

Fortrengning av gass med en væskestrøm: Småskala forsøk

Andreas Navjord Winnem

Master i produktutvikling og produksjon
Oppgaven levert: Juni 2009
Hovedveileder: Ole Jørgen Nydal, EPT

Oppgavetekst

Hovedoppgaven under arbeidet er å utvikle et småskala system for å undersøke problematikken rundt fortregning av et luft fylt, bølgeformet rør med en vannstrøm inn i røret som har konstant trykk. De eksperimentelle dataene vil bli sammenlignet med beregninger i strømningssimulatoren, OLGA, og muligens andre modeller.

Oppgaven gitt: 27. januar 2009

Hovedveileder: Ole Jørgen Nydal, EPT



MASTER THESIS

for

Stud. techn. Andreas N Winnem

Spring 2009

Norsk tekst

Fortrengning av gass med en væskestrøm: småskala forsøk

Engelsk tekst

Liquid flushing of a pipeline: small scale experiments

Background

A special transient two phase flow case is the flushing of a pipeline with a liquid stream. When pipelines are pressure tested with water, no residual gas can remain in the pipeline. If pumping of water behind a pig is not possible, then flushing of the line may be attempted under the natural flow of water after opening the pipeline to water inflow subsea. This is the background for a MSc thesis on experiments and associated numerical simulations.

Objective

The main objective of the work is to develop a small scale experimental system to investigate the problem of flushing an air filled, undulating pipe with an inlet water stream at constant pressure. The experimental data will be compared with predictions using the flow simulator OLGA, and possibly with other models.

Experiments

A small scale pipeline with a W shaped setup (in the range of 4 meters in length, 2 cm ID) will be connected to a water reservoir at constant height. An inlet valve will be fully opened and the setup will be video recorded as the water flushes into the pipe. The main variable in the experiments will be the height of water in the reservoir. At sufficiently high levels the high velocity water will displace all the air, whereas residual air will remain in the pipe at lower water heights, which gives lower flow velocities.

Simulations

The experiments will be simulated using OLGA. A critical phenomenon for the correct flow simulations is the bubble turning process in the downwards inclined pipes.

Tasks

The following tasks are foreseen:

1. Prepare a small scale, simple and schematic experimental set-up in the laboratory.
2. Arrange for one or more cameras to track the flow evolution in time.
3. Evaluate the use of digital image processing in Matlab for the determination of the liquid-air front propagations.
4. Perform OLGA simulations and compare with data
5. Report

Work place and supervision

The MSc work will be made in the Multiphase Flow Laboratory at EPT, NTNU. The candidate will be in contact with a research group on flow assurance at FMC, Asker (contact person Randi Moe).

Within 14 days of receiving the written text on the diploma thesis, the candidate shall submit a research plan for his project to the department.

When the thesis is evaluated, emphasis is put on processing of the results, and that they are presented in tabular and/or graphic form in a clear manner, and that they are analyzed carefully.

The thesis should be formulated as a research report with summary both in English and Norwegian, conclusion, literature references, table of contents etc. During the preparation of the text, the candidate should make an effort to complete a well presented report. In order to ease the evaluation of the thesis, it is important that the cross references are correct. In the making of the report, strong emphasis should be placed on both a thorough discussion of the results and an orderly presentation.

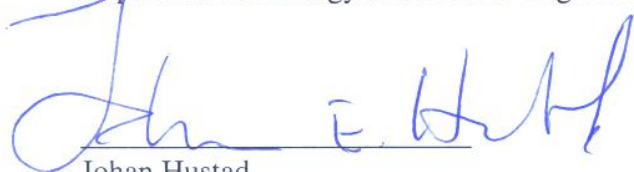
The candidate is requested to initiate and keep close contact with his/her specialist teacher and academic supervisor(s) throughout the working period. The candidate must follow the rules and regulations of NTNU as well as passive directions given by the Department of Energy and Process Engineering.

Pursuant to “Regulations concerning the supplementary provisions to the technology study program/Master of Science” at NTNU §20, the Department reserves the permission to utilize all the results for teaching and research purposes as well as in future publications.

One – 1 complete original of the thesis shall be submitted to the authority that handed out the set subject. (A short summary including the author’s name and the title of the thesis should also be submitted, for use as reference in journals (max. 1 page with double spacing)).

Two – 2 – copies of the thesis shall be submitted to the Department. Upon request, additional copies shall be submitted directly to research advisors/companies. A CD-ROOM (Word format or corresponding) containing the thesis, and including the short summary, must also be submitted to the Department of Energy and Process Engineering

Department of Energy and Process Engineering, 12. January 2009



Johan Hustad
Department Manager



Ole Jørgen Nydal
Academic Supervisor

Research Advisors and external supervisors:

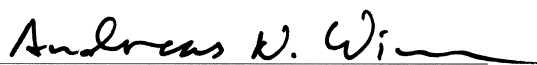
Randi Moe(FMC)

Preface

This Master Thesis is written at the Department of Energy and Process Engineering at the Norwegian University of Science and Technology (NTNU) in Trondheim, spring 2009.

The Thesis has included laboratory work, simulation and programming. In all, this has been a quite demanding and rewarding work. A variety of different practical and theoretical problems have been encountered. With the guidance of excellent lab engineers, PhD candidates and my supervisor, Professor Ole Jørgen Nydal, I have been able to overcome the challenges I have encountered.

I would like to credit Erling Mikkelsen for his contribution to setting up the experimental rig. PhD candidates Gandi Rahmawan Setyadi for guidance with the peculiarities of OLGA 6.0 and PhD candidate Angela De Lèebèeck for vital feedback. Finally, I would like to credit Professor Ole Jørgen Nydal for formulating a Thesis that was very demanding but manageable with my academic background.



Andreas N Winnem, NTNU Trondheim

Abstract

Liquid flushing is the displacement of gas with a flowing liquid column and is important in pressure testing of pipelines. To evaluate the capability of the multiphase simulator OLGA 6.0 to predict flushing of a pipeline, simulations in OLGA have been compared to small-scale experiments.

A test rig has been set up with the configuration of an undulating pipeline. The main variable was the height of the water in the reservoir. The experiments were video recorded. The end state of the flow was logged by measuring the height of the liquid column in the different pipe sections. This was compared with the end state in OLGA simulations. A Matlab script was developed to perform image analysis of the video. The image analysis script was used to compare the transient development of the experiments with the simulations in OLGA.

The end state in the cases where the pipe was not flushed in the experiments was in good correspondence with the OLGA simulations. The transient progress was however much faster in OLGA. The ratio in time for the water to reach the outlet in the experiment where the pipe was flushed and the OLGA simulation was 2.5. This ratio declined with the inlet pressure. The reason for this discrepancy is thought to be an effect of that there is no model for surface tension between the fluid and the wall in OLGA. In order to find a minimum range for the head needed in OLGA to flush the pipe, a parametric study was carried out. The factor between the head needed to flush the pipe in the experiments and the head predicted in OLGA was 0.84. This was surprising since OLGA predicted a much quicker transient progress with higher velocities and momentum. The reason for the over prediction of the head needed to flush the pipe is thought to come from that OLGA to a small extent takes into account the flow history. The effect of this is that slugs are killed at the end of an upwards or downwards pipe.

Sammenheng

Spyling (Flushing) av gass med en væske kolonne er viktig i forbindelse med trykk testing av rørledninger. For å vurdere multifase simulatoren OLGA 6.0 sin evne til å predikere spyling av en rørledning har simuleringer i OLGA blitt sammenlignet med små skala forsøk.

En test rigg har blitt satt opp med konfigurasjonen av en bølgeformet rørledning. Den viktigste variabelen var høyden på vannet i reservoaret. Forsøkene ble filmet med et video kamera. Slutt tilstanden ble logget ved å måle den vertikale høyden av væske kolonnene i de ulike rør seksjonene. Dette ble sammenlignet med slutt tilstanden i OLGA simuleringene. Et Matlab skripte ble utviklet for å gjøre bilde analyse av filmen. Bilde analysen ble brukt til å sammenligne det transiente forløpet av eksperimentene med simuleringene i OLGA.

Slutt tilstanden i forsøkene hvor røret ikke ble spylt var i god overensstemmelse med simuleringene i OLGA. Det transiente forløpet var mye raskere i OLGA. Forholdet mellom tiden det tok væsken å nå utløpet i eksperimentet hvor røret ble spylt og simuleringen i OLGA var 2.5. Dette forholdet avtok med innløpstrykket. Grunnen til denne uoverensstemmelsen er vurdert å komme av at det ikke er noen modell for overflatespenning mellom fluid og vegg i OLGA. For å finne minste løftehøyde for at OLGA skulle predikere spyling av røret, ble en parameterstudie av innløpstrykket utført. Faktoren mellom løftehøyden som var nødvendig for å spyle røret i eksperimentene og OLGA simuleringen var 0.84. Dette var overraskende siden OLGA predikerte et mye raskere transient forløp med større hastighet og bevegelsesmengde. Grunnen til over prediksjonen av den nødvendige løftehøyden antas å komme av at OLGA til en liten grad tar høyde for strømnings historikk. Effekten av dette er at væskepluggen forsvinner i overgangen mellom et oppover rør og et nedover rør.

Nomenclature

Parameters

α	Volume fraction [-]
μ	Dynamic viscosity [kg/ms]
ρ	Density [kg/m^3]
A	Cross sectional area [m^2]
c	Celerity [m/s]
C_d	Valve discharge coefficient [-]
C_v	Valve sizing coefficient [$gal/min/psi^2$]
D	Diameter [m]
ϵ	Roughness [m]
f	Friction factor [-]
g	Gravity [m/s^2]
h	Head [m]
H	Holdup [-]
Fr	Froude number [-]
p	Pressure [Pa]
Q	Volumetric flux (flow) [m^3/s]
Re	Reynolds number [-]
t	Time [s]
θ	Angle [-]
u	Velocity [m/s]
W_b	Bubble propagation rate [-]
z	Elevation [m]
σ	Surface tension [N/m]
QLT	Total liquid volume flow (OLGA) [m^3/s]
UL	Liquid velocity (OLGA) [m/s]
HOL	Holdup (OLGA) [-]
W	Mass flux (OLGA) [kg/s]

Abbreviations and annotations

OLGA	A multiphase, transient flow simulator.
PVT	Pressure, specific Volume and Temperature
Steady state	System with constant properties in time
Transient	A process that changes with time
Three-fluid model	A two-phase flow model in which the momentum equation is solved for each phase. The phases are often a mix of fluids e.g. oil and gas in the liquid phase.
Holdup	Liquid fraction of a volume
Water fraction	Water fraction of a volume
Gas fraction	Gas fraction of a volume

Contents

Assignment	II
Preface	II
Abstract	III
Sammendrag (Abstract in Norwegian)	V
Nomenclature	VII
Abbreviations and annotations	IX
Contents	XIII
List of Figures	XV
List of Tables	XVII
1 Introduction	1
2 Theory	3
2.1 Flow regimes	3
2.2 Flow regimes in the undulating pipe	4
2.2.1 First uphill pipe	4
2.2.2 Bubble turning and stratification of the flow	4
2.2.3 Criteria for stratification of the flow	5
2.2.4 Slug flow	8
2.3 Pressure drop in two limiting cases	9
2.3.1 Filled pipe	9
2.3.2 Stratified downhill flow	11
2.3.3 Solution of the cases	12
2.4 Valve coefficient	12
2.5 Multiphase flow simulator OLGA	13

