtained using an open loop step response method. Figure 15 shows the step response for the water level. The same method was used for all of the three dynamic states and similar behaviours are obtained. Based on these analysis PI controller tunings are found for processes with no time delay and τ_c determined manually. Table 1 shows the results for two different values of τ_c.

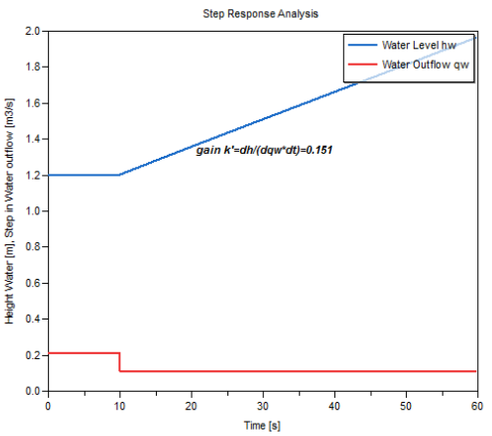

Figure 15: Open loop step response. Step of u=-0.1 is done on the water outflow, consequently the level increases. The pattern of increase indicates an integrating process.

Table 1: Tuning parameters for PI controllers of water and oil levels, as well as pressure with different values for τ_c.

k'=0.151	**Water Level PI-Controller Tunings**	
τ_c [s]	5	10
K_c	1.32	0.66
τ_I [s]	20	40
k'=0.180	**Liquid Level PI-Controller Tunings**	
τ_c [s]	5	10
K_c	1.11	0.55
τ_I [s]	20	40
k'=0.1.89e-6	**Gas Phase Pressure PI-Controller Tunings**	
τ_c [s]	1	2
K_c	1.89e-6	9.46e-7
τ_I [s]	4	8

The measurement and outflow responses for the various tunings with the different values of τ_c are plotted in Figures 16 and 17. As indicated there with increasing values of τ_c the control time to reach steady state, and the measurement variation increases. In such separation processes the control should be smooth and allow large variations in levels. However, it is hard to determine which is better without having an actual separator.

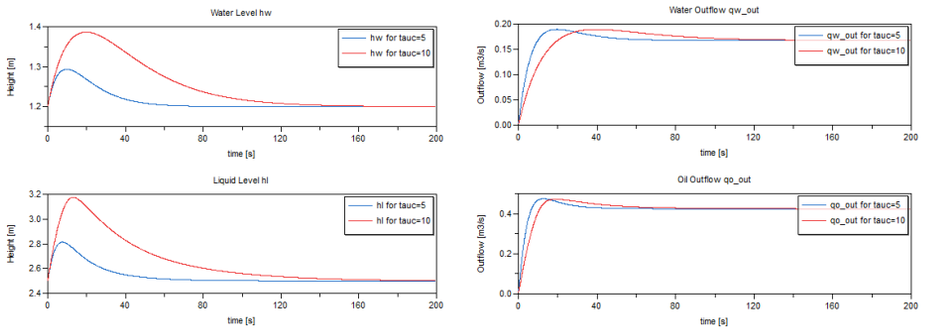

Figure 16: On the left water and liquid measurements, on the right their corresponding manipulated variables water and oil outflows.

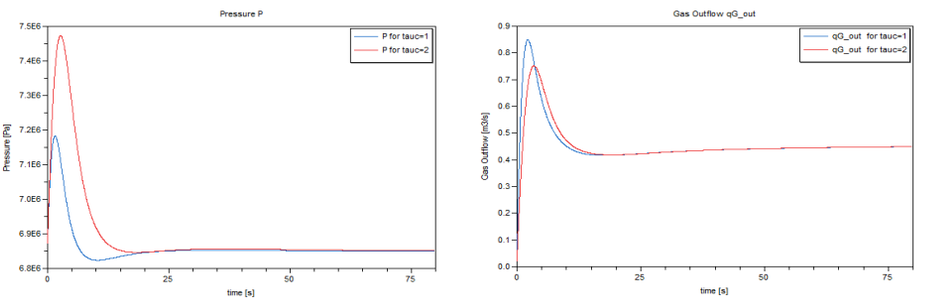

Figure 17: On the left pressure measurements, on the right gas outflow.

The tunings used in the rest of the simulations are slightly modified to obtain more accurate and steady state simulations. In these the integral time is left untouched, however, the gain of the controller for liquid and water levels is set to 1 as indicated in Table 2. This will reduce the value for the first peak, however the control will remain with the same time scale, meaning the time to reach this peak is the same. Figure 18 shows the simulation results of such tunings.

Table 2: tunings used for the rest of the simulations

	Water Level	**Liquid Level**	**Pressure**
K_c	1	1	1.89e-6
τ_I [s]	40	40	4

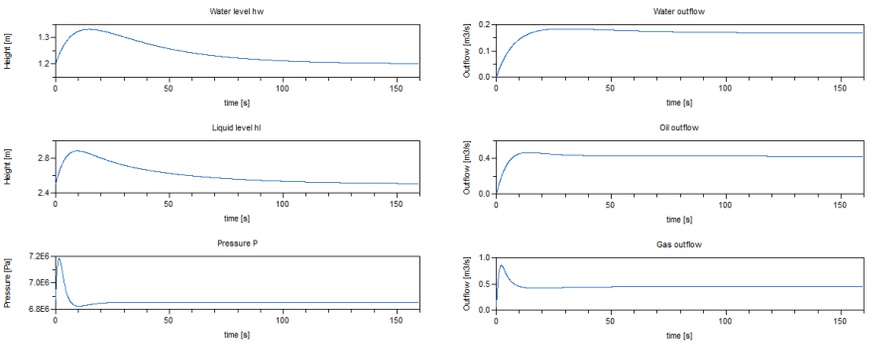

Figure 18: On the left height, and pressure measurements, on the right corresponding outflows.

4.2. Simulation for initial production rates

The simulations presented in this section are based on the production rates specified in [11]. The parameters are set to fit the case for the startup of the well with a low water cut $\alpha=0.135$, and gas production rate $q_{G,in}=0.456\ m^3/s$, liquid production rate $q_{L,in}=0.59\ m^3/s$, a setpoint for the water level $h_{W,s}=1.2\ m$ and for the liquid level $h_{L,s}=2.5\ m$.

In Figure 19 the three state variables are displayed in the plots on the left hand side together with the respective control variables on the right hand side. In this simulation a series of stepwise setpoint changes of +0.2 m were employed every 500 s to the water level starting at an initial value of 1.2 m in order to find out how this change will affect the overall efficiency of oil separation from the water phase. The left hand side in Figure 19 shows that the PI controllers immediately act to control the system and move it to its new nominal values. The right hand side in Figure 19 shows how their respective outflows are adjusted in order to reach the new nominal value. Obviously, $q_{w,out}$ saturates for some seconds in order to raise the level of water faster.

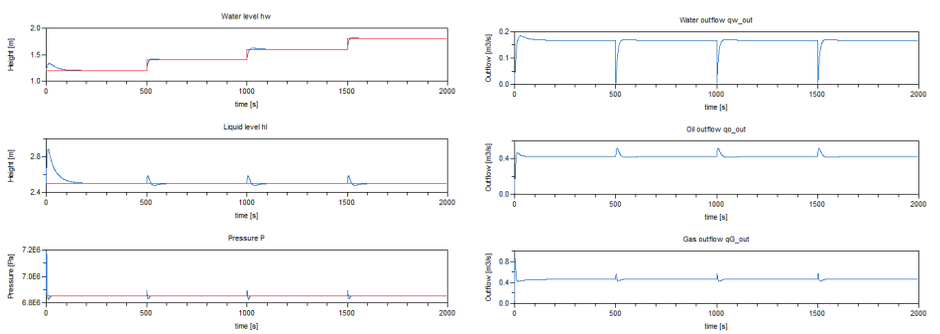

Figure 19: State variables water, liquid levels, and pressure together with their respective manipulated variables the outflows over 2000 seconds simulation.

Figure 20 shows different efficiencies of oil removal from the water phase (red) and of water removal in the oil phase (blue). The efficiencies were subject to the changes in water level every 500 s. The figure shows 3 different scenarios for fractions of oil and water initially dispersed in the water and oil phases respectively. Namely, $\phi_{ow} = \phi_{wo} = 0.3$ (solid lines), $\phi_{ow} =$

ϕ_{wo} *=0.4* (dashed lines) and ϕ_{ow}= ϕ_{wo}*=0.5* (dotted lines). The results indicate that the highest separation efficiencies are obtained for small fractions of oil initially entering the water phase. However, a reverse phenomenon is observed for water initially entering the oil phase. The efficiency is calculated such that a 99% denotes that 1% of the initial dispersed volume is still present at the outlet. Figure 20 indicates that overall oil separation from water phase is generally better than water removal from the oil phase. Figure 20 also shows that the factors F for oil and water droplets calculations are displaying correct and desired behavior. An increase in the level of water leads to an increase in the oil droplet distribution and consequently the volume in that segment, and a decrease in water droplets distribution, since, oil level is decreasing to keep total liquid level at a constant setpoint.

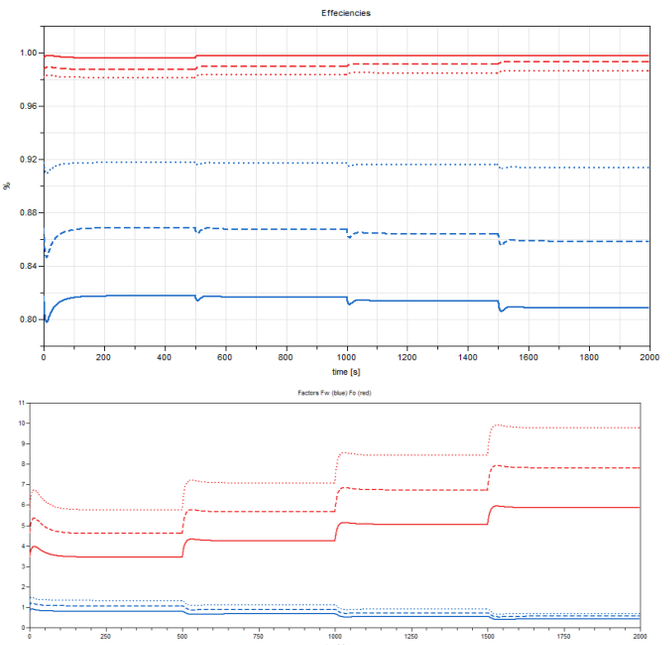

Figure 20: On top displayed efficiencies of removal of oil from water (red), and water from oil (blue). Bottom displayed the factors F of oil and water droplets over time. ϕ_{ow} = ϕ_{wo}= 0.3 (solid lines), ϕ_{ow} = ϕ_{wo} =0.4 (dashed lines) and ϕ_{ow}= ϕ_{wo}=0.5 (dotted lines).

Figure 21 shows the amount of oil droplets at the water outlet over time and their respective diameter. The top corresponds to results when the fraction of oil initially entering water is ϕ_{ow} = ϕ_{wo}= *0.3*, in the middle ϕ_{ow} = ϕ_{wo}= *0.4*, and the bottom plot is for ϕ_{ow} = ϕ_{wo}= *0.5*. As indicated in Figure 21, the number of oil droplets for each class increases from top to bottom plots for both increasing the fractions ϕ_{ow}, ϕ_{wo} as well as water level. However, as time progresses and when changes in the level of water and for the fractions ϕ_{ow}, ϕ_{wo} are applied, the cut-off size at 250 µm remains the same. The latter indicates that all particles with a diameter larger than 250 µm will exit to the oil bulk phase before they reach the outlet.

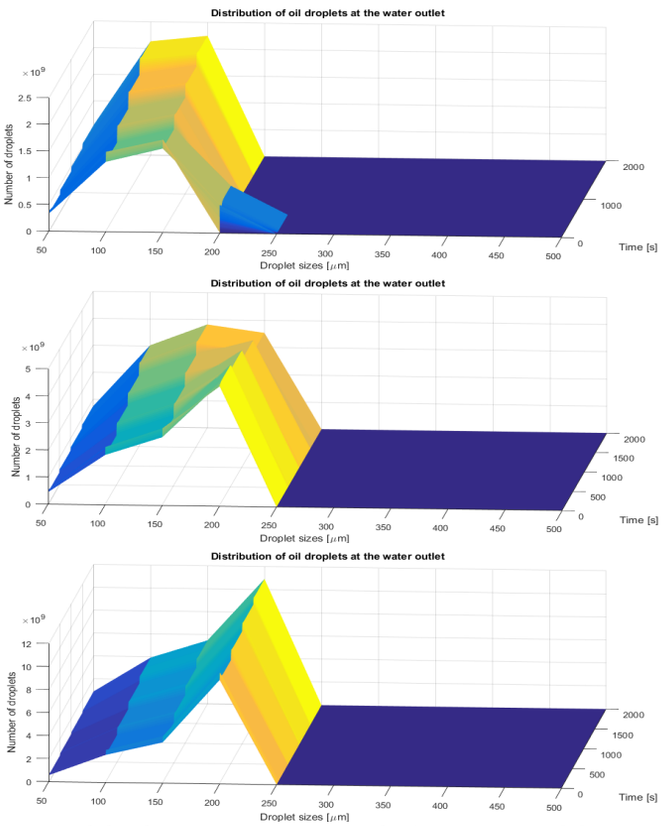

Figure 21: Amount of oil droplets going into water outlet for $\phi_{ow} = \phi_{wo} = 0.3$, $\phi_{ow} = \phi_{wo} = 0.4$ and $\phi_{ow} = \phi_{wo} = 0.5$.

4.3. Simulation for future production rates

The simulations concluded in this section are for the scenario when the well is at the end of its lifetime, then the water cut will be much higher $\alpha = 0.475$, gas outflow will remain $q_{G,in} = 0.456\ m^3/s$, and liquid inflow will increase to compensate for the oil cut loss of $q_{L,in} = 0.73\ m^3/s$. Step changes on the water level setpoint are done in the same way as in Section 4.2.

Figure 22 shows the three state variables on the left, and their respective manipulated variables on the right. In this case the PI controllers work to keep the system stable with smooth control. When having step changes on the water level the manipulated variable does not reach a saturation point, that is a good observation since it is better to control the level with a manipulated variable that does not reach a saturation point and keeps the control freely, especially when disturbances occur. Note that at the beginning of the control the levels are allowed to vary. In the case of the liquid level it is about 40 cm away from its setpoint and that is fine because it allows dampening especially when there are disturbances in the feed.

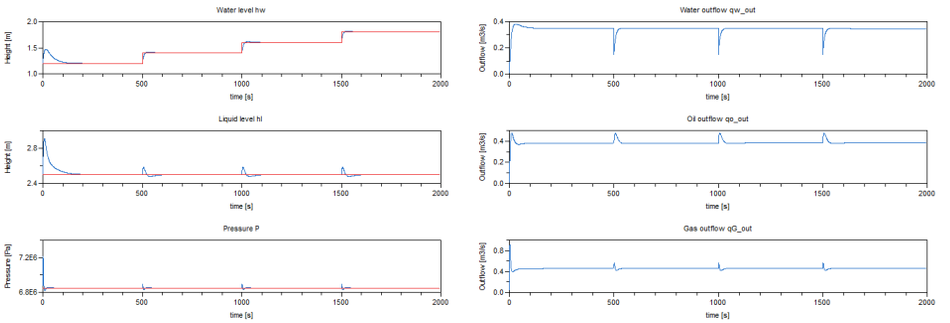

Figure 22: State variables water, liquid levels, and pressure together with their respective manipulated variables the outflows over 2000 seconds simulation.

In Figure 23 the efficiencies of oil and water removal from their respective dispersed phases are shown together with the factor F for water and oil droplets. As can be seen, for future production rates efficiencies are highest for the lowest fractions of oil initially entering the water phase and with increasing water level. While, the reverse phenomenon is obtained for the water removal efficiency. This trend is similar for initial production rates when the well is at the start of its lifetime, however, here overall efficiencies decrease in comparison to the simulations done in Section 4.2. In addition, changes in the fractions ϕ_{ow}, ϕ_{wo} do not show significant effect on the efficiency in this case. When it comes to the factor F for oil and water the significant differences are obtained for different values of ϕ_{ow} *and* ϕ_{wo} especially for F_w when compared to Figure 20.

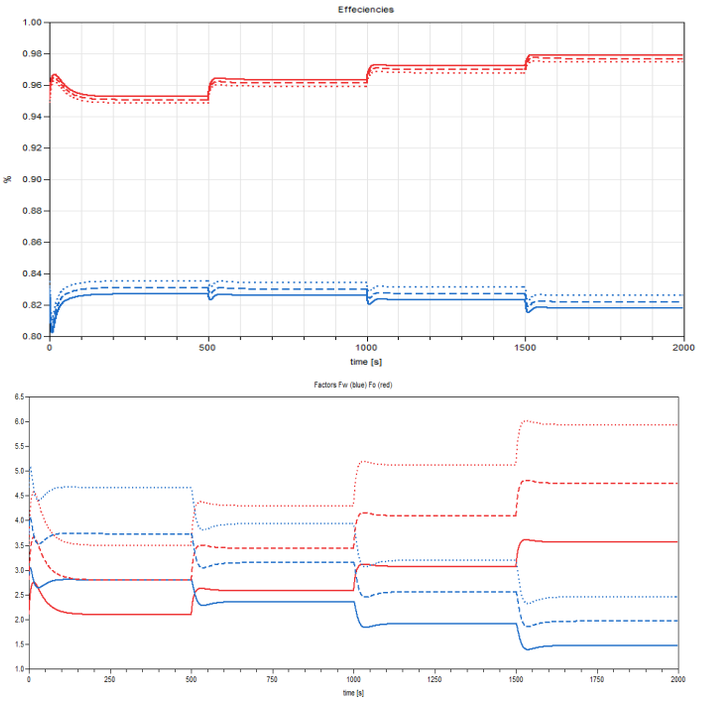

Figure 23: On top displayed efficiencies of removal of oil from water (red), and water from oil (blue). Bottom displayed the factors F of oil and water droplets over time. $\phi_{ow} = \phi_{wo} = 0.3$ (solid lines), $\phi_{ow} = \phi_{wo} = 0.4$ (dashed lines) and $\phi_{ow} = \phi_{wo} = 0.5$ (dotted lines).

Figure 24 as in Section 4.2 shows the amount of oil droplets for each class at the water outlet with respect to time for the new production rates. The arrangement of the plots is similar where The top corresponds to results when the fraction of oil initially entering water is $\phi_{ow} = \phi_{wo} = 0.3$, in the middle $\phi_{ow} = \phi_{wo} = 0.4$, and the bottom plot is for $\phi_{ow} = \phi_{wo} = 0.5$. In contrary to Figure 21 the amount of oil droplets present at the water outlet increases significantly as the fractions increase and production rate is changed. In addition, the cut-off droplet size becomes 300 µm. the latter indicates that the separation process becomes slightly worse than in Section 4.2 as more particles are not removed.