3	Experimental setup	15
3.1	Modifications	15
3.2	Experimental procedure	16
3.2.1	Piping and video	16
3.3	Geometric relations	19
4	OLGA Case description	21
4.1	Overview	21
4.2	The configuration	22
4.3	Initial and boundary conditions	23
4.3.1	Feed pipe	23
4.3.2	Inlet source	24
4.3.3	Initial pressure	24
4.3.4	Pressure drop	24
4.4	PVT tables	25
4.4.1	PVTsim	25
4.4.2	Interpolation	26
4.5	Numerical issues	26
4.5.1	CFL condition	26
4.5.2	Stability vs resolution	27
4.5.3	Stable model	27
4.6	Other issues	28
4.6.1	Slugs	28
4.6.2	Valve	28
4.7	Simplifications and limitations	28
5	Post processing	31
5.1	Matlab	31
5.2	Image analysis	31
5.2.1	Overview	31
5.2.2	Importing video files to MatLab	32
5.2.3	Investigation of pixels	32
5.2.4	Color filter	32
5.2.5	Search algorithms	35
5.2.6	Output	36
5.2.7	Future improvements	37
5.3	Post processing of output from video analysis	37
5.4	Post processing of OLGA simulations	38
5.5	Comparison of the results	39
6	Results	41
6.1	A few notes about the plots	41
6.2	Comparison of OLGA and experimental results	41
6.3	Discretized bends vs sharp bends	47

6.4	Coarse vs fine grid	48
6.5	Parametric study of C_d	49
6.6	Study of velocity profile in OLGA	49
7	Discussion	53
7.1	Comparison of OLGA and experimental results	53
7.1.1	End state	53
7.1.2	Transient progress	54
7.1.3	Flushing	57
7.2	Discretized bends vs sharp bends	58
7.3	Coarse vs fine grid	59
7.4	Parametric study of C_d	59
7.5	Study of velocity profile in OLGA	60
8	Conclusions	63
	Bibliography	67
A	Matlab script for the marginal cases	69
B	Matlab script for discretization of the pipe	73
C	Matlab script for video analysis	77
D	Matlab script for post processing of the video analysis	91
E	Matlab script for post processing of the OLGA output	95
F	Matlab script for comparing the video analysis and the OLGA output	103

List of Figures

2.1	Flow regimes in a horisontal and vertical pipe	4
2.2	Flow phenomena in the undulating pipe	8
2.3	Development of terrain slugs in the pipe	10
2.4	Marginal cases	12
3.1	The original rig and modifications	16
3.2	Schematics of the rig	18
3.3	Geometric relations for the bends	19
4.1	Sharp and sectioned bends	22
4.2	Full and simplified model	25
5.1	Matlab flowcharts for video analysis	33
5.2	Investigation of the pixels in filled and empty sections	34
6.1	Experimental and simulation data for water surface 0.450 m, PVT AirWater2	43
6.2	Experimental and simulation data for water surface 0.450 m, PVT AirWater4	44
6.3	Experimental and simulation data for water surface 0.675 m .	45
6.4	Experimental and simulation data for water surface 0.750 m .	45
6.5	Experimental and simulation data for water surface 0.825 m .	46
6.6	Parametric study of the inlet pressure vs flushing. Y-axes: dark; vertical height, red; holdup, blue; total volumetric flow .	46
6.7	Comparison of sharp and discretized bends	47
6.8	Comparison of coarse and fine grid	48
6.9	Parametric study C_d , 0.04 seconds. Y-axes; dark; height, pink; holdup	49
6.10	Parametric study C_d , 20 seconds. Y-axes; dark; height, pink; holdup	50
6.11	Profile after 0.04 seconds	50
6.12	Profile after 10 seconds	51

List of Tables

3.1	Straight sections	17
3.2	Bends	17
3.3	Valve specifications	17
3.4	Colorant	18
3.5	Video camera	18
3.6	Discharge pipe	18
6.1	Column height at end state	42

Chapter 1

Introduction

Liquid flushing of pipelines is the displacement of gas with a flowing liquid column. The purpose is to completely empty the pipeline of gas. In order to do pressure testing on pipelines no residual gas can remain in the pipeline and thus it is critical to know that the pipeline is actually free of gas. This can be investigated by pressure - volume correlations applying pressure from one side of the pipe and measuring the compressibility of the fluid in the pipeline. If the compressibility is higher than the fluid assumed present in the pipe, one can suspect that gas is present in the pipeline.

However, trying to flush pipelines by trial and error is not an efficient way to operate. Instead simulations should be performed in order to achieve flushing with a minimum of driving pressure. Using as low an inlet pressure as possible is important to safety and reduces costs. Using an excessive pressure results in a higher liquid velocity than what is necessary. This causes wear on the equipment due to corrosion and acceleration of liquid slugs that may result in damage to the equipment and injury to personnel.

OLGA is a multiphase simulator that could be used to predict the pressure needed to flush a pipe. In order to investigate OLGA's capability to do such simulations a small scale experimental set-up have been developed. In the experiments an undulating, air filled pipeline is filled with an inlet water stream at constant pressure. The end state of a partially filled pipe is logged by measuring the height of the water column in each pipe segment. The experiments are filmed with a video camera. A Matlab script has been developed to investigate some transient parameters. The end state in partially filled pipes and the transient development of the filling process has been compared with simulations of the experiments in OLGA. Through these investigations OLGA's performance in reproducing the laboratory experiments has been evaluated.

Chapter 2

Theory

In this section concepts regarding multiphase flow in pipelines and characteristic equations for the problem at hand. This is to give the reader a general understanding of the flow and a reference of equations especially developed for phenomena that evolves in the experiments and simulations. This background is given as a reference for future work on this topic. A more thorough recapitulation of multiphase theory may be found in lecture notes by Prof. Ole Jørgen Nydal in [6].

2.1 Flow regimes

In multiphase flow (2 or more phases) a variety of flow regimes may occur. A flow regime describes the configuration of the phases, i.e. the shape and mixing of the phases. Multiphase flow is a term for a large variety of flow phenomena with very different physical mechanisms involved. Since the phenomena are so different, multiphase flow is studied by investigating the characteristics of each flow regime. An overview of flow regimes in a horizontal and vertical pipe is found in figure 2.1.

In the experiments the pipe is constructed of straight sections with an upward or downward angle of about 40 degrees. The flow regimes of a horizontal pipe (figure 2.1a) will tend to occur in the downward sections and the regimes of a vertical pipe (figure 2.1b) will tend to occur in the upward sections. Also, not all of the regimes will be present since the experimental set-up is a low pressure system with low velocities.

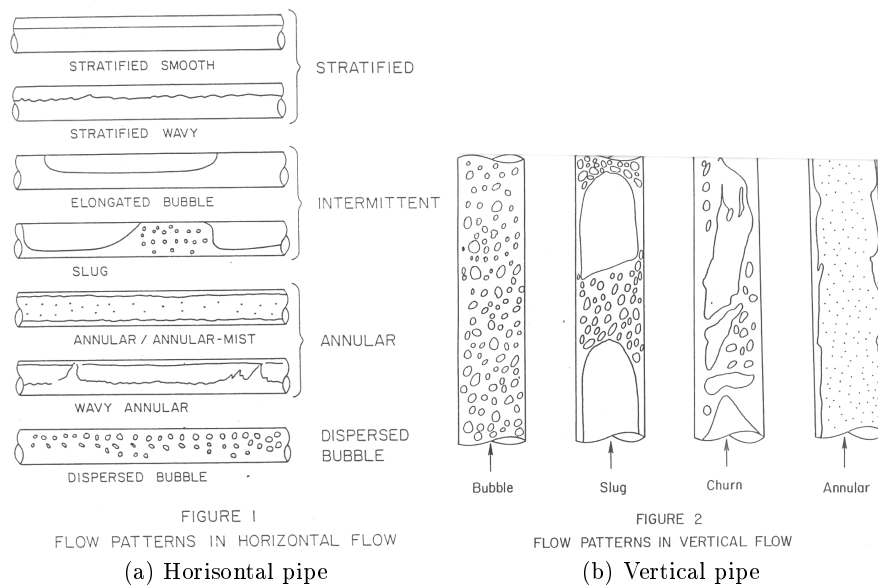


Figure 2.1: Flow regimes in a horizontal and vertical pipe

2.2 Flow regimes in the undulating pipe

In this section, theory for the experiments is presented and some criteria to verify the results are given. The flow regimes observed in the experimental set-up is found in figure 2.2.

2.2.1 First uphill pipe

The observed flow phenomena in the experiments are single phase flow of liquid in the first uphill pipe and then stratified flow and a mixture of stratified, slug and bubble flow in the subsequent sections. Since the source (single phase water) is at the bottom of the first uphill section it is expected that the water column is continuous as long as the pipe is inclined upwards see figure 2.2a. This is because the pressure is higher in the source than in the pipe and that gravity is counter current, pulling the dense water phase towards the ground.

2.2.2 Bubble turning and stratification of the flow

In the downwards sections the flow is stratified or elongated bubble flow. When the waterfront passes a peak, gravity is co-current. This is a mechanism that tends to stratify a flow as long as the pipe is inclined at an angle between 0 and 45 degrees downwards. According to Zukoski (1966) (see [2])

stratification of a flow is most likely to occur at an angle of 45 degrees downwards. In a vertical pipe liquid will flow down on all sides of a bubble in an axis symmetrical fashion. This is because the gravitational field is parallel to the flow as opposed to all other configurations, where gravity is pulling the dense phase towards the bottom of the pipe. Stratified flow in the pipe is illustrated in figure 2.2b and figure 2.2d. In figure 2.2b the velocity of the water phase is too slow to maintain a front as it passes the peak. In figure 2.2d the stratified flow in the downhill pipe 4 is blocked by a water lock in the bend. The discharge rate from the tank is almost zero. In this case water is flowing as a thin film underneath the air bubble that is propagating up towards the peak. In the former case, the head pressure drives the water front down towards the bend. In the latter, the head pressure is too low to push the bubble through the bend.