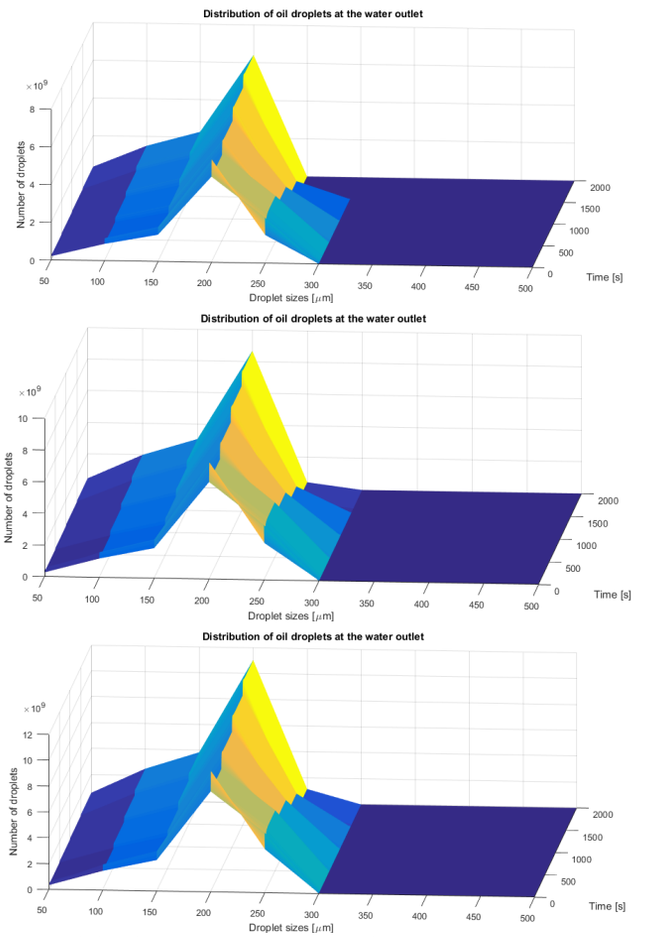

Figure 24: Amount of oil droplets going into water outlet for $\phi_{ow} = \phi_{wo} = 0.3$, $\phi_{ow} = \phi_{wo} = 0.4$ and $\phi_{ow} = \phi_{wo} = 0.5$.

4.4. Simulations with disturbances

This section investigates the effects of disturbances in the feed on the system's efficiency for initial and future production rates. The disturbances were introduced to the system through the feed every 200 seconds with $-/+ 0.1\ m^3/s$ in each step on the the liquid feed rate $q_{L,in}$. It is important to mention that the disturbance applied in the simulation is not based on actual data collected rather randomly selected steps merely for the sake of research.

A. Simulation for initial production rates

The simulations presented in this section are based on the production rates specified in [11], similar to Section 4.2. the parameters are set to fit the case for the startup of the well with a low water cut $\alpha=0.135$, and gas production rate $q_{G,in}=0.456\ m^3/s$. However, the liquid feed rate $q_{L,in}=0.59\ m^3/s$ is subject to change of -/+ $0.1\ m^3/s$ every 200 seconds. The water and liquid level are kept at their setpoints as specified earlier, $h_{W,s}=1.2\ m$ and $h_{L,s}=2.5\ m$, respectively.

As can be seen in Figure 25 the controllers' manipulated variables respond perfectly when disturbances occur. The results show no extreme behavior of outflows over time when disturbances happen. The latter are expected results since the simulations done in 4.1 make sure that the controller is fit for several scenarios when disturbances, and step changes are introduced.

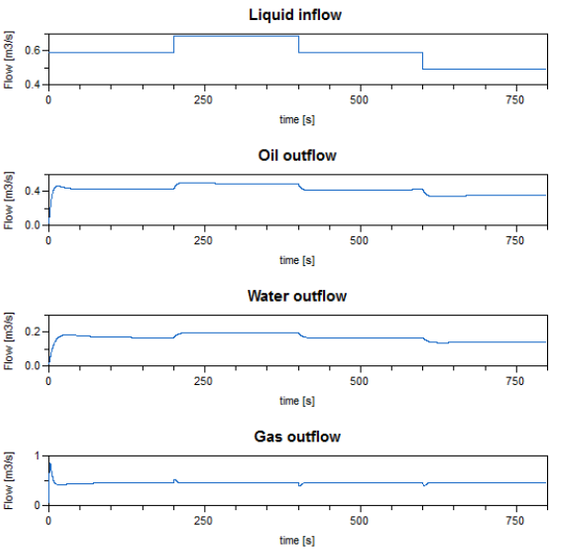

Figure 25: Effect of disturbance on in- and outflows of liquid oil, water and gas over 800 seconds simulation.

Figure 26 shows the effect of the disturbances on the separation of oil and water droplets. As can be seen from Figure 26, for initial production rates the disturbances do not have a significant effect on the oil in water separation efficiency, and this because of the small water cut on the initial stages. However, when it comes to water in oil separation, the efficiency decreases for high liquid inflow and increases with low liquid inflow, as expected. The reason for the latter, is that this model adapts a plug flow regime, with an average horizontal velocity for each phase including droplets. Therefore, for decreasing liquid inflow the residence time increases which gives more time for droplets to separate. In addition, Figure 26 displays the factors where no significant changes are observed.

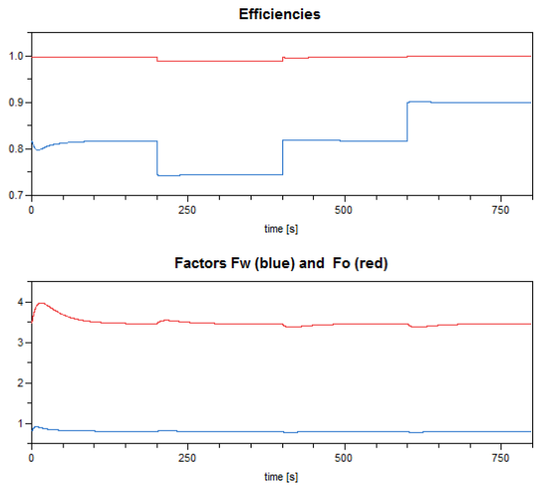

Figure 26: Effect of disturbance on separation efficiencies of liquid oil and water droplets, as well as factor calculations, over 800 seconds simulation.

B. Simulation for future production rates

The parameters here are set to fit the case for a well that is at the end of its lifetime with a high water cut $\alpha=0.475$, and gas production rate $q_{G,in}=0.456\ m^3/s$. However, the liquid production rate $q_{L,in}=0.73\ m^3/s$ is subject to change of -/+ 0.1 m^3/s every 200 seconds. The water and liquid level are still kept at their setpoints as specified earlier, $h_{W,s}=1.2\ m$ and $h_{L,s}=2.5\ m$, respectively.

Here the same plots as in the earlier case are presented for the future production rates. In Figure 27 the in- and outflows are presented. As can be seen, the controller works very well to reduce the effect of disturbances, without having the need to execute extreme changes on the outflows, which are the manipulated variables.

.

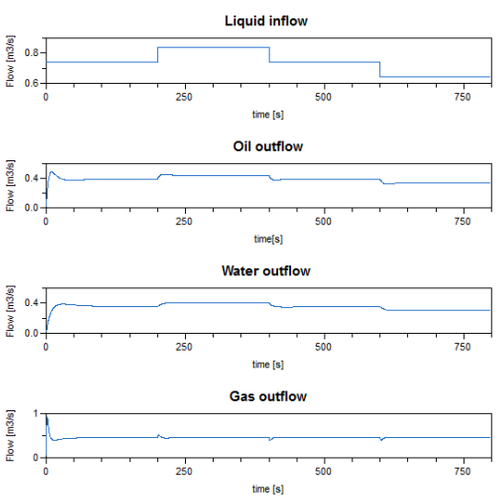

Figure 27: Effect of disturbance on in- and outflows of liquid, oil, water and gas over 800 seconds simulation.

In addition, Figure 28 shows the effect of disturbances on the efficiencies for the separation of oil and water droplets in their dispersed phases. For the case of water droplets in oil phase the simulations show similar results, the efficiency of the water separation will increase with decreasing inflow. For the case of oil droplets separation, unlike for initial production rates, the disturbances have a significant effect on the efficiencies. The simulations indicate that with decreasing inflow the efficiencies rise and vice versa. This observation occurs due to the time residence increase. The droplets will have longer time to separate since their horizontal velocity is decreased while their vertical velocity remains the same.

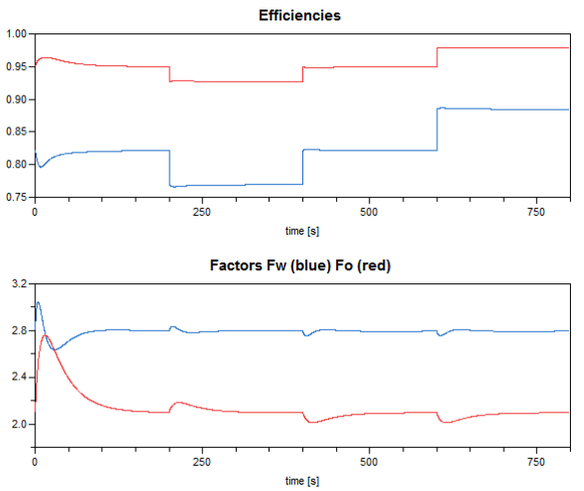

Figure 28: Effect of disturbance on separation efficiencies of oil and water droplets, as well as factor calculations, over 800 seconds simulation.

4.5. Subpro Gravity Separator Model Simulations

In this section the model presented in Figure 14 in Section 3.3 is simulated for 6000 seconds. This model keeps the gas flow constant and uncontrolled and rather controls only the liquid and water levels by manipulating their respective outflow valves as indicated in Figure 29. Figure 29 also shows that the control is implemented smoothly, where the integral time τ_I is equal to 700 s and 500 s with K_c corresponding to 15 and 10 respectively. Furthermore, the inlet feeds are separated initially and enter the separator through 3 different ports, gas, oil and water in dispersed form. The latter is done merely for simplification when disturbances are applied. The gas inlet is set to 12 kg/s, and is composed of the two components oil and gas. Oil inlet is set to 5.55 kg/s and is composed of the two components oil and water. Finally, the water inlet is composed of 3 components water, heavy and light oil droplets. The model does not include any steady state calculations, but rather adopts empirical data for steady state calculations which assumes that all phases separate before exiting the separator. The purpose of this model is merely to obtain the dynamic states, such as water and liquid level.

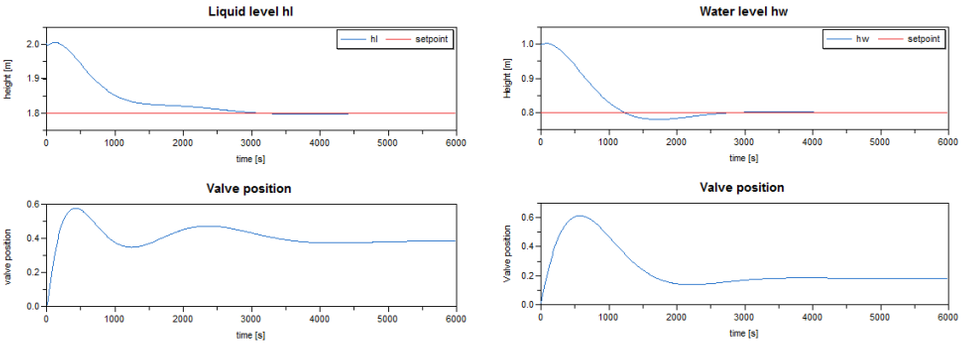

Figure 29: Liquid and water levels, as well as their manipulated variables the valve positions.

The simulations are also done with some disturbances, as indicated in Figure 30. The oil inlet was disturbed at time 1000 s with a step change from 5.70 kg/s to 9.70 kg/s of +/- 4 kg/s , and water is disturbed at time 3600s with a ramp with a duration of 120s that increases the water inflow from 3 kg/s to 4 kg/s. The resulting liquid and water dynamic states are displayed in Figure 31, where the simulations are carried out for the period of 6000 s.

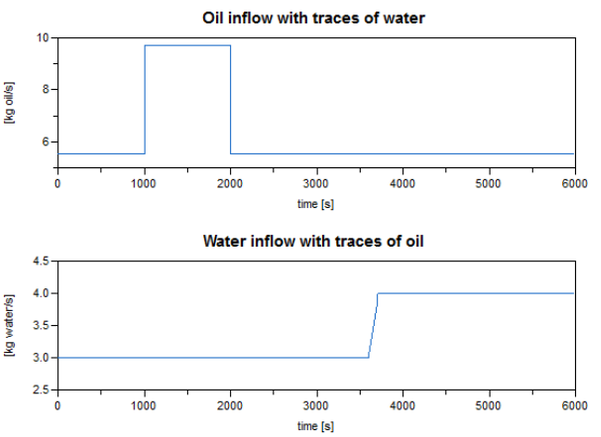

Figure 30: Oil and water Inflow over time.

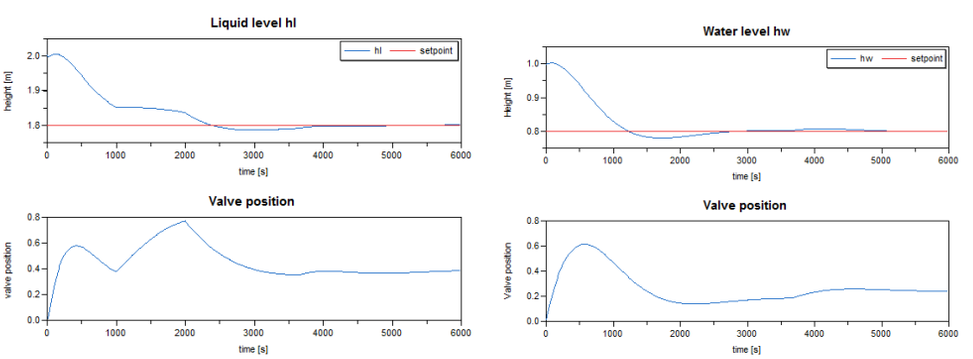

Figure 31: Liquid and water levels, as well as their manipulated variables the valve positions.

Conclusion

In subsea application it is beneficial to separate gas, oil and water phases at early stages in the process in order to obtain as pure single phase streams as possible. The work done in this project emphasizes on a three phase subsea gravity separator model. Which is one of the separation techniques commonly used for bulk separation of phases. The gravity separator allows particles in the tank to settle, sediment, or cream, given their density differences. This project adopts a simple mathematical model which takes into consideration merely sedimentation and creaming phenomena, and simplifies the model by not considering other aspects, such as coalescence, breakage or emulsion layers, but to name a few. Furthermore, the model employs spatial discretization to obtain more accurate results for droplet balances and investigates the behavior of each class of different particle sizes in each segment. Hence, in future work it will simplify the modeling approach when incorporating other phenomena into the model, which have significant influence on separation (e.g. adding demulsifiers). on separation. The work was primarily done in Modelica, where an existing Matlab model was incorporated into Modelica. Modelica is an object oriented, equation based programming language that has the feature of graphical modeling. In this work Modelica was studied and used for the purpose of computer simulation of dynamic systems where behavior evolves as a function of time.

The advantage of modeling in Modelica, is that this approach can scale to larger problems. Such as including other separation units, extending separation units, or reusing classes with similar objectives. In addition, Modelica is topologically based because that is the best way to approach plant modeling.

One of the goals of this project was to investigate the effect of water levels on the overall separation process efficiency for water in oil. The simulation results show, that larger water levels increase the quality of the water outlet stream. On the contrary the oil outlet stream purity is less influenced by the water level changes. Another aim of this project was to investigate the effect of initial separation on the overall quality of streams. The results of the simulations show that a better separation and higher efficiencies are achieved when the initial separation factors for water and oil, ϕ_{ow} and ϕ_{wo} respectively, are smaller. Furthermore, the cut-off droplet sizes at the water outlet for initial and future production rates are 250 µm and 300 µm, respectively. These results presented here correspond largely to the results obtained in the Matlab model done by Backi, and concludes that the model was successfully incorporated to Modelica.

Finally, some simulations were done to investigate the Subpro gravity separator, where the dynamic states of the separator were tested for disturbances. The model requires additional work to be able to compare it with the gravity separator modeled in this project. In addition, the classes and models created in Modelica in this project, could be very useful for future use in the Subpro library mainly to introduce steady state and efficiency calculations.

References

[1] McClimans, O., Fantoft, R., 2006. Status and new developments in subsea processing. In: Offshore Technology Conference, 1-4 May,Houston, Texas, USA.URL https://www.onepetro.org/conference-paper/ ,OTC-17984-MS .

[2] Statoil ASA, 2015. Subsea processing on Tordis. Accessed: 2016-11-06. URL http://www.statoil.com/en/TechnologyInnovation/FieldDevelopment/AboutSubsea/SubseaProcessingOnTordisIOR/Pages/default.aspx .

[3] Clifford Neal Prescott. Subsea separation and processing of oil, gas and produced water - past, present and future: Why we need it now. 2012.

[4] A. Aadal, A. Djuraev, C. Costa, H. Malik, M. Dilma, S. Tianqi. Liquid-Liquid Separation Subsea. 03/05/2016.

[5] Tianguang Fan, et al. (2002). "Evaluating Crude Oils by SARA Analysis" Society of Petroleum Engineers, Proceeding of SPE/DOE Improved Oil Recovery Symposium, 13–17 April 2002, Tulsa, Oklahoma.

[6] The pharmaceutics and compounding labaratory, 2016-11-06. URL http://pharmlabs.unc.edu/labs/emulsions/agents.htm .

[7] Oil demulsification,2016-11-07. URL http://petrowiki.org/Oil_demulsification#Coalescence .

[8] Subsea Separation and Processing of Oil, Gas & Produced Water , Past, Present and Future, Why we need it now. Accessed: 2016-11-10. URL: http://www.forum.rice.edu/wp-content/uploads/2011/06/011312RT_Prescott.pdf .

[9] In line separation, Accessed on 2016-11-10.
URL: http://www.statoil.com/en/technologyinnovation/fielddevelopment/topside/in-lineseparation/pages/default.aspx .

[10] Troll Statoil. Accessed 2016-11-10.
URL: http://www.statoil.com/en/TechnologyInnovation/FieldDevelopment/AboutSubsea/SubseaPresentation/Pages/Troll.aspx .

[11] C.J. Backi and S. Skogestad. A simple gravity model for separation efficiency evaluation incorporating level and pressure control. April 2016.

[12] S. Skogestad. Simple analytic rules for model reduction and PID controller tuning. Modeling, Identification and Control, 2004, col.25 No.2, 85-120.

[13] Advances in Multiphase flow and Heat Transfer, p.86, Volume 3; Volume 2012, Lixin Cheng, Dieter Mewes, Bentham Science Publishers, Jan 1, 2012.

[14] A.F Sayda and J. H. Taylor. Modeling and Control of Three-Phase Gravity Separators in Oil Production Facilities. In proceedings of the 2007 American Control Conference, New York City, USA, July 2007.

[15] A. Hallanger, C. Michelsen, F. Soenstaboe, and T. Knutsen. A Simulation Model for Three-Phase gravity Separators. In Proceedings of the 1996 SPE Annual Technical Conference and Exhibition, pages 695-706, Denver, CO, USA, October 1996.

[16] A. P. Laleh, W. Y. Svrcek, and W. D. Monnery. Computational Fluid Dynamics-Based Study of an Oilfield Separator- Part II: An Optimum Design. *Oil and Gas Facilities*, 2(1):52-59, February 2013.

[17] Modelica Association, Modelica – A Unified Object-Oriented Language for Physical Systems Modeling. Version 1.4 December 15, 2000.