2.2.3 Criteria for stratification of the flow

According to Liou and Hunt [2] air intrusion occurs when the celerity, c of a long air cavity exceeds the discharge velocity in the filled portion of a pipe. Liou and Hunt have developed a method for calculating the celerity by use of Zukovski's data and method for finding the bubble propagation rate [11]. Bubble propagation rate has the same meaning as bubble rise (velocity) in a stagnant column. Zukoski investigated the dependency of the celerity on viscosity, surface tension, the slope of the pipe and the diameter. The findings were that viscosity is not significant, surface tension becomes more pronounced for smaller pipe diameters and the celerity increases by approximately 20 % as the downward angle increases from zero to 45 degrees. The celerity decreases upon a further increase in the angle. Liou and Hunt have proposed one correlation for the critical celerity for a horizontal pipe with diameters between 4 and 18 cm:

$$c_{critical} = 0.5\sqrt{gD} \quad (2.1)$$

Air intrusion (bubble turning) should not occur if the discharge velocity exceeds $c_{critical}$. Townson [8] found that the parameter $\frac{c}{\sqrt{gD}}$ reaches a maximum of 0.57, slightly higher than in equation 2.1. This factor is appropriate for larger pipes. For smaller diameters Liou and Hunt has suggested using Zukoski's data to find $\frac{c}{\sqrt{gD}}$ and multiplying with a factor between 1 and 1.2 to account for inclination to find the celerity of a pipe:

$$c_{critical} = C_{Zukoski}C_{angle}\sqrt{gD} \quad (2.2)$$

where the parameter C_{angle} is between 1 and 1.2. In the set-up used in

the configuration with angles at approximately 40 degrees it is appropriate to set $C_{angle} = 1.2$. To find the parameter $C_{Zukoski}$ one must read off the number corresponding to the pipe diameter for a horizontal pipe ($\theta = 0$) in figure 5 in Zukoski's paper [11]. This is a dimensionless bubble propagation rate, $\frac{W_b}{\sqrt{\frac{\Delta\rho}{\rho}ga}}$. W_b is the bubble propagation rate, $\Delta\rho$ is the density difference between water and air, ρ the density of water, g the gravitational acceleration and a the radius of the pipe. Now the celerity c is calculated as

$$c = C_{chart} \frac{W_b}{\sqrt{\frac{\Delta\rho}{\rho}ga}} \quad (2.3)$$

where C_{chart} is the read-off from figure 5 in Zukoski's paper [11]. Finally the factor $\frac{c}{\sqrt{gD}}$ is found as

$$\frac{c}{\sqrt{gD}} = \frac{W_b}{\sqrt{g2a}} = C_{Zukoski} \quad (2.4)$$

The factor $\frac{c}{\sqrt{gD}}$ may now be cross checked with the values found for a 17.8cm and a 1.36cm pipe in [2]. The pipe in Zukoski's paper with diameter 2.16cm is fairly close to the pipe in the experimental set-up. The critical celerity of the experimental set-up in this report is then

$$c_{critical} = 0.35 \cdot 1.2 \sqrt{9.81 \cdot 0.02} \approx 0.20m/s \quad (2.5)$$

Nydal [5] took a different approach to finding a criteria for stratification of a flow by air intrusion. He investigated at what conditions a co-current (downstream) bubble (air intrusion) would change direction and flow counter current (upstream). This phenomenon is known as bubble turning since the bubble changes the direction in which it flows. The assumption is that the bubble will flow towards the low pressure side. Starting off by setting up the steady state liquid momentum balance (neglecting acceleration)

$$-\frac{\partial p}{\partial x} = \frac{1}{D} \tau_{L,W} - \rho g \sin \theta \quad (2.6)$$

where $-\frac{\partial p}{\partial x}$ is the pressure drop along the pipe, D is the pipe diameter, $\tau_{L,W} = \frac{1}{2} \lambda \rho_L u_L^2$ is the liquid/wall shear stress, u_L the liquid phase velocity and λ the Darcy friction factor. The ratio F between the frictional and gravitational forces is given by

$$F = \frac{8\lambda F r^2}{\sin \theta} \quad (2.7)$$

where the Froude number $Fr = \frac{u_L}{\sqrt{gD}}$. The flow is critical when $F = 1$. That is when the gravitational forces balance the frictional forces. For a factor F smaller than 1 bubble turning will occur. The criterion is

$$u_{turning} = u_{critical} + u_0 \quad (2.8)$$

where $u_{turning}$ is the liquid velocity at which the bubble turns. u_0 is the rise velocity of a bubble in stagnant fluid. The equations 2.6, 2.7 are found in Johansens doctoral thesis [3]. Nydal set up the friction-gravity balance as

$$\frac{1}{2}f\rho U_l^2 S = \rho g A \sin \theta \quad (2.9)$$

where f is the fanning friction factor, S the pipe perimeter and A the pipe cross section area. Blasius friction factor for turbulent flow was applied

$$f = 0.046Re^{-0.2} \quad (2.10)$$

with the Reynolds number $Re = U_l \frac{D}{\nu}$ where ν is the kinematic viscosity. The ratio between friction and gravity forces was written as

$$R_c = \frac{2fF^2}{\sin \theta} \quad (2.11)$$

Equation 2.11 is equivalent with 2.7 and the same criterion of the critical flow velocity, equation 2.8 was applied. Nydal [5] and Johansen and Nydal [4] proposed the use of two different equations for calculating u_0 in equation 2.8. Nydal suggested the use of equation 2.12 (Bendiksen [1]) that is simpler, but does not take into account surface tension as this equation is for large pipe diameters typically for oil and gas pipelines:

$$u_0 = 0.54\sqrt{gD} \cos \theta + 0.35\sqrt{gD} \sin \theta \quad (2.12)$$

Johansen and Nydal [4] suggested the use of equation 2.13 for the rise velocity of a bubble in stagnant liquid is taking into account surface tension:

$$u_0 = \left[0.54 - \frac{1.76}{E_o^{0.56}} \right] \sqrt{gD} \cos \theta + 0.35\sqrt{gD} \sin \theta \quad (2.13)$$

where $E_o = \frac{\rho L g D^2}{\sigma}$ is the Eötvös number and σ the surface tension. Johansen [3] measured the surface tension for filtered tap water to $\sigma = 0.075 N/m$. The discrepancy between equation 2.12 and 2.13 is the term $-\frac{1.76}{E_o^{0.56}} \sqrt{gD} \cos \theta$ in

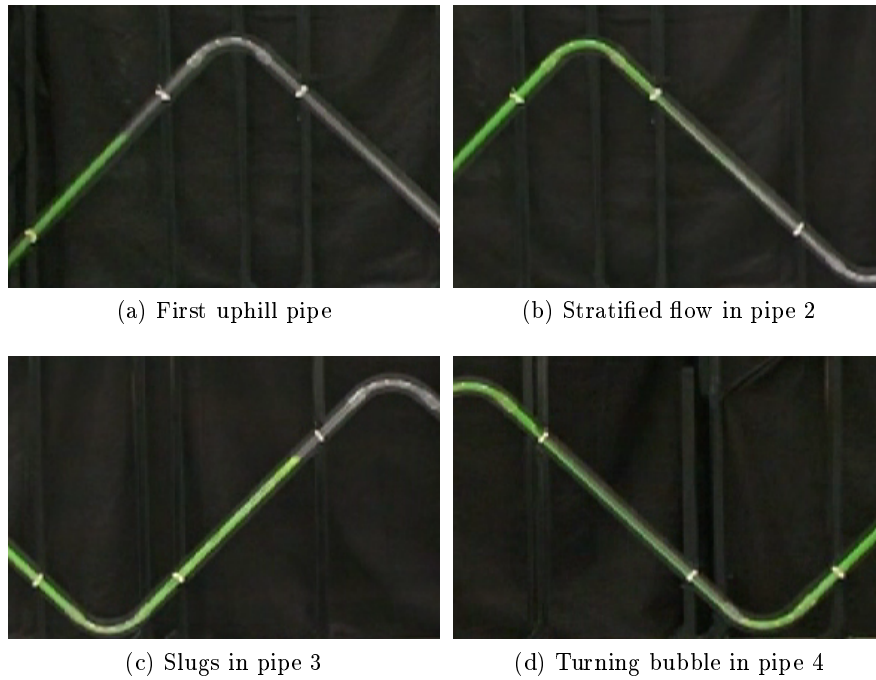


Figure 2.2: Flow phenomena in the undulating pipe

equation 2.13 that takes into account the surface tension and reduces the rise velocity.

Measurements of $u_{turning}$, $u_{critical}$ and u_0 for a pipe with $D = 1.92cm$ is found in Nydals paper [5]. The pipe diameter is very close to the diameter used in the experiments in this report and Nydals results may be used at least as a guideline with regards to these parameters. In Johansens results [3], three phase flow with $D = 3.2cm$ has been used. This is probably too different from the set-up in this report to be relevant.

2.2.4 Slug flow

Slug flow occurs when the superficial velocities of the liquid and gas phase are relatively low. This is illustrated in flow regime maps with the superficial velocities on the axes e.g. flow regime maps found in [6]. In order to achieve slug flow a phase with high compressibility (air) and a phase with low compressibility (water) must be present. In order to develop slugs a liquid blocking of the flow is needed, typically at a low point. The upstream pressure compresses the gas phase until the gas pressure equals the downstream column. Then gas will flow past the liquid block until the pressure upstream in the gas phase is lower than the pressure of the downstream column. This phenomenon occur in the pipe as the stratified water phase

flows down the downhill pipes and blocks the bends. However, at the point when water is blocking a bend, the height of the downstream water column is very small. A mix of air and water flow past the bend before the water lock increases and slugging becomes more prominent. In figure 2.2c the liquid lock is significant. One can see that there are slugs in the uphill pipe where the green color is lighter indicating presence of air. When the flow reaches a peak, gravity pulls the water in the slugs down towards the bottom of the pipe and the slugs disintegrate. A water column with slugs is lighter than a water column with single phase water. If slugging starts to occur, the slugs may accelerate since the water column becomes increasingly lighter and the required back pressure in the gas to flow through the bend and cause gas declines. In the experiments, the slugs died as the gas bubbles displaced through the water front and into the air filled part of the pipe. The slugs made the liquid column in the upwards inclined pipes lighter (not the first since there was single phase flow from the source). Then it is possible for the water to propagate through the pipe, with stratified downhill flow and no pressure recovery with a back pressure equivalent to less than the sum of the vertical height of the upward sections.

Mandal, Bhuyan, Das and Das (Mandal et al.) [7] carried out experiments of two-phase flow through a undulating pipe. The observations in the experiments committed during this thesis supports qualitatively the results from their work. Mandal et al. used measurement techniques to quantify the phenomena (slugging, stratification) during their work. In this thesis the rig has not been set up in this way as the scope is not to perform this kind of research but to compare some parameters from the experiments with OLGA.