[18] Peter Fritzson. Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica. 12.12.2016, IEEE Press, A John Wiley & Sons, Inc., Publication.

Appendix

i. LevelSep Model code

```
model LevelSep

   Real Al " cross sectional area liquid level (oil and water)";
   Real Vl "volume of liquid in separator [m3]";
   Real Vg "volume of liquid in separator";
   Real Aw(start=2.8101) "cross sectional area water";
   Real Ao "cross sectional area oil";
   Real qw_a;// inflow to water phase with oil //
   Real qo_a;// inflow to oil phase with water //
   Real dVdt_w[Ns] "dynamics array of each segment of water segment";
   Real dVdt_l[Ns] "dynamics array of each segment of liquid segment";
   Real F_o "correction factor for oil droplets";
   Real F_w "correction factor for water droplets";
   Real Volume[10] "volume of a sphere droplet";
   Real Vp_total_oil[10] "total volume of oil droplet with diameter ci";
   Real Vp_total_water[10] "total volume of water droplet with diameter ci";
   Real Volume_oil "total volume of oil droplets";
   Real Volume_water "total volume of water droplets";
   Real rhs_water
   "right hand side of differential equation without dynamics dV/dt";
   Real rhs_liquid
   "right hand side of differential equation without dynamics dV/dt";

    //Real hw "total level of water";
    //Real vv_o[10] "vertical velocity Oil in water";
    //Real vv_w[10] "vertical velocity water droplets in Oil";

   //Real hl(start=2.5) "level of liquid in separator";
   //Real hw( start=1.2) "level of water in separator";
   //Real p( start=68.7*100000) "pressure inside separator";

   /* here a indicates inflow , b outflow.
   note in this example all values are set to 1 initially
   until further info about dimensions are given */

   //parameter Real ql_a=0.59;   // disturbance given number this number is divided by five 0.59/5
   //parameter Real qG_a=0.456;

  //parameter Real qG_b=0.456;
  //parameter Real qw_b=0.3;   // manipulated variable to control liquid level in sep unit
  //parameter Real qo_b=0.3;  // alternative manipulated variable
  //parameter Real Al=50;      // This value should be checked !!

   constant Real r=1.65;     // radius of crosssectional area of sep
   constant Real L=10;     // length of effective separation area this value is divided by 5: 10/5
   constant Real Vsep=85.53;     // volume of separator unit

  // Initial Production parameters

    /*parameter Real ql_a=0.59; disturbance given number this number is divided by five 0.59/5 */
/*
   parameter Real qG_a=0.456;
   parameter Real alpha=0.135 "water cut ";
   parameter Real beta=1-alpha "Oil cut";
   parameter Real th_ow= 0.3 "fraction of oil going to water";
   parameter Real th_ww= 1-th_wo "fraction of water going to water";
   parameter Real th_wo= 0.3 "fraction of water going to oil";
   parameter Real th_oo= 1-th_ow "fraction of oil going to oil";
*/
  // Future Production parameters

   parameter Real qG_a=0.456;
   parameter Real alpha=0.475 "water cut ";
   parameter Real beta=1-alpha "Oil cut";
```

```
  parameter Real th_ow= 0.3 "fraction of oil going to water";
  parameter Real th_ww= 1-th_wo "fraction of water going to water";
  parameter Real th_wo= 0.3 "fraction of water going to oil";
  parameter Real th_oo= 1-th_ow "fraction of oil going to oil";

 // Pressure rate of change additionl parameters

  parameter Real R=8.314; // gas constant
  parameter Real T=328.5;   // in kelvin
  parameter Real d_G=49.7;     // density of gas
  parameter Real M_G=0.01604;     // molar weight of gas

  // the following parameters and variables are for t.residence calculations

 /*Real ho "level of oil phase"; note this value is obtained from average values of hw and hl after running simulations.
                          this value is used to calculat horizontal residence time*/

              //Oil in Water
  //Real th_w "horizontal residence time in water phase";
  /*Real tv_o[10]
   "vertical residence time of oil particle classes in water phase";*/
  //Real Aw(start=2.8101) "cross sectional area water";

              // Water in Oil
  //Real th_o "horizontal residence time in oil phase";
  //Real tv_w[10] "vertical residence time of water particle classes in oil phase";
  //Real Ao "cross sectional area oil";

  parameter Integer n=10; // number of particle classses//
  parameter Real c[10]= {50e-6,100e-6,150e-6, 200e-6,250e-6,300e-6,350e-6,400e-6,450e-6, 500e-
6}; //Particle classes diameter first element smallest, and last largest//
  parameter Integer Ns=5; // number of segments//
  final constant Real pi=Modelica.Constants.pi;
  parameter Real Np_oil_init[10]={1e8,5e8,1e9,5e9,1e10,1e10,5e9,1e9,5e8,1e8};

  parameter Real Np_water_init[10]={1e8,5e8,1e9,5e9,1e10,1e10,5e9,1e9,5e8,1e8};

    // parameter Real qw_a=ql_a*(alpha*th_ww+beta*th_ow);// inflow to water phase with oil //
 // parameter Real qo_a=ql_a*(beta*th_oo+alpha*th_wo);// inflow to oil phase with water //

  //Real qw_a;// inflow to water phase with oil //
 //Real qo_a;// inflow to oil phase with water //

  parameter Real u_o=0.001; //oil viscosity kg (s m )^-1
  parameter Real u_w=0.0005;//water viscosity kg (s m )^-1

  parameter Real g= 9.8; // gravity acceleration m s^-2

  parameter Real d_o= 831.5; // oil density kg m^-3
  parameter Real d_w= 1030; // water density kg m^-3

  // Adding dVdt variable which depends on mass transfer in each segment  //

  //Real dVdt_w[Ns] "dynamics array of each segment of water segment";

 Modelica.Blocks.Interfaces.RealInput qw_b1
  annotation (Placement(transformation(extent={{-4,-4},{4,4}},
       rotation=-90,
       origin={-56,0}),
       iconTransformation(extent={{-3,-3},{3,3}},
       rotation=-90,
       origin={-61,-45})));
 Modelica.Blocks.Interfaces.RealOutput h_water(start=1.2) annotation (Placement(
       transformation(extent={{-70,10},{-58,22}}), iconTransformation(
       extent={{-3,-3},{3,3}},
       rotation=90,
       origin={-61,-13})));
 Modelica.Blocks.Interfaces.RealInput qG_b
  annotation (Placement(transformation(extent={{2,38},{8,44}}),
       iconTransformation(
       extent={{-3,-3},{3,3}},
```

```
      rotation=-90,
      origin={45,61})));
Modelica.Blocks.Interfaces.RealOutput p(start=68.7e5)
 annotation (Placement(transformation(extent={{-6,-6},{6,6}},
      rotation=90,
      origin={-8,66}), iconTransformation(
      extent={{-3,-3},{3,3}},
      rotation=90,
      origin={33,61})));
Modelica.Blocks.Interfaces.RealInput qo_b1
 annotation (Placement(transformation(extent={{-4,-4},{4,4}},
      rotation=-90,
      origin={-56,16}),
      iconTransformation(
      extent={{-3,-3},{3,3}},
      rotation=-90,
      origin={-61,1})));
Modelica.Blocks.Interfaces.RealOutput h_liquid(start=2.5) annotation (
    Placement(transformation(extent={{-70,30},{-60,40}}), iconTransformation(
      extent={{-3,-3},{3,3}},
      rotation=90,
      origin={-61,29})));

      /*Try to add the above below so we wont have a mess in
      addition try to find a way to put it in a for loop*/

       //water input/control divided into 5 segments //

 Modelica.Blocks.Interfaces.RealInput qo_b2
 annotation (Placement(transformation(
       extent={{52,-14},{62,-4}}),    iconTransformation(
      extent={{-3,-3},{3,3}},
      rotation=-90,
      origin={-35,1})));
 Modelica.Blocks.Interfaces.RealInput qo_b3
 annotation (Placement(transformation(
       extent={{64,-12},{72,-4}}),   iconTransformation(
      extent={{-3,-3},{3,3}},
      rotation=-90,
      origin={-9,1})));
 Modelica.Blocks.Interfaces.RealInput qo_b4
 annotation (Placement(transformation(
       extent={{72,-12},{80,-4}}),   iconTransformation(
      extent={{-3,-3},{3,3}},
      rotation=-90,
      origin={17,1})));
 Modelica.Blocks.Interfaces.RealInput qo_b5
 annotation (Placement(transformation(
       extent={{80,-12},{88,-4}}),  iconTransformation(
      extent={{-3,-3},{3,3}},
      rotation=-90,
      origin={43,1})));

      //water input/control divided into 5 segments //

 Modelica.Blocks.Interfaces.RealInput qw_b2
 annotation (Placement(transformation(
       extent={{54,-24},{60,-18}}),    iconTransformation(
      extent={{-3,-3},{3,3}},
      rotation=-90,
      origin={-35,-45})));
 Modelica.Blocks.Interfaces.RealInput qw_b3
 annotation (Placement(transformation(
       extent={{62,-24},{70,-16}}),   iconTransformation(
      extent={{-3,-3},{3,3}},
      rotation=-90,
      origin={-9,-45})));
 Modelica.Blocks.Interfaces.RealInput qw_b4
 annotation (Placement(transformation(
       extent={{70,-24},{78,-16}}),   iconTransformation(
      extent={{-3,-3},{3,3}},
```

```
      rotation=-90,
      origin={17,-45})));
Modelica.Blocks.Interfaces.RealInput qw_b5
 annotation (Placement(transformation(
       extent={{82,-24},{90,-16}}),  iconTransformation(
      extent={{-3,-3},{3,3}},
      rotation=-90,
      origin={43,-45})));

      // liquid output/measurement divided into 5 segments //

Modelica.Blocks.Interfaces.RealOutput vv_o[10] annotation (Placement(
      transformation(
      extent={{-10,-10},{10,10}},
      rotation=180,
      origin={-162,-22}), iconTransformation(
      extent={{-2,-2},{2,2}},
      rotation=180,
      origin={-190,-46})));
Modelica.Blocks.Interfaces.RealOutput tv_o[10] annotation (Placement(
      transformation(
      extent={{-10,-10},{10,10}},
      rotation=180,
      origin={-182,-50}), iconTransformation(
      extent={{-2,-2},{2,2}},
      rotation=180,
      origin={-190,-58})));
Modelica.Blocks.Interfaces.RealOutput th_w annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=180,
      origin={-162,-34}), iconTransformation(
      extent={{-2,-2},{2,2}},
      rotation=180,
      origin={-190,-52})));
/*Modelica.Blocks.Interfaces.RealInput q1(start=0)
  annotation (Placement(transformation(extent={{-192,-42},{-180,-30}}),
      iconTransformation(extent={{-184,-34},{-180,-30}})));*/
Modelica.Blocks.Interfaces.RealInput q1
  annotation (Placement(transformation(extent={{-204,-78},{-188,-62}}),
      iconTransformation(extent={{-192,-66},{-188,-62}})));

Modelica.Blocks.Interfaces.RealOutput Np_w[10]( start={1e8, 5e8, 1e9,5e9,1e10,1e10,5e9,1e9,5e8,1e8}) annotation (Placement(
      transformation(
      extent={{-10,-10},{10,10}},
      rotation=180,
      origin={-160,-6}),  iconTransformation(
      extent={{-2,-2},{2,2}},
      rotation=180,
      origin={-190,-104})));
Modelica.Blocks.Interfaces.RealInput q2 annotation (Placement(