2.3 Pressure drop in two limiting cases

2.3.1 Filled pipe

There are two limiting cases in which the pressure drop may be calculated analytically. The simplest case is single phase stationary flow. That is the case when the pipe is filled with water and all air has been displaced. Applying Bernoulli's equation with minor losses along a streamline in stationary flow [10] page 385:

$$-\Delta p = \rho g [\Delta z + \Delta h_t] \quad (2.14)$$

where the total system loss is

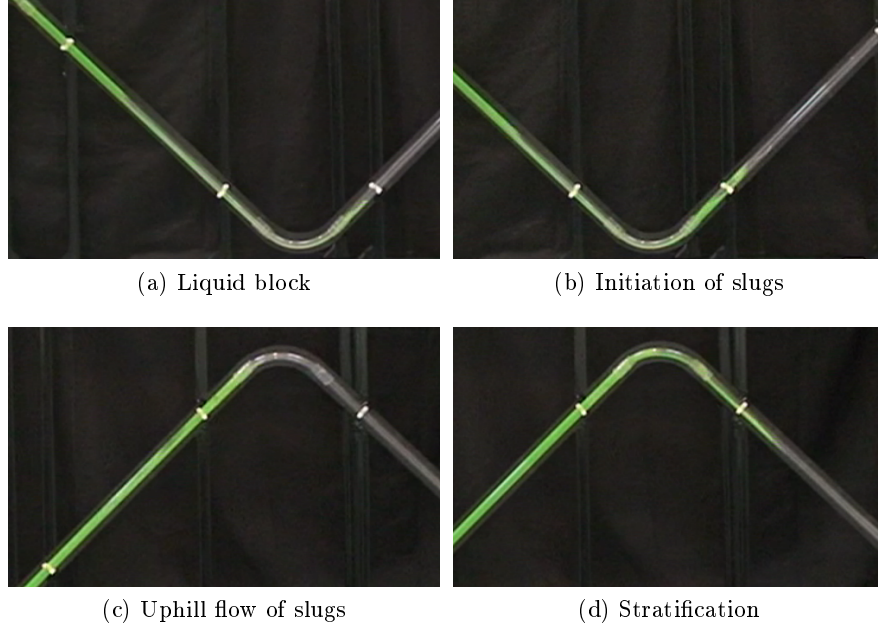


Figure 2.3: Development of terrain slugs in the pipe

$$\Delta h_t = \frac{u^2}{2g} \left(\frac{fL}{D} + \sum K \right) \quad (2.15)$$

Δp is the head loss, Δz is the difference in elevation, f is the friction factor and K denote minor losses. The friction factor f is dependent on the flow regime, if it is laminar, turbulent or transitional. In these rough estimations friction factor for Poiseuille flow is used for laminar flow; $f = f(Re) = \frac{64\mu}{\rho u D} = \frac{64}{Re_d}$. For turbulent flow, the explicit Haaland friction factor is used; $f = f(Re) = \left(-1.8 \log \left[\frac{6.9}{Re_d} + \left(\frac{\epsilon}{3.7D} \right)^{1.11} \right] \right)^{-2}$. The loss in the elbows (bends) has been approximated as the loss for a 1 inch pipe with a 90 degree long radius, flanged elbow in [10], page 387, $K_{elbow,90} = 0.40$. The last elbow is half the angle of the previous ones and the head loss is according to the mentioned table $K_{elbow,45} = 0.21$. The valve factor has been found from the same table, fully open, screwed globe valve with a 2 inch diameter, $K_{valve} = 6.9$. The pipe diameter in the feed pipe is 4cm and the diameter in the test section is 2 cm. These are connected by overlapping the feed pipe onto the the thinner pipe, resulting in a sudden contraction of the flow in the coupling. This is also known as vena contracta [10] page 389. The loss due to vena contracta is $K_{SC} \approx 0.42 \left(1 - \frac{d^2}{D^2} \right)$ where d is the diameter of the thin pipe and D the diameter of the thick pipe. In the rig $K_{SC} = 0.105$. Head loss from a sharp inlet (the tank) is set to $K_{inlet} = 0.5$. A sharp inlet

has the highest head loss. Since it is not known what the loss is at the inlet, using a sharp inlet is deemed conservative in the sense that a higher pressure loss will be calculated. The outlet is deemed where the vent is mounted, reducing the pressure in the fluid to atmospheric pressure (1bara). The head loss coefficient is then $K_{outlet} = 1$ since the velocity of the fluid is (approximately) zero at the surface in the tank. Instead of using velocities, the volume flow, Q is used. This is due to that there are two different pipe diameters. The velocity in a pipe section is found by dividing with the area. The total pressure drop in a filled pipeline is then according to equation 2.14 and 2.15

$$\begin{aligned}
-\Delta p = \rho g \left[\Delta z + \frac{Q^2}{2g} \left(\left[\frac{fL}{D} \right]_{feed} + K_{inlet} + K_{valve} \right) \frac{1}{A_{feed}^2} \right. \\
\left. + \left[\frac{fL}{D} \right]_{pipe} + 4K_{elbow,90} + K_{elbow,45} + K_{SC} + K_{outlet} \right] \frac{1}{A_{pipe}^2} \quad (2.16)
\end{aligned}$$

2.3.2 Stratified downhill flow

The second case is stratified flow through all the downhill sections and single phase in the uphill sections. Then the pressure drop in the pipeline will be the sum of pressure drop due to vertical height difference in the uphill sections plus the friction force and the head loss in the feed pipe. That is, there is no recovery of pressure in the downhill sections due to gravity because the flow is stratified. The friction force in the downhill sections and the gravitational force cancel out since they are equal in a stratified, stationary flow. The same equations as for the filled pipe may be applied with some modifications. The pressure drop in the downhill pipes due to friction is canceled out by gravitational force as argued. The 90 degree bends are assumed to have an effect of a 45 degree bend as stratification is assumed at the top point of an uphill pipe. Single phase flow is assumed to occur analogously in the bottom of a valley, with the effect of a 45 degree bend. Equation 2.16 is now

$$\begin{aligned}
-\Delta p = \rho g \left[\sum z_{uphill} + \frac{Q^2}{2g} \left(\left[\frac{fL}{D} \right]_{feed} + K_{inlet} + K_{valve} \right) \frac{1}{A_{feed}^2} \right. \\
\left. + \left[\frac{fL_{uphill}}{D} \right]_{pipe} + 5K_{elbow,45} + K_{SC} + K_{outlet} \right] \frac{1}{A_{pipe}^2} \quad (2.17)
\end{aligned}$$

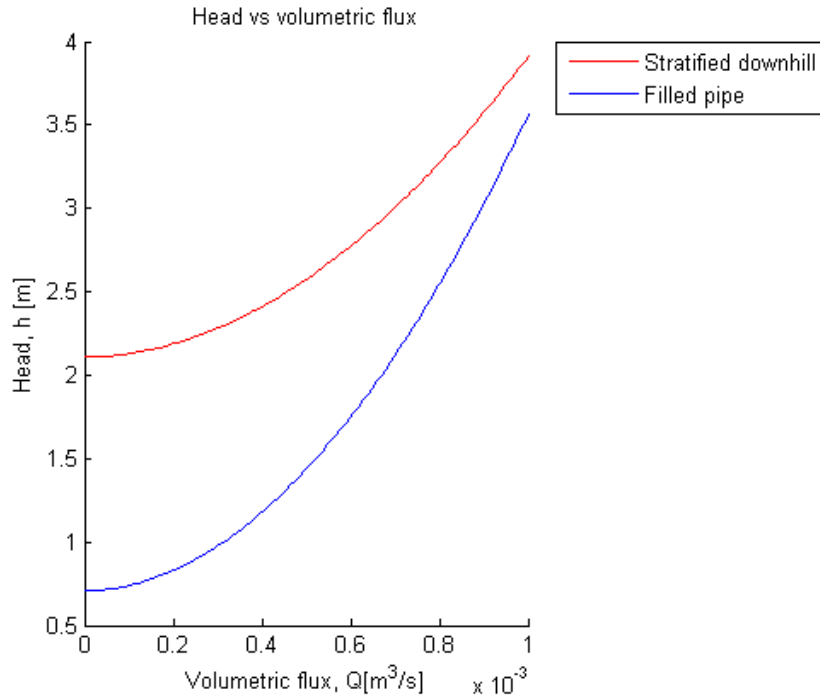


Figure 2.4: Marginal cases

2.3.3 Solution of the cases

Equation 2.16 and 2.17 have been solved for a range of volumetric fluxes, Q and plotted in figure 2.4. This figure displays the head, h needed to achieve the volumetric flux, Q for the cases described above. The lower curve (filled pipe) is the lower threshold and the upper (stratified downhill flow) is the upper threshold. Then, it is known that the head needed to achieve the corresponding volumetric flux lies between these two curves. The limiting cases for achieving any flow is at $Q = 0$. One may also calculate the corresponding bubble turning criteria, to see if stratified flow is possible according to the criteria that is chosen. The Matlab script for calculating and plotting the marginal cases is in appendix A.

2.4 Valve coefficient

C_d in OLGA is the discharge coefficient for the valve. A valve coefficient states in a context the head loss due to the valve. The head loss may be calculated by a number of variables and includes volumetric flow rate. The valve used in the experiments is specified with a valve sizing coefficient, C_v .

In OLGA 6.0 the option for valve coefficient is limited to C_d .

The equation for calculating the discharge coefficient is

$$\Delta P_{orf} = \frac{1}{2} \left[\left(\frac{A}{A_o C_d} \right)^2 - 1 \right] W_{tot} \sum_i \alpha_i U_i \quad (2.18)$$

where ΔP_{orf} is the pressure drop over the valve, A the pipe flow area, A_o is the choke (orifice) flow area, W_{tot} the total mass flux, α_i the volume fractions of flow field i and U_i velocity of flow field i . The stroke time is zero (diaphragm) thus $A_o = A$ and there is single phase flow, $\sum_i \alpha_i U_i = U_{liquid}$. Equation 2.18 simplifies to

$$\Delta P_{orf} = \frac{1}{2} \left[\left(\frac{1}{C_d} \right)^2 - 1 \right] W_{tot} U_{liquid} \quad (2.19)$$

From this one may observe two marginal cases regarding C_d . $C_d = 0$ will cause infinite pressure drop and consequently zero flow. $C_d = 1$ causes $\Delta P_{orf} = 0$ and no pressure drop over the valve.

The equation for the pressure drop using valve sizing coefficient C_v is

$$\Delta P = G \left(\frac{Q}{C_v} \right)^2 \quad (2.20)$$

The author could not find any correlations for finding a corresponding discharge coefficient for the valve sizing coefficient for an arbitrary flow rate/state. Having the volumetric flow rate and the density, the corresponding discharge coefficient may be calculated. But, the experiments are transient and the flow rate varies significantly in time (variation in density for water is insignificant since the temperature is very close to constant and the pressure variations are relatively low). With the varying volumetric flux it is not possible to convert the coefficients analytically for an arbitrary flow rate.

2.5 Multiphase flow simulator OLGA

OLGA is a 1-dimensional multiphase simulator that is designed to be used by the oil and gas industry to predict the state of multiphase flow in pipelines and through process equipment on a system level. A three-fluid model is used. One for the liquid water and oil phase, one for the gas phase and one for droplets of water and hydrocarbons in the gas phase. An n -fluid model denote that n continuity equations are applied. To couple the continuity

equations three momentum equations are used, one for oil, one for water and one for droplets combined with the gas. A slip relation is applied for calculation of the droplet velocity in the gas phase. A mixed energy equation for all the phases is used. This implies that the temperature flashes instantly between the phases resulting in equal temperature. A total of seven conservation equations need to be solved. OLGA uses a minimum slip relation to determine the flow regime. Two flow regimes are assumed; separated and distributed. In separated flow, the phases flow separately through a cross-section whilst in distributed flow the phases are mixed, see figure 2.1. To close the system of equations, fluid properties (PVT tables), initial and boundary conditions must be applied.

Some measures have been taken to enhance stability and robustness of the program. To calculate pressure, the temperature from the previous time step is used (decoupling of pressure and temperature). A semi-implicit scheme for solving the conservation equations is utilized. To redeem the numerical error that occurs in the scheme, volumetric error is corrected for over time. Volumetric error is the discrepancy between the calculated volume of fluids in a control volume and the actual (physical) volume.

Chapter 3

Experimental setup

A rig has been built to carry out small scale experiments in an undulating pipe. Schematics of the rig are given in figure 3.2 The rig used has two peaks and two valleys. The length of all the straight pipe segments and bends are almost equal. The inlet is situated at the floor of the rig on the left hand side. At the last uphill section some modifications have been tried out. The aim with these modifications was to prevent the siphoning effect that will occur if a water column passes the high point of the loop. A siphoning effect may also occur with air bubbles behind the water front if a slug of water passes the highest point. The solution used in the experiments is illustrated in figure 3.2. This and the other modifications for the last peak on the pipe that were tried out are illustrated in figure 3.1.