      transformation(extent={{-202,-82},{-188,-68}}), iconTransformation(
       extent={{-192,-72},{-188,-68}})));
Modelica.Blocks.Interfaces.RealInput q3 annotation (Placement(transformation(
       extent={{-204,-90},{-188,-74}}), iconTransformation(extent={{-192,
        -78},{-188,-74}})));
Modelica.Blocks.Interfaces.RealInput q4 annotation (Placement(transformation(
       extent={{-168,-96},{-152,-80}}), iconTransformation(extent={{-192,-84},
        {-188,-80}})));
Modelica.Blocks.Interfaces.RealInput q5 annotation (Placement(transformation(
       extent={{-204,-102},{-188,-86}}),iconTransformation(extent={{-192,
        -90},{-188,-86}})));
Modelica.Blocks.Interfaces.RealOutput vv_w[10] annotation (Placement(transformation(
       extent={{-180,-116},{-160,-96}}),  iconTransformation(
      extent={{-2,-2},{2,2}},
      rotation=180,
      origin={-190,-128})));
Modelica.Blocks.Interfaces.RealInput q_w_out_of_oil_1 annotation (Placement(
      transformation(extent={{-204,-148},{-188,-132}}), iconTransformation(
       extent={{-192,-136},{-188,-132}})));
Modelica.Blocks.Interfaces.RealOutput ho
  annotation (Placement(transformation(extent={{-150,-102},{-130,-82}}),
```

```
        iconTransformation(
        extent={{-2,-2},{2,2}},
        rotation=180,
        origin={-190,-110})));
  Modelica.Blocks.Interfaces.RealOutput th_o
   annotation (Placement(transformation(extent={{-118,-120},{-98,-100}}),
        iconTransformation(
        extent={{-2,-2},{2,2}},
        rotation=180,
        origin={-190,-122})));
  Modelica.Blocks.Interfaces.RealOutput tv_w[10]
   annotation (Placement(transformation(extent={{-174,-136},{-154,-116}}),
        iconTransformation(
        extent={{-2,-2},{2,2}},
        rotation=180,
        origin={-190,-116})));
  Modelica.Blocks.Interfaces.RealInput q_w_out_of_oil_2 annotation (Placement(
        transformation(extent={{-76,-126},{-60,-110}}), iconTransformation(
          extent={{-192,-142},{-188,-138}})));
  Modelica.Blocks.Interfaces.RealInput q_w_out_of_oil_5 annotation (Placement(
        transformation(extent={{-92,-72},{-78,-58}}), iconTransformation(extent={{-192,
          -160},{-188,-156}})));
  Modelica.Blocks.Interfaces.RealInput q_w_out_of_oil_4 annotation (Placement(
        transformation(extent={{-30,-118},{-14,-102}}), iconTransformation(
          extent={{-192,-154},{-188,-150}})));
  Modelica.Blocks.Interfaces.RealInput q_w_out_of_oil_3 annotation (Placement(
        transformation(extent={{-48,-136},{-32,-120}}), iconTransformation(
          extent={{-192,-148},{-188,-144}})));

  Modelica.Blocks.Interfaces.RealInput ql_a
   annotation (Placement(transformation(extent={{-108,6},{-68,46}}),
        iconTransformation(extent={{-80,34},{-68,46}})));
  Modelica.Blocks.Interfaces.RealOutput Np_o[10]( start={1e8,5e8,1e9,5e9,1e10,1e10,5e9,
      1e9,5e8,1e8}) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=180,
        origin={-116,-42}), iconTransformation(
        extent={{-2,-2},{2,2}},
        rotation=180,
        origin={-190,-40})));
equation

  // inflow calc
          qw_a=ql_a*(alpha*th_ww+beta*th_ow);
          qo_a=ql_a*(beta*th_oo+alpha*th_wo);
        // adding dVdt[] //

            dVdt_w[1]= ((ql_a/5)*(alpha*th_ww + beta*th_ow) - qw_b1)-q1+q_w_out_of_oil_1;
            dVdt_w[2]= ((ql_a/5)*(alpha*th_ww + beta*th_ow) - qw_b2)
   -q2+q_w_out_of_oil_2;
            dVdt_w[3]= ((ql_a/5)*(alpha*th_ww + beta*th_ow) - qw_b3)
   -q3+q_w_out_of_oil_3;
            dVdt_w[4]= ((ql_a/5)*(alpha*th_ww + beta*th_ow) - qw_b4)
   -q4+q_w_out_of_oil_4;
            dVdt_w[5]= ((ql_a/5)*(alpha*th_ww + beta*th_ow) - qw_b5)
   -q5+q_w_out_of_oil_5;

    dVdt_l[1]=((ql_a/5) - qw_b1 - qo_b1);
    dVdt_l[2]=((ql_a/5) - qw_b2 - qo_b2);
    dVdt_l[3]=((ql_a/5) - qw_b3 - qo_b3);
    dVdt_l[4]=((ql_a/5) - qw_b4 - qo_b4);
    dVdt_l[5]=((ql_a/5) - qw_b5 - qo_b5);
       //level of liquid //

   rhs_liquid =sqrt(h_liquid*(2*r - h_liquid))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - h_liquid)/r))));
   der(h_liquid)=sum(dVdt_l)*rhs_liquid;

  /*der(hl_1)= ((ql_a/5) - qw_b1 - qo_b1)*sqrt(hl_1*(2*r - hl_1))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hl_1)/r))));

  der(hl_2)= ((ql_a/5) - qw_b2 - qo_b2)*sqrt(hl_2*(2*r - hl_2))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hl_2)/r))));
  der(hl_3)= ((ql_a/5) - qw_b3 - qo_b3)*sqrt(hl_3*(2*r - hl_3))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hl_3)/r))));
```

```
 der(hl_4)= ((ql_a/5) - qw_b4 - qo_b4)*sqrt(hl_4*(2*r - hl_4))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hl_4)/r))));
 der(hl_5)= ((ql_a/5) - qw_b5 - qo_b5)*sqrt(hl_5*(2*r - hl_5))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hl_5)/r))));*/

      //levels of water//
/*
 der(hw_1)= dVdt_w[1]*sqrt(hw_1*(2*r - hw_1))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hw_1)/r))));
 der(hw_2)= dVdt_w[2]*sqrt(hw_2*(2*r - hw_2))*(1/(r^2*(L/5)))*(1/(1 - cos(2*
  acos((r - hw_2)/r))));
 der(hw_3)= dVdt_w[3]*sqrt(hw_3*(2*r - hw_3))*(1/(r^2*(L/5)))*(1/(1 - cos(2*
  acos((r - hw_3)/r))));
 der(hw_4)= dVdt_w[4]*sqrt(hw_4*(2*r - hw_4))*(1/(r^2*(L/5)))*(1/(1 - cos(2*
  acos((r - hw_4)/r))));
 der(hw_5)= dVdt_w[5]*sqrt(hw_5*(2*r - hw_5))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hw_5)/r))));*/

 // Right Hand Side for each segment with different dynamics since droplets are entering and exiting //
 rhs_water=sqrt(h_water*(2*r - h_water))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - h_water)/r))));
 /*
 rhs_water_1= dVdt_w[1]*sqrt(hw_water*(2*r - hw_water))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hw_water)/r))));
 rhs_water_2= dVdt_w[2]*sqrt(hw_water*(2*r - hw_water))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hw_water)/r))));
 rhs_water_3= dVdt_w[3]*sqrt(hw_water*(2*r - hw_water))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hw_water)/r))));
 rhs_water_4= dVdt_w[4]*sqrt(hw_water*(2*r - hw_water))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hw_water)/r))));
 rhs_water_5= dVdt_w[5]*sqrt(hw_water*(2*r - hw_water))*(1/(r^2*(L/5)))*(1/(1 - cos(2*acos((r - hw_water)/r))));
 */

 der(h_water)=sum(dVdt_w)*rhs_water;

 Al= 0.5*(r^2)*(2*acos((r - h_liquid)/r) - sin(2*acos((r - h_liquid)/r)));
 Vl= Al*L;
 Vg= Vsep - Vl;

   //hl= (hl_1 + hl_2 + hl_3 + hl_4 + hl_5)/5 "average of levels of segments";
 //hw= (hw_1 + hw_2 + hw_3 + hw_4 + hw_5)/5 "average of levels of segments";

 ho= h_liquid - h_water;
                                  // cross sectional area using averaged level for each phase
 Aw= 0.5*(r^2)*(2*acos((r - h_water)/r) - sin(2*acos((r - h_water)/r)));
                                    // for water
 Ao= Al - Aw;
        // cross sectional area oil only

 /* y = 1e-5* 1/(pi*r^2*L-r^2/2*(2*acos((r-hL)/r)-sin(2*acos((r-hL)/r)))*L) * (constants*(qGin-qGout) + 1e5*p*(qLin-qOut-
qWout));*/
 // pressure ODE

 der(p)= R*T*d_G*(1/Vg)*(1/M_G)*(qG_a - qG_b) + p*(ql_a - (qo_b1 + qo_b2 +qo_b3 + qo_b4 + qo_b5) - (qw_b1 + qw_b2 + q
w_b3 + qw_b4 + qw_b5))*(1/Vg);

  //Factor F calculations

 for i in 1:10 loop // calc volume of each particle
   Volume[i] =  4/3*pi*(c[i]/2)^3;
   Vp_total_oil[i]=  Np_oil_init[i]*Volume[i];
   Vp_total_water[i]=  Np_water_init[i]*Volume[i];
 end for;
 Volume_oil=sum(Vp_total_oil);
 Volume_water=sum(Vp_total_water);

 F_w=alpha*th_wo*(L/5)*Ao*(1/Volume_oil);
 F_o=beta*th_ow*(L/5)*Aw*(1/Volume_water);

 Np_o=F_o*Np_oil_init;
 Np_w=F_w*Np_water_init;

 //Np_o=Np_oil_init;
 //Np_w=Np_water_init;

 th_w= L*Aw*(1/Ns)*(1/qw_a); // note that here L/Ns  5 segments bitch //
 th_o= L*Ao*(1/Ns)*(1/qo_a); // note that here L/Ns  5 segments bitch //

 for i in 1:n loop
 tv_o[i]= h_water*18*u_w*(1/g)*(1/(c[i]^2))*(1/(d_w - d_o));
```

```
tv_w[i]= (-1)*ho*18*u_o*(1/g)*(1/(c[i]^2))*(1/(d_o - d_w));//i multiplied by (-1) to get positive value since water is going down
vv_o[i]= (-1)*g*(c[i])^2*(d_o-d_w)*(1/18)*(1/u_w);
vv_w[i]= g*(c[i])^2*(d_w-d_o)*(1/18)*(1/u_o);

end for;

annotation (
 Diagram(coordinateSystem(extent={{-180,-140},{100,100}}),
       graphics={
     Rectangle(
      extent={{-64,58},{4,0}},
      lineColor={28,108,200},
      fillColor={255,0,0},
      fillPattern=FillPattern.Solid),
     Rectangle(
      extent={{-64,34},{4,0}},
      lineColor={28,108,200},
      fillColor={42,42,42},
      fillPattern=FillPattern.Solid),
     Rectangle(
      extent={{-64,16},{4,0}},
      lineColor={28,108,200},
      fillColor={28,108,200},
      fillPattern=FillPattern.Solid),
     Text(
      extent={{28,56},{96,46}},
      lineColor={0,0,0},
      fillColor={231,231,231},
      fillPattern=FillPattern.Solid,
      textString="Liquid/water level measurement ",
      fontSize=14),
     Text(
      extent={{18,14},{90,0}},
      lineColor={0,0,0},
      fillColor={231,231,231},
      fillPattern=FillPattern.Solid,
      fontSize=14,
      textString="Liquid/water level input")}),
 Icon(coordinateSystem(extent={{-180,-140},{100,100}}),
     graphics={
     Ellipse(
      extent={{-48,58},{-88,-42}},
      lineColor={135,135,135},
      fillColor={135,135,135},
      fillPattern=FillPattern.Solid),
     Ellipse(
      extent={{72,58},{32,-42}},
      lineColor={135,135,135},
      fillColor={135,135,135},
      fillPattern=FillPattern.Solid),
     Line(points={{-68,32},{-68,-28}}, color={0,0,0}),
     Line(points={{52,32},{52,-28}}, color={0,0,0}),
     Rectangle(
      extent={{-68,58},{52,32}},
      lineColor={0,0,0},
      fillColor={198,142,220},
      fillPattern=FillPattern.Solid),
     Rectangle(
      extent={{-68,32},{52,-2}},
      lineColor={0,0,0},
      fillColor={188,52,47},
      fillPattern=FillPattern.Solid),
     Rectangle(
      extent={{-68,-10},{52,-42}},
      lineColor={0,0,0},
      fillColor={170,255,255},
      fillPattern=FillPattern.Solid),
     Rectangle(
      extent={{-68,-2},{52,-10}},
      lineColor={28,108,200},
      fillColor={255,255,170},
```

```
        fillPattern=FillPattern.Solid),
      Line(
        points={{-48,32},{-48,-42}},
        color={0,0,0},
        thickness=0.5),
      Line(
        points={{-20,32},{-20,-42}},
        color={0,0,0},
        thickness=0.5),
      Line(
        points={{30,32},{30,-42}},
        color={0,0,0},
        thickness=0.5),
      Line(
        points={{6,32},{6,-42}},
        color={0,0,0},
        thickness=0.5),
      Rectangle(
        extent={{-188,-30},{-180,-160}},
        lineColor={28,108,200},
        fillColor={255,0,0},
        fillPattern=FillPattern.Solid)}));
end LevelSep;
```