3.1 Modifications

The original rig is illustrated in 3.1a. During initial testing it became clear that this setup would display a siphoning effect. The first modification was to elongate the last uphill section to an elevation high enough so that the driving pressure would not be able to drive any fluid over the peak. However, air bubbles seemed to be trapped behind the waterfront due to the rapidly decreasing velocity in the last high uphill section. It was found that it would be a good idea to have stratified flow near the final peak. Secondly the modification in figure 3.1c was attempted. During the first experiments it was observed that the seemingly small difference in elevation (approximately 10 cm) from the final peak to the vent resulted in a definite siphoning effect. Thus the modification in figure 3.1d was attempted. With stratification through a level pipe segment at the level of the peaks it was deemed that this modification would display the most correct results with regards to flushing.

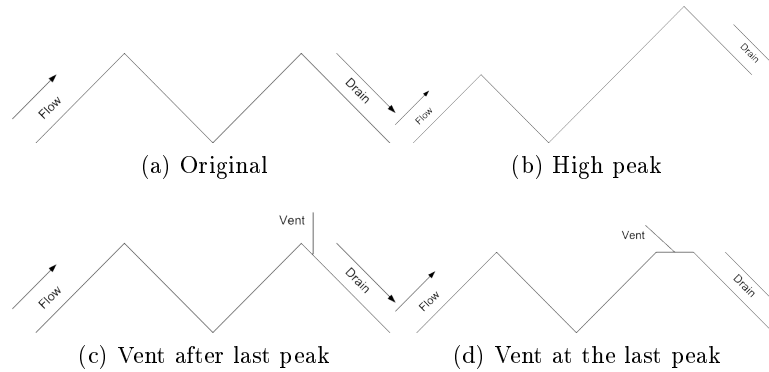


Figure 3.1: The original rig and modifications

3.2 Experimental procedure

The experiments are conducted by opening a magnetic valve and letting the water flow until it comes to rest. A hand operated valve was used in one set of experiments. It was experienced that it was difficult to open the valve fast enough not to cause a significant influence on the flow. Also hand operation will in any case be a source of uncertainty. In the experiments presented in this report only the magnetic valve has been used. A reservoir with a relatively large diameter compared to the volume of the pipe being filled is used. This is to apply an inlet pressure which is close to constant throughout the experiment. After each experiment the pipe is emptied by pressurized air which is applied close to the inlet. The air flow is applied until the interior of the pipe is free from droplets and the pipe is visually deemed free of water.

The height difference between the inlet and the water surface is found by measuring the elevation of the reservoir and the water level in the reservoir. In the various experiments the level of the reservoir is changed to alter the pressure at the inlet.

After each experiment the water is pumped back into the reservoir to approximately the same level. Fine tuning of the height difference of the water surface and the inlet is done by adding small amounts of water to the reservoir manually. The reservoir rests on a manual jack.

3.2.1 Piping and video

The piping consists of clear acrylic pipes in the straight sections and clear flexible pipes in the bends. Fluorescent color is added to the water to cause a strong color in the propagating water. This is to produce video with

Straight sections	Length [m]	Inner/ Outer diameter [m]
P1	0,910	0,016/0,020
P2	0,830	0,016/0,020
P3	0,832	0,016/0,020
P4	0,830	0,016/0,020
P5	0,832	0,016/0,020
P6	0,085	0,016/0,020

Table 3.1: Straight sections

Bend	Length [m]	Inner/ Outer diameter [m]	Elevation [m]
B1	0,155	0,016/0,020	0,700
B2	0,155	0,016/0,020	0,080
B3	0,155	0,016/0,020	0,700
B4	0,155	0,016/0,020	0,075
B5	0,155	0,016/0,020	0,715

Table 3.2: Bends

better visibility of the propagating water and facilitate computer based video manipulation.

A standard high definition video camera has been used to record the experiments. To reduce the influence of the surroundings a black sheet was mounted behind the rig. Along the pipeline a scale was mounted with markings every tenth centimeter. The markings were removed as they caused bad contrast resulting in poorer performance of the video analysis.

The tank elevations tested are in the results chapter, 6

Brand	Make	Type	Cd
ASCO	Magnetic	Diafragma	?

Table 3.3: Valve specifications

Brand	Chemicals	Color
Merck	Natrium and Sodium	Fluorescent green

Table 3.4: Colorant

Brand	Model	FPS	Resolution
Sony	HDR-UX7E	25	1920x1080

Table 3.5: Video camera

Diameter [m]	Length [m]
0.04	2.3

Table 3.6: Discharge pipe

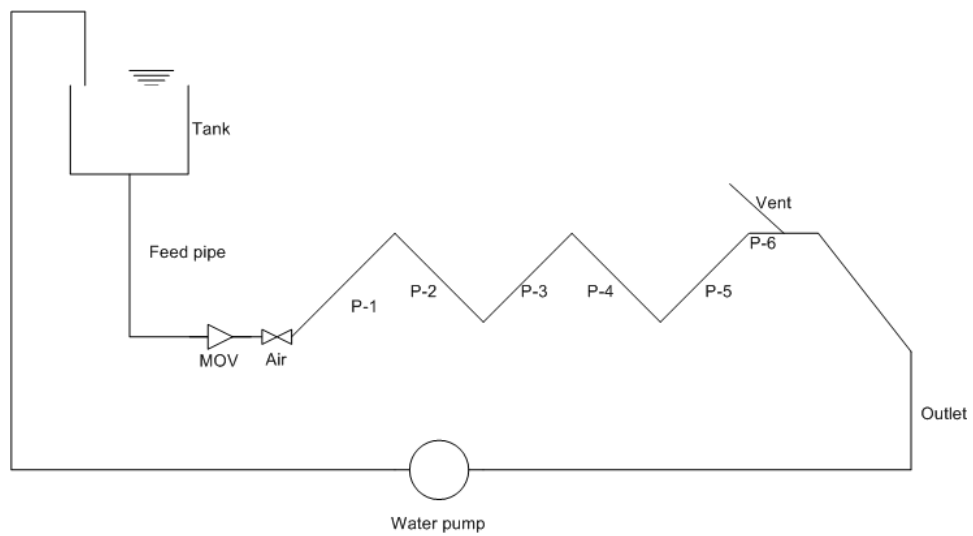
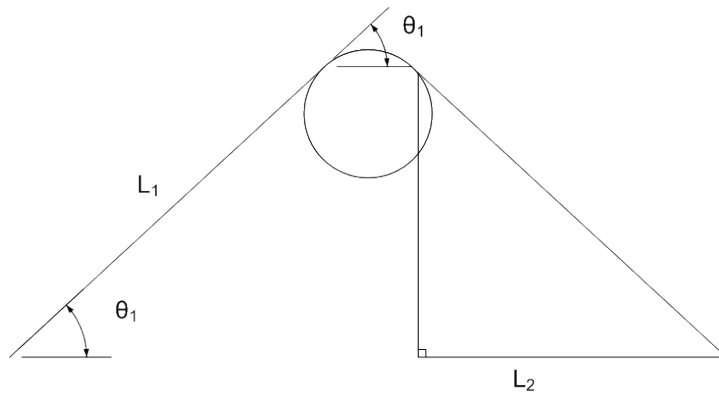
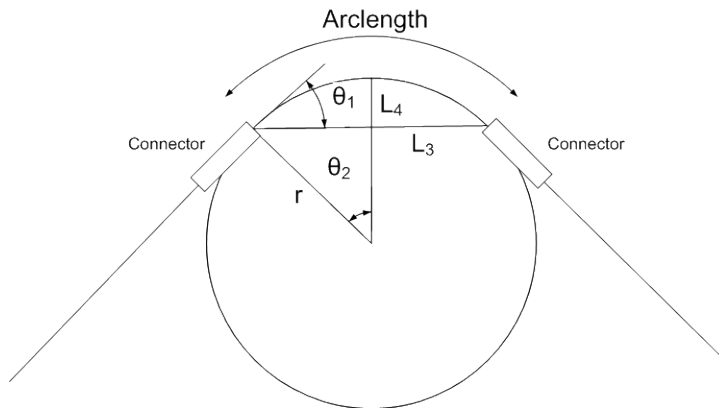


Figure 3.2: Schematics of the rig



(a) Bend



(b) Close-up

Figure 3.3: Geometric relations for the bends

3.3 Geometric relations

For the OLGA simulations with discrete sections modeling the bends, some relations for the bends had to be found in order to approximate the curvature and arch length.

The equation for the arclength is

$$S = \frac{\theta_2 L_3}{\cos\left(\frac{\pi}{2} - \theta_1\right)} \quad (3.1)$$

where $\theta_2 = 2\theta_1$.

θ_1 is found from

$$\theta_1 = \arccos \frac{L_2}{L_1} \quad (3.2)$$

where L_1 is the length of a straight pipe section. Connectors are mounted between the flexible bends and the stiff straight pipes. These connectors are so stiff that the part of the flexible bend that was mounted inside the connector was considered part of the straight section. However, this is only relevant for setting the length of the straight sections in OLGA since only the slope of the straight sections are of interest. The length L_2 was then measured as the horizontal length L_2 in fig 3.3a. The length L_3 was measured as the flexible pipe between the connectors. All measurements on the pipes were conducted from the radial middle of the pipe.

The radius of the bend is found from

$$r = \frac{L_3}{\frac{\pi}{2} - \theta_1} \quad (3.3)$$

The depth of the bends, L_4 is of interest when setting up the OLGA case in order to have the correct elevation of the peaks relative to the valleys.

$$L_4 = r - L_3 \tan \left(\frac{\pi}{2} - \theta_1 \right) \quad (3.4)$$

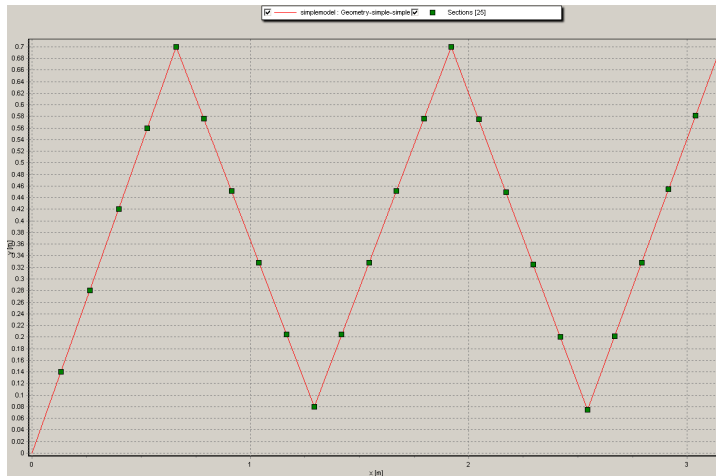
Chapter 4

OLGA Case description

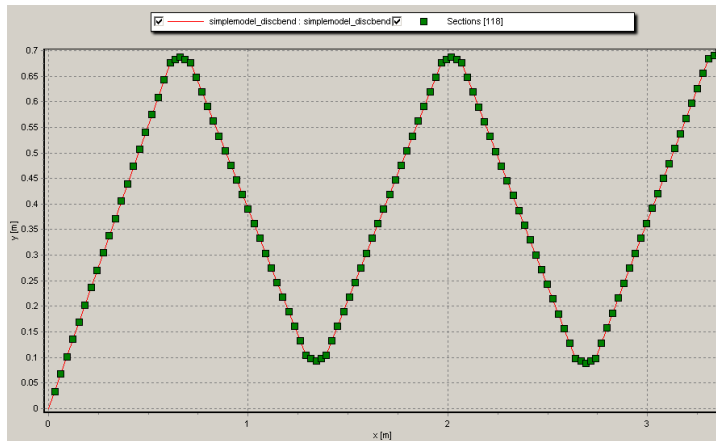
4.1 Overview

The scope of this project is to compare small scale experiments in an undulating pipe with simulations performed with the multiphase simulator OLGA. The most important is to investigate OLGA's capability to calculate the head pressure needed to flush a pipe. Setting up the experimental rig with downhill and uphill sections of slopes around 40 degrees is thought to put the capabilities of OLGA to a test. This is due to that stratification of flow is most likely to occur at a downward slope of 45 degrees according to Zukoski (1966) [11] (section concerning Criterion for transition to open channel flow in [2]). Having multiple uphill and downhill sections around this slope with relatively short straight pipe sections it was assumed that this configuration would be a good test case for OLGA versus flushing of pipelines.

The OLGA model has known and unknown simplifications versus the real test rig. In this report the simplifications that are thought to have the most influence on the results will be emphasized. Especially issues regarding simplifications made due to limitations in OLGA that prevents the end user to regenerate flow phenomena, initial and boundary conditions and geometry. Finally the documentation of the OLGA simulator was often conferred with. Unfortunately, in many cases the documentation did not provide enough information about the options to know what the intended function of the option was. This means the user has to guess the effect of setting a number of options.



(a) Sharp bend



(b) Bend divided in sections

Figure 4.1: Sharp and sectioned bends

4.2 The configuration

There are two types of configurations that have been used:

1. Sharp bends
2. Bends that are divided into sections

The two different types of configuration are illustrated in figure 4.1.

In the OLGA documentation it is stated that if there are changes in inclination in a pipe, it should be divided into at least two sections. There are simulations both with sharp bends as in figure 4.1a and of bends that are divided into sections as in 4.1. In the configuration with sectioned bends,

there has to be at least one node per section. Since the bends in the first place are short compared to the straight sections, the distance between the nodes in the bends will be small. By experience it is recommended that the ratio between adjacent sections should be equal to or less than a factor 2. There are two ways to remedy this for the straight sections.

1. Have an equidistant grid throughout the pipe based on the distance used in the bends
2. Increase the distance of sections from the bend towards the middle of a straight section

To calculate the position of the sections in the bends, a Matlab script has been developed. The length of sections and position is calculated by differentiating the equation for the arc length $s = r\theta$ and integrating it numerically to achieve discrete nodes (coordinates) with straight sections in between. The Matlab script is in appendix B

4.3 Initial and boundary conditions

The test case in OLGA is as close to the laboratory rig as possible. However, numerical instabilities have been encountered. A pressure controlled system is prone to instability. The experiments are pressure controlled by the elevation of the water surface of the tank and the friction of the fluids flowing through the pipe. The alternative in OLGA is to have a mass transfer controlled system which is not consistent with the experiment. A mass controlled system will force the mass flow rate into the system regardless of the pressure. This is not the case in the experiments where the friction from the increasing water column and the effects from the undulating terrain retard the flow.

4.3.1 Feed pipe

One reasonable setup has been found. This is illustrated in figure 4.2a. The feed pipe from the tank has been omitted in figure 4.2a but kept in 4.2b. Omitting the feed pipe produces a smaller pressure drop because there is less wall friction. In the full model the feed pipe is horizontal but the pressure is set at the source "tank". Then the pressure at the source can quickly be modified between each simulation to replicate the change of elevation of the tank in the lab experiments. An alternative configuration is to vary the height of the feed pipe. This is cumbersome in OLGA and has not been performed except for some testing to see if this configuration was more stable. The head loss coefficients for the sudden contraction between the feed pipe

and the pipeline as well as the inlet head loss in the tank outlet has been added in the simple configuration. This has been included as "Loss" process equipment. This loss coefficient in the OLGA case was set to 0.605, the sum of head loss due to sudden contraction and the inlet at the tank, see 2.3.1.

4.3.2 Inlet source

The source used is pressure driven, referred to as SOVA in OLGA. The source has to be pressure driven in order to be consistent with the experiments as explained above. A SOVA comprises a source as it is understood in the literature and a valve. There are two valve equations for the SOVA, Liquid valve sizing equation and Gas valve sizing. This has been set to Liquid valve sizing which is appropriate for a pure water source. The valve discharge coefficient has been set to the default value, 0.86. In the full model a regular valve is set at the point where the actual valve is in the experiments. Further information about source and SOVA in OLGA can be found in the OLGA documentation.

4.3.3 Initial pressure

In all experiments the initial pressure in the air filled part of the pipe is hold at 1bara (atmospheric pressure). The pressure at the outlet is hold to 1bara throughout the simulation which is consistent with the vent at the top of the last uphill pipe on the rig. In the full model the initial pressure in the water column in the feed pipe was set to the corresponding hydrostatic pressure in the rig. Another option that was tried out was to set the initial pressure to 1bara in the feed pipe and let the mass build up in the feed pipe before opening the valve, that is labeled MOV (Magnetic Operated Valve). Neither of these cases were numerically stable and consequently no simulation results have been produced with this set-up. The error message from OLGA was either that the pressure was negative or out of the PVT table, up to several hundred bara.

4.3.4 Pressure drop

Finally, the pressure from the tank in the experiments is not constant. Neither is the pressure drop through the feed pipe and the magnetic valve. First the water surface in the tank drops during the experiment. This is in the millimeter scale and can be approximated as constant pressure. Secondly the pressure drop in the feed pipe is varying with velocity. In the feed pipe, there is single phase flow and the pressure drop can be calculated with single phase fluid mechanics as described in [10] pages 369 - 370. The flow will be

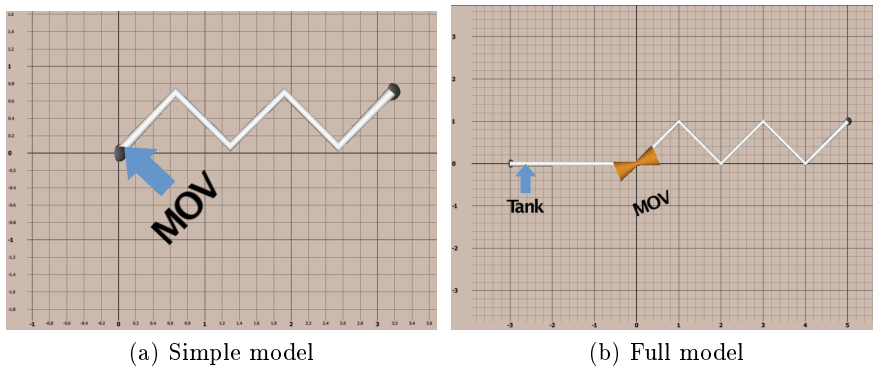


Figure 4.2: Full and simplified model

single phase in the section since the feed pipe is initially filled with water and it is connected with an uphill section preventing any bubble turning process to take place at this location in the pipe see section 2.2.1. The pressure drop through the magnetic valve is determined by the valve equation (see section 2.4). The pipe roughness for the pipes used was not retrievable. Both the local vendor in Trondheim, Hatling AS, and the producer, Gevacril, were unable to deliver this information upon request. The surface roughness was set to $1E-5$ meters.

4.4 PVT tables

PVT is an acronym for Pressure Velocity Temperature. The PVT tables used in OLGA consist of fluid property definitions related to pressure, velocity and temperature. They are based to a large extent on experimental data. The reason why PVT tables are necessary when simulating a multi-phase flow is because single phase correlations are not applicable in a multiphase flow. Relations between the fluid properties have to be measured experimentally and then applied to simulations.

4.4.1 PVTsim

The PVT tables used in the simulations in this report have been generated using a program called PVTsim developed by Calsep (<http://www.pvtsim.com/>). This program is aimed at calculating fluid properties for flows containing hydrocarbons and process fluids for petroleum industry and especially designed to be used with OLGA. The experiments are carried out with air and water. In OLGA 6 which has been used for the simulations there is only an option to choose 3-phases (fluids). Then OLGA 6 requires a PVT table with 3 sets

of fluid properties. The way this has been resolved is to use air as the base component in PVTsim and add water. The result is that the gas and liquid phase have the properties of air and the water phase has properties of water. This generates warning messages about the liquid phase in OLGA. These warnings can be disregarded since this phase is not added in the simulation.

4.4.2 Interpolation

The PVT tables consists of discrete tabulated values with a range and interval defined by the end user of PVTsim. OLGA in turn interpolates between these values to find an approximation that is consistent with the pressure, velocity and temperature calculated. It is recommended in the OLGA documentation to generate PVT tables that have a range well outside of what is anticipated for the calculations. This is due to the iterative process OLGA uses to reach a solution in a single time step.

4.5 Numerical issues

As described above, the test rig is pressure dominated. This forces the simulations in OLGA to be pressure controlled in order to simulate the same case. In OLGA this is set up as a source at the inlet with a constant pressure equivalent to the pressure of the water column from the tank to the inlet. The 'node' in this end of the pipe (upstream from the source) is closed. At the outlet the node is open and set to a constant pressure. Experimenting with setting up the OLGA case it was found at an early stage that a pressure controlled system was vulnerable to instabilities.

4.5.1 CFL condition

In OLGA the Courant-Friedrich-Levy (CFL) condition is automatically switched on. This condition prevents the mass from being transported over the whole section (control volume) in one step (see literature e.g. [9]). The CFL condition is:

$$\frac{u \cdot \Delta t}{\Delta x} \leq C \tag{4.1}$$

u is the velocity, Δt is the time step, Δx is the length interval and C depends on the particular equation to be solved. In OLGA, the CFL condition limits the time step Δt to satisfy the condition since the length interval (the

sections) is user defined. The alternative is to have an adaptive grid depending on the velocity. That adjusts the length interval Δx so the condition is fulfilled. There are also other stability criteria that may be switched on in OLGA (see OLGA documentation). The CFL condition should be the most limiting of these and no improvements to stability were observed by using other stability conditions available in OLGA.

4.5.2 Stability vs resolution

Stability decreases with the number of nodes ($n = \frac{L}{\Delta x}$) and increases with the number of time steps ($k = \frac{t_{end} - t_{start}}{\Delta t}$) (note that Δt is calculated with the CFL condition, equation 4.1). In this context nodes are the calculation node in the end of each section in OLGA. Since the number of nodes (sections) is given by manual input in OLGA, the stability of the case is to an extent governed by the input from the end user. Reducing the number of nodes gave increased stability. Since the calculations in OLGA are compared to video manipulation of the experiments, the number of nodes should correspond to the output of the video post processing. That is, the resolution i.e. accuracy of the OLGA output should be at least the same as the accuracy of the video output. This is to facilitate comparison of the output from simulations and experiments. Having large control volumes smears out the results which in turn makes like for like comparison less accurate with the experiments. The frame rate (the number of frames per second) in the video is 25 (one frame per 0.04 seconds). Then the number of nodes in OLGA should be such that there is at least one node in the OLGA calculation grid per position of the water front in each frame of the video.