ii. particleSepOil model code

```
model particleSepOil1
  "calculates the volume of particles leaving the segment water , new number of particles present , new position of particles "

 // input Real hw_1;   // these are dynamic values and i want them to be display dynamic results later
  //input Real tv_o[10];
  //input Real th_w;
 // input Real vv_o[10];

 //output Real q1 "for first segment oil out of water";    //!!!!!!!!! do we have to put [10]
  //output Real N_particle[10];
  //output Real Position[10]; // same for those

Real last_particle_position[10](start=fill(0,10));
Real Vp[10] "volume of each particle";
Real Vp_total[10] "volume of total particles in each class";
Real n_new[10] "new Np";
//parameter Real Np[10]= {1e8, 5e8, 1e9,5e9,1e10,1e10,5e9,1e9,5e8,1e8};// these values should be adjusted after moving to the nex
t stage talk to christoph about this

protected
   parameter Real u_o=0.001; //oil viscosity kg (s m )^-1
   parameter Real u_w=0.0005;//water viscosity kg (s m )^-1

   parameter Real g= 9.8; // gravity acceleration m s^-2

   parameter Real d_o= 831.5; // oil density kg m^-3
   parameter Real d_w= 1030; // water density kg m^-3
   parameter Real c[10]= {50e-6,100e-6,150e-6, 200e-6,250e-6,300e-6,350e-6,400e-6,450e-6, 500e-
6}; //Particle classes diameter first element smallest, and last largest//

   final constant Real pi=Modelica.Constants.pi;                          //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!//

public
  Modelica.Blocks.Interfaces.RealInput hw annotation (Placement(transformation(
        extent={{-20,-20},{20,20}},
      rotation=-90,
      origin={-80,58}),               iconTransformation(extent={{-10,-10},
          {10,10}},
      rotation=-90,
      origin={-80,110})));

  Modelica.Blocks.Interfaces.RealOutput q1 annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
      rotation=-90,
```

40

```
      origin={-44,-12}),          iconTransformation(extent={{-10,-10},{
        10,10}},
      rotation=-90,
      origin={0,-110})));

  Modelica.Blocks.Interfaces.RealInput tv_o[10]
   annotation (Placement(transformation(extent={{-20,-20},{20,20}},
      rotation=-90,
      origin={-42,58}),
      iconTransformation(extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={-40,110})));
  Modelica.Blocks.Interfaces.RealInput th_w
   annotation (Placement(transformation(extent={{-20,-20},{20,20}},
      rotation=-90,
      origin={-8,60}),
      iconTransformation(extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={0,110})));
  Modelica.Blocks.Interfaces.RealInput vv_o[10] annotation (Placement(
      transformation(extent={{-20,-20},{20,20}},
      rotation=-90,
      origin={26,58}),                    iconTransformation(
       extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={40,110})));

  Modelica.Blocks.Interfaces.RealInput Np[10] annotation (Placement(
      transformation(
      extent={{-20,-20},{20,20}},
      rotation=-90,
      origin={64,58}), iconTransformation(
      extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={80,110})));
  Modelica.Blocks.Interfaces.RealOutput N_particle[10] annotation (Placement(
      transformation(
      extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={-4,-12}), iconTransformation(
      extent={{-10,-10},{10,10}},
      rotation=0,
      origin={110,40})));
  Modelica.Blocks.Interfaces.RealOutput Position[10] annotation (Placement(
      transformation(
      extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={30,-12}), iconTransformation(
      extent={{-10,-10},{10,10}},
      rotation=0,
      origin={110,-40})));
algorithm
  for i in 1:10 loop // calc volume of each particle
    Vp[i] := 4/3*pi*(c[i]/2)^3;

  end for;

  for i in 1:10 loop

  if th_w>=tv_o[i] then
    Vp_total[i]:= Np[i]*Vp[i];
    n_new[i]:=0;
    last_particle_position[i]:= hw;
  else
    last_particle_position[i]:=vv_o[i]*th_w;
    Vp_total[i]:= Np[i]*(last_particle_position[i]/hw)*Vp[i];
    n_new[i]:=Np[i]*(1-(last_particle_position[i]/hw));
  end if;
  end for;

  Position:=last_particle_position;
```

```
N_particle:=n_new;

q1:=sum(Vp_total)/th_w;

annotation (Icon(coordinateSystem(preserveAspectRatio=false, extent={{-100,
        -100},{100,100}},
    initialScale=0.1), graphics={
    Rectangle(
      extent={{-100,100},{100,-100}},
      lineColor={28,108,200},
      fillColor={85,255,255},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{-8,-32},{12,-52}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{-76,-44},{-56,-64}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{30,18},{50,-2}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{-46,0},{-26,-20}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{58,76},{78,56}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{-4,60},{16,40}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{-72,40},{-52,20}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{30,-64},{50,-84}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{58,-16},{78,-36}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{-48,-72},{-28,-92}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{-12,12},{-4,4}},
      lineColor={28,108,200},
      fillColor={162,29,33},
      fillPattern=FillPattern.Solid),
    Ellipse(
      extent={{-74,-2},{-66,-10}},
      lineColor={28,108,200},
      fillColor={162,29,33},
```

```
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{-54,58},{-46,50}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{38,44},{46,36}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{-26,36},{-18,28}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{26,-28},{34,-36}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{64,-58},{72,-66}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{-10,-78},{-2,-86}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{-32,-34},{-24,-42}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{10,4},{18,-4}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{-2,22},{10,10}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{40,-38},{52,-50}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{-78,74},{-66,62}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{-30,60},{-18,48}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{28,72},{40,60}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
Ellipse(
  extent={{64,4},{76,-8}},
  lineColor={28,108,200},
  fillColor={162,29,33},
  fillPattern=FillPattern.Solid),
```

```
      Ellipse(
        extent={{-70,-24},{-58,-36}},
        lineColor={28,108,200},
        fillColor={162,29,33},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{-16,-10},{-4,-22}},
        lineColor={28,108,200},
        fillColor={162,29,33},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{34,-14},{46,-26}},
        lineColor={28,108,200},
        fillColor={162,29,33},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{-28,-52},{-16,-64}},
        lineColor={28,108,200},
        fillColor={162,29,33},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{10,-64},{22,-76}},
        lineColor={28,108,200},
        fillColor={162,29,33},
        fillPattern=FillPattern.Solid)}),                    Diagram(
      coordinateSystem(preserveAspectRatio=false, extent={{-100,-100},{100,
        100}}),
      initialScale=0.1)));
end particleSepOil1;
```