4.5.3 Stable model

The OLGA case has been set up as close to the rig in the laboratory as possible. Modeling the feed pipe with a source ((figure 4.2b)) with stable calculations was not possible. The closest stable OLGA case was to set a pressure controlled source at the inlet of the first pipe (figure 4.2a). With this set-up some assumptions about the pressure drop at the outlet of the tank and through the feed pipe have been made, see the section 2.3.1.

4.6 Other issues

4.6.1 Slugs

In setting the SLUGVOID option AIR was used. The slugvoid determines the void fraction in liquid slugs. According to the documentation this is appropriate for simulating an air/water case. According to the OLGA documentation flow history at a pipe location is to a small extent considered in the flow regime. That is, if slug flow is predicted in an upward pipe and stratified flow is predicted in a downward pipe, the slugs will die at the entrance of the latter pipe. In reality slugs may persist in the downward pipe. For the experiments with a set-up that is to obtain stratification in the downhill pipe, see section 4.1, this is relevant. The limitation of OLGA in this matter should not be as important as it would in, say, a flatter configuration with more slug flow tendency in the downhill pipes. An experimental configuration with smaller angles is consequently recommended as a future test case.

4.6.2 Valve

The magnetic valve has a finite opening time greater than zero. That is, the water will flow through a partially opened valve from it starts to open until it is fully open in the experiments. The time it takes for a valve to fully open from fully closed is called 'stroke time'. There is an option to set the stroke time in OLGA. But there is no option to state at what time in the simulation the valve is starts to open. A time series may be used instead. In the time series one may give input at what time the valve is closed and when it is open. OLGA makes a linear interpolation of the opening time. However, it is not known what has been interpolated. There might be at least two possibilities; the surface area is proportional to the opening time or the valve position is proportional to the opening time. Due to these uncertainties, stroke time has been omitted.

4.7 Simplifications and limitations

A list of simplifications and limitations in the OLGA case is mentioned for future reference.

1. Sharp or sectioned bends (not curved)
2. Connectors between pipe sections are omitted
3. Finite number of nodes (calculation points)

4. Feed pipe is omitted
5. Constant pressure in the source
6. Stroke time = 0

Chapter 5

Post processing

5.1 Matlab

Matlab (version R2007a) was used to analyze the transient progress of the experiments. Matlab is a scientific computer language for computer programming that is simple and quick to write programs in. A lot of functions are predefined to facilitate quick programming. This has been exploited to a large extent in the code for post processing of the experiments. However, the structure of the video analysis and the most important coding are explained for future reference. The Matlab codes are attached in the appendices.

5.2 Image analysis

5.2.1 Overview

Image analysis of the video from the experiments was performed. The output of this analysis is compared to the output from the simulations in OLGA. The information that is extracted is the position and time stamp of the last liquid pixel. The median velocity, see equation 5.1 and appendix D can be computed by the change of the position in time. The velocity of the liquid is often high enough so that the liquid passes through the bend between two frames. The position and velocity of the liquid is compared to the corresponding parameters in the OLGA output. The script for the video analysis is found in appendix C.

5.2.2 Importing video files to MatLab

In the version of Matlab used the video has to be imported in a special avi format. The video was recorded in a high definition format, mpeg 2 transport stream (m2ts). To keep all information from the high definition video, it was converted into uncompressed avi format in one step. When processing the images in Matlab, the images are loaded one frame at a time since uncompressed avi files are very large. A flowchart for the video analysis is illustrated in figure 5.1.

5.2.3 Investigation of pixels

The current analysis consists of observing the change of pixels from one video frame to another. A significant change of color corresponding to liquid motion is defined as the change in color above a certain threshold. Since the video almost always has small changes in color due to differences in lighting, the changes under a given threshold need to be filtered out. Setting up criteria for change of pixel value requires that the frame is stored and compared to the previous and / or next frame using a relatively complicated filter. This will take time in both developing the code and running it. Therefore a color has been added to the fluid to generate a good contrast with the background. Then one is able to filter out the motion by using a color filter, see figure 5.1b.

5.2.4 Color filter

The color format of the transformed video was UINT8. In the transformed images on matrix form, each matrix comprising an image consists of $n \times m \times k$ numbers. n and m are the vertical and horizontal position of the pixel respectively (starting with 1,1 at the top left corner). k denotes the last dimension in the 3-dimensional matrix. The length of k is 3. k consists of three components corresponding to the intensity of red, green and blue (RGB) respectively. The intensity of each color is an integer ranging from 0, the darkest to 255, the lightest.

In developing the color filter for detecting liquid motion, two different filtering methods were tried. The first method was to set the intensity of green to a minimum allowing all pixels with a value of green over some threshold to be registered. Then setting the maximum value of red and blue so that dark gray scale pixels are filtered out was tried. This also caused dark green pixels to be lost. Then the pixel values in the green fluid were investigated. Most of the green water pixels had a green intensity over 75 and the blue and red pixels in the unwanted gray scale areas i.e. empty pipe had almost

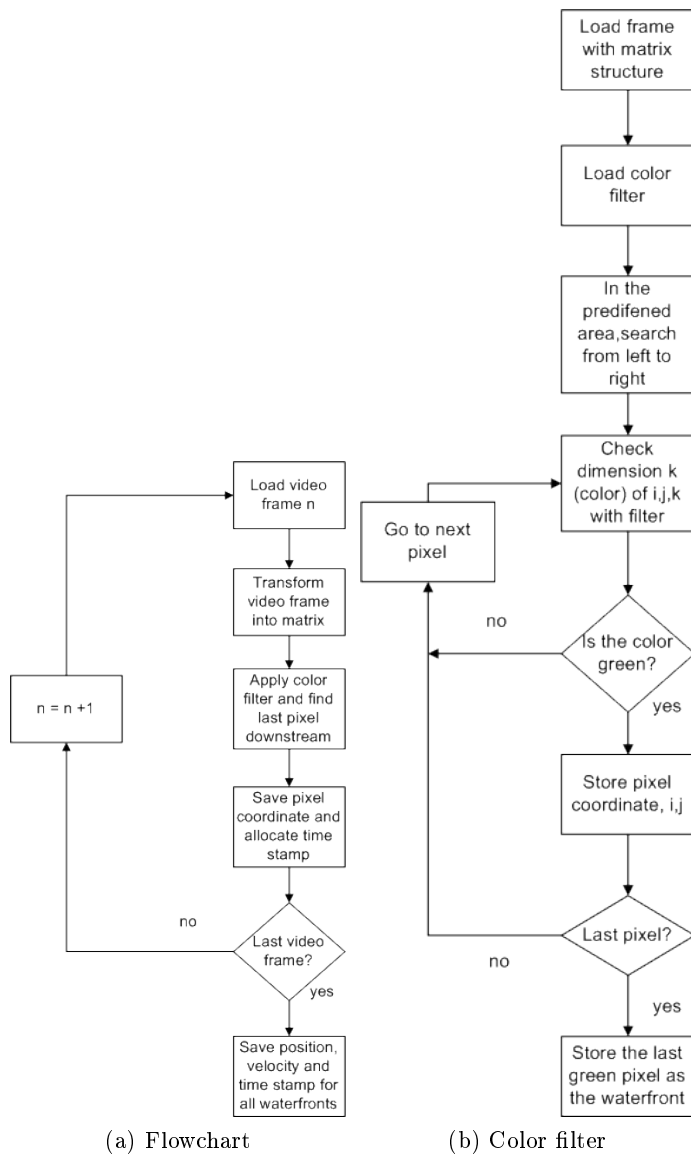


Figure 5.1: Matlab flowcharts for video analysis

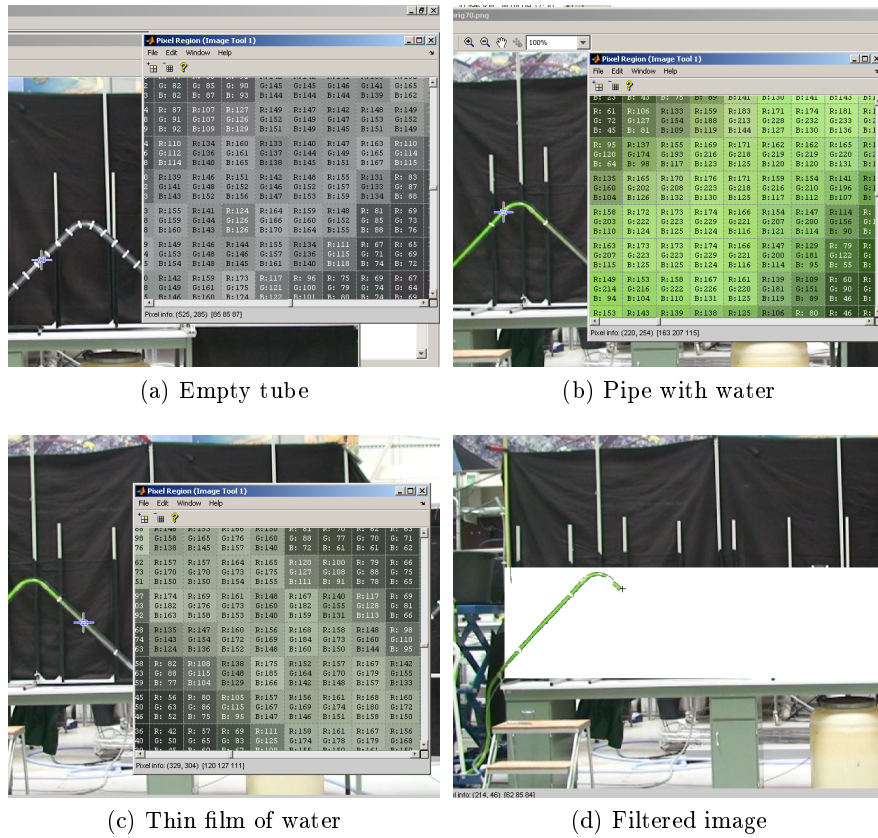


Figure 5.2: Investigation of the pixels in filled and empty sections

the same value as green, e.g. 142,146,147, see figure 5.2a as opposed to a filled pipe with values e.g. 173,223,125, see figure 5.2b.

The second method tested involved keeping a minimum value of green (to avoid black pixels) and a ratio between red and green and blue and green to avoid gray pixels was tested. This filter proved to be a lot more effective than the first one. From investigation of some pixel values it was found that in the pipe containing fluid, green was of strongest intensity and the other two colors were weaker. Tuning of the filter was accomplished by inspecting the images produced with the filter and manually adjusting values for each of the ratios. Therefore this method is subjective. An example of a filtered image using this method is in figure 5.2c. A good example of subjective filtering occur when the head pressure is low and a thin film of stratified water flows in the downward sections before the cross section is filled to a larger extent. The tube looks slightly green, but should this thin film of water be considered as a water front? Catching this thin film of water with the filter also increases the chances of picking up unwanted gray scale pixels. An illustrative image of the output of the filter is in figure 5.2d. A black cross-hair marks the last

pixel that was considered green by the filter. This was the last pixel stored and it was the furthest downstream. Using this filter, the thin stratified film in the downhill section is not considered as green (water). Setting a high sensitivity (catching thin water films) also considers green shadows outside the pipe to be water. The filter used in figure 5.2d is conservative in the sense that the filter only catches what really is water. On the other hand one can observe a white inclusion in the water column close to the front in figure 5.2d. Investigating the unfiltered image e.g. 5.2b one can see that liquid is actually present in this section. In this example it is not critical to catch the water front since it is further downstream. However, if on a later occasion one would like to use the technique described in this report to perform video analysis in the entire pipe, e.g. for tracking slugs, one has to be very careful in tuning the filter with regards to when water is present. In the final version of the image processing script, the background is subtracted from each frame. This allows setting very low tolerances between green and red / blue intensity. The fraction that was set between the intensities caused the processing script to catch the thin sheet of liquid in downhill pipes to be considered as liquid.