iii. particleSepWater model

```
model particleSepWater1

Real last_particle_position[10](start=fill(0,10));
Real Vp[10] "volume of each particle";
Real Vp_total[10] "volume of total particles in each class";
Real n_new[10] "new Np";
//parameter Real Np[10]= {1e8, 5e8, 1e9,5e9,1e10,1e10,5e9,1e9,5e8,1e8};// these values should be adjusted after moving to the nex
t stage talk to christoph about this

protected
   final constant Real pi=Modelica.Constants.pi;
  parameter Real u_o=0.001; //oil viscosity kg (s m )^-1
  parameter Real u_w=0.0005;//water viscosity kg (s m )^-1

  parameter Real g= 9.8; // gravity acceleration m s^-2

  parameter Real d_o= 831.5; // oil density kg m^-3
  parameter Real d_w= 1030; // water density kg m^-3
  parameter Real c[10]= {50e-6,100e-6,150e-6, 200e-6,250e-6,300e-6,350e-6,400e-6,450e-6, 500e-
6}; //Particle classes diameter first element smallest, and last largest//
public
  Modelica.Blocks.Interfaces.RealInput ho annotation (Placement(transformation(
        extent={{-20,-20},{20,20}},
      rotation=-90,
      origin={-70,68}),               iconTransformation(extent={{-10,-10},
        {10,10}},
      rotation=-90,
      origin={-40,110})));
  Modelica.Blocks.Interfaces.RealOutput q_w_out_of_oil_1
   "water particles out of oil phase to water" annotation (Placement(
      transformation(
      extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={-34,-2}), iconTransformation(
      extent={{-10,-10},{10,10}},
      rotation=-90,
```

```
      origin={0,-110})));
  Modelica.Blocks.Interfaces.RealInput tv_w[10]
   annotation (Placement(transformation(extent={{-20,-20},{20,20}},
      rotation=-90,
      origin={-32,68}),
      iconTransformation(extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={0,110})));
  Modelica.Blocks.Interfaces.RealInput th_o
   annotation (Placement(transformation(extent={{-20,-20},{20,20}},
      rotation=-90,
      origin={2,70}),
      iconTransformation(extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={40,110})));
  Modelica.Blocks.Interfaces.RealInput vv_w[10] annotation (Placement(
      transformation(extent={{-20,-20},{20,20}},
      rotation=-90,
      origin={36,68}),                          iconTransformation(
        extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={80,110})));
  Modelica.Blocks.Interfaces.RealInput Np[10] annotation (Placement(
      transformation(
      extent={{-20,-20},{20,20}},
      rotation=-90,
      origin={74,68}), iconTransformation(
      extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={-80,110})));
  Modelica.Blocks.Interfaces.RealOutput N_particle[10] annotation (Placement(
      transformation(
      extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={6,-2}),   iconTransformation(
      extent={{-10,-10},{10,10}},
      rotation=0,
      origin={110,40})));
  Modelica.Blocks.Interfaces.RealOutput Position[10] annotation (Placement(
      transformation(
      extent={{-10,-10},{10,10}},
      rotation=-90,
      origin={40,-2}),  iconTransformation(
      extent={{-10,-10},{10,10}},
      rotation=0,
      origin={110,-40})));

algorithm
  for i in 1:10 loop // calc volume of each particle
    Vp[i] := 4/3*pi*(c[i]/2)^3;
  end for;

  for i in 1:10 loop

  if th_o>=tv_w[i] then
    Vp_total[i]:= Np[i]*Vp[i];
    n_new[i]:=0;
    last_particle_position[i]:= ho;
  else
    last_particle_position[i]:=vv_w[i]*th_o;
    Vp_total[i]:= Np[i]*(last_particle_position[i]/ho)*Vp[i];
    n_new[i]:=Np[i]*(1-(last_particle_position[i]/ho));
  end if;
  end for;

  Position:=last_particle_position;
  N_particle:=n_new;

  q_w_out_of_oil_1:=sum(Vp_total)/th_o;

  annotation (Icon(coordinateSystem(preserveAspectRatio=false), graphics={
```

```
      Rectangle(
        extent={{-100,100},{100,-100}},
        lineColor={28,108,200},
        fillColor={162,29,33},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{-34,28},{-14,8}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{0,78},{20,58}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{-80,78},{-60,58}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{-54,-46},{-34,-66}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{-86,-72},{-66,-92}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{20,-70},{40,-90}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{-94,16},{-74,-4}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{42,22},{62,2}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{8,-16},{28,-36}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{46,56},{66,36}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid),
      Ellipse(
        extent={{56,-32},{76,-52}},
        lineColor={28,108,200},
        fillColor={85,255,255},
        fillPattern=FillPattern.Solid)}), Diagram(coordinateSystem(
        preserveAspectRatio=false)));
end particleSepWater1;
```

iv. purityCalc model

```
model purityCalc
  Real Volume[10];
  Real Efficiency;
  Real Vp_total_start[10];
```

```
  Real Vp_total_end[10];

  final constant Real pi=Modelica.Constants.pi;
  parameter Real c[10]= {50e-6,100e-6,150e-6, 200e-6,250e-6,300e-6,350e-6,400e-6,450e-6, 500e-6};

  Modelica.Blocks.Interfaces.RealInput N_start[10] "from first segment "
    annotation (Placement(transformation(extent={{-140,20},{-100,60}}),
        iconTransformation(extent={{-140,20},{-100,60}})));
  Modelica.Blocks.Interfaces.RealInput N_end[10] "from last segment "
    annotation (Placement(transformation(extent={{-140,-60},{-100,-20}}),
        iconTransformation(extent={{-140,-60},{-100,-20}})));

equation
  for i in 1:10 loop // calc volume of each particle
    Volume[i] =  4/3*pi*(c[i]/2)^3;
    Vp_total_start[i]=  N_start[i]*Volume[i];
    Vp_total_end[i]=  N_end[i]*Volume[i];
  end for;

  Efficiency=1-(sum(Vp_total_end)/sum(Vp_total_start));
  annotation (Icon(coordinateSystem(preserveAspectRatio=false), graphics={
        Rectangle(
          extent={{-100,100},{100,-100}},
          fillColor={40,40,40},
          fillPattern=FillPattern.Solid,
          pattern=LinePattern.None),
        Rectangle(
          extent={{-100,92},{100,-100}},
          fillColor={61,61,61},
          fillPattern=FillPattern.Solid,
          pattern=LinePattern.None),
        Rectangle(
          extent={{-100,82},{100,-100}},
          fillColor={86,86,86},
          fillPattern=FillPattern.Solid,
          pattern=LinePattern.None),
        Rectangle(
          extent={{-100,60},{100,-100}},
          fillColor={107,107,107},
          fillPattern=FillPattern.Solid,
          pattern=LinePattern.None),
        Rectangle(
          extent={{-100,20},{100,-100}},
          fillColor={118,118,118},
          fillPattern=FillPattern.Solid,
          pattern=LinePattern.None),
        Rectangle(
          extent={{-100,-18},{100,-100}},
          fillColor={135,135,135},
          fillPattern=FillPattern.Solid,
          pattern=LinePattern.None),
        Rectangle(
          extent={{-100,-52},{100,-100}},
          fillColor={149,149,149},
          fillPattern=FillPattern.Solid,
          pattern=LinePattern.None),
        Rectangle(
          extent={{-100,-74},{100,-100}},
          fillColor={165,165,165},
          fillPattern=FillPattern.Solid,
          pattern=LinePattern.None)}),                    Diagram(
      coordinateSystem(preserveAspectRatio=false)));
end purityCalc;
```