5.2.5 Search algorithms

To optimize the script with respect to finding the water front search algorithms have been briefly investigated. The crudest way to find the waterfront is to step through the whole image, pixel by pixel from left to right (downstream since the water flows from left to right) and updating a parameter containing the position of the current pixel that is considered to be water. In the end, this waterfront parameter will contain the position of the last updated pixel within the criteria of the filter and thus be considered as the waterfront. This is a simple but not very reliable method. Directly behind the rig a black sheet has been mounted to create a good contrast towards the liquid. The struts have also been masked with black tape to avoid this being captured by the filter. Initially, a rectangular area is declared in the program containing the pipe. To optimize the code both with regards to speed and accuracy a search algorithm could be implemented. This has not been done since optimization of the computer code is not a part of this project. However, some suggestions on how this could be done are presented. A strategy for estimating the position of the waterfront in the next frame based on the velocity of the current waterfront and the time step between the frames could be implemented. Using this, one should keep in mind the configuration of the rig, with straight sections and bends. Another approach could be to start the search at the previous water front and perform a search stepping away from the start point in all directions (a circle) until a criteria is met. The quickest algorithm that has been investigated is to implement

binary search. Binary search is suitable for structured arrays e.g. containing numbers with increasing values. But the pixels of the image are not structured. Then one has to implement a method in the script to find the exact location of the pipe within the matrix and only search in this area. The pipe containing water will be identified by the color filter. Then the binary search algorithm may be adjusted to search for the last pixel that has a color spectrum within the criteria of the color filter. What has been implemented in the video analysis script is an option to choose the size of a box in which the water front is searched for. This box is shifted 50% of its length to the right. The program will leave a warning if the water is found on the edge of the box indicating that the box is too small to find the front of the water. It is shifted to the right (downstream if the velocity is positive) to minimize the search area since the largest velocities occur when the water is propagating in positive x-direction. After an experiment, some liquid droplets would remain in the pipe. This was especially a problem at different fittings. To avoid the program to deem these droplets as the water front, an algorithm was implemented to ignore droplets. A droplet is defined in the program if the number of neighboring liquid pixels is smaller than a value chosen by the end user.

5.2.6 Output

The scope of this project is to find to what extent OLGA is able to reproduce experimental results. The most important data to extract from video analysis was the position of the waterfront when it propagates through the pipe. Having that, and the corresponding time stamp, median velocity for the waterfront can be found. From the contractors point of view (Professor Ole Jørgen Nydal), the video analysis was successful if the position of the water front was found. In addition, the color filter could be used to analyze the motion behind the waterfront. Then hold-up and velocities of air bubbles and slugs could be found. However, this requires correlations between the intensity of the fluid color and water fraction. Droplets will stick to the pipe wall, the interface of the water phase might not be straight and diffusion of light makes the air appear green. These phenomena make analyzing cross sections containing both air and water very difficult. Also, in finding the water fraction based on the color, will need a very high resolution. With the distance from the camera to the rig and the resolution of the video, the pipe diameter is in the range of 20 pixels. This resolution is insufficient for calculating water fraction.

5.2.7 Future improvements

There are two key issues that need to be resolved in order to perform video analysis behind the waterfront:

1. Finding the water fraction behind the water front
2. Resolution

To determine the water fraction correlations for finding the water fraction based on the color of a pixel, one can measure the water fraction while filming at some location. Then either make a table of water fraction versus pixel value to look up the pixel values in the experiments or try to make a correlation based on pixel values versus water fraction to be used explicitly. To resolve the issue of resolution one can either simply move the camera closer to the rig but having to use several cameras to film the whole rig. This may result in considerable editing of the video from each camera. The other way of resolving this is to use a camera with higher definition resulting in larger files for post processing. One may also use pipes with large diameter, but the the rig will be longer.

5.3 Post processing of output from video analysis

The output from the video analysis script consists of and x and y coordinate in the image matrix allocated with a time stamp. The coordinates need to be scaled to physical coordinates (dimension [m]) to be compared with OLGA output. This is done in the post processing script for the image analysis in appendix D. Scaling of the pixels is done by measuring the distance between two points on the rig. Then the pixel values of the points is found and the ratio between pixels and meters on the rig is found. To a small extent, the camera optics project the picture in a non-linear fashion and could result in an error in the scaling method mentioned. However, this error is assumed to be small. Another issue about the script is that some remaining liquid may be in the pipe. If at some time this moves, the pixel value in the area will differ from the background image. The image analysis script will catch this movement if it occurs in front of the waterfront. This is an error due to difficulties in completely emptying the pipe from liquid after an experiment and it should be disregarded. In the post processing script there is an option to either linearly extrapolate the previous velocity or to set the velocity to zero when this happens. Extrapolation may be elaborated to take into account a set of earlier points. It cannot use later points, as these are prone to be erroneous at the current location. I.e. if the remaing liquid has been displaced in a time interval, it is likely that subsequent points will be erroneous.

The velocity calculation comes from the following equation.

$$u = \text{sign}(\Delta x) \frac{\Delta S}{\Delta t} \quad (5.1)$$

where ΔS was set as the displacement since the previous position, using Pythagoras' rule on the difference of x- and y-coordinate. The sign (positive or negative velocity) is found from the displacement in x-coordinate. Positive means that the water is flowing to the right further towards the outlet, and negative that it flows to the left. Δt is the time step between the previous and current position, set to 0.04 seconds, the frame rate of the camera. This velocity is actually the velocity in the middle of the previous and current position. For plotting purposes these velocities is allocated to the current position. The error of doing this is considered small since Δt is small, i.e. high resolution in time.

5.4 Post processing of OLGA simulations

In order to be able to compare the OLGA results to the the results from the video analysis post processing, the OLGA output need to be processed as well. The Matlab script for the post processing of the OLGA results is found in appendix E. The results using the trend plot are exported with the mode "on time". The needed output is a parameter to deem if liquid is present i.e. holdup, a time stamp, the x- and y-coordinate. The y-coordinate comes from "geometry". The "geometry" has a different allocation towards the pipe length than the holdup. The geometry is allocated to a position that is in the middle of two pipe length positions. This is thought to be a result of that OLGA uses a staggered grid for calculation of velocity. To remedy this, the same type of velocity calculation for the OLGA output is used as for the video analysis, see equation 5.1. Then the velocity is exact with regards of space for the OLGA post processing. The criteria for liquid present in a control volume is set by a minimum holdup. To remedy numerical diffusion effects and getting unphysical results, this was was set to a rather strict criteria, holdup ≥ 1 .

The length coordinate in OLGA is not the horizontal length. It is the length along the pipe. Since the angles of the pipe sections in OLGA are very close, a factor between the length along the pipe and the actual length of the rig has been used to scale the "length" in the OLGA output. If the pipe sections have a varying slope, a separate factor has to be found for each pipe section.

Depending on the parameters chosen, the output from OLGA shifts its structure. This has forced a lot of changes to the OLGA output script. Currently

some of the names for the parameters in this script are misleading. The reason for that different parameters has been extracted is that it was thought that velocity variables could be used directly to compare with the experimental velocity. Some comparisons were made, but the OLGA velocities, water sheet velocity, volumetric flow liquid and volumetric flow water turned out to be unusable in this context. For reasons that are not known, they seemed to be unrealistic and misleading in comparison of the propagation rate of the water.

5.5 Comparison of the results

Finally the processed output from the video analysis and OLGA was plotted using the script that is found in appendix F. Since the time step in the simulations and the experiments (equal to the framerate of the camera) the outcome of the two post processing scripts could be plotted. It was chosen to plot both velocity and position in the same plot as to facilitate investigation of correlations between the position and velocity. If the time step would have been different in the data sets, some interpolation must have been carried out. Either interpolating the data with the smaller time step to the data with large time steps or the other way around to keep the resolution of the data set with the small time steps.

Chapter 6

Results

6.1 A few notes about the plots

The author has considered how velocity should be plotted vs position for data that has been processed with post processing scripts that have been made. The issue is that the velocity does not go to a constant value when the liquid reaches the end of the pipe. If that had been done, the correlation between position and velocity might have been clearer. The velocity, however is calculated as the change in position vs time. Consequently the velocity goes to zero when there is no change in position (i.e. the liquid has reached the end of the pipe). Then there will be one extra row with change in velocity compared to change in position. Thus, the velocity graph will have one increment extra before going to zero (in the cases the liquid reaches the end of the pipe). Also, one should have in mind that the velocity is calculated from the current position and the previous for a time step.

In some plots from OLGA the labels on the y-axis is not included. This has been included in the caption. Note that "geometry" is the vertical height and "pipe length" is the position following the pipe. The position in all other plots refer to the horizontal position.

6.2 Comparison of OLGA and experimental results

The end state for the experiments (where applicable) is listed in table 6.1. Note that the maximum height of the water column in the simulations is determined by the discretization of the pipe. Since the configuration used in these simulations was with sharp bends, the maximum height of the water column is the elevation of the highest node in the pipe section. The water

Exp/Sim	h [m]	P1 [m]	P2 [m]	P3 [m]	P4 [m]	P5 [m]
Exp	0.45	0.448	0	0	0	0
Sim1	0.45	0.490	0	0	0	0
Sim2	0.45	0.490	0.142	0.142	0	0
Exp	0.675	0.680	0.267	0.237	0	0
Sim	0.675	0.490	0.514	0.514	0	0
Exp	0.705	0.690	0.690	0.681	0.286	0.330
Sim	0.705	0.638	0.638	0.514	0.638	0.518

Table 6.1: Column height at end state

column was determined as the elevation of the node with a holdup ≈ 1 . The holdup in P4 Sim2 was 0.73.

All simulations have been run with the PVT file AirWater3.tab except Sim1 and Sim2. The PVT files are not included in the appendix since they are very long. The PVT files are retrievable from the disk enclosed with one of the copies of this thesis. The simulation in figure 6.1 (Sim1) was ran with the PVT file AirWater2 and for the simulation in figure 6.2 with the PVT file AirWater4. The difference between the PVT file AirWater3 and the others is that AirWater3 has a much shorter range in pressure. The main difference between AirWater2 and AirWater4 is that the amount of water added in PVTsim is less in AirWater4 than in AirWater2. Figure 6.3 and 6.4 are two cases where the water passed the first peak but did not flush the pipe. In figure 6.5 the pipe was flushed in the experiment but not in the simulation. Figure 6.6 displays a parametric study in OLGA of the pressure just below and just above what is needed for OLGA to flush the pipe.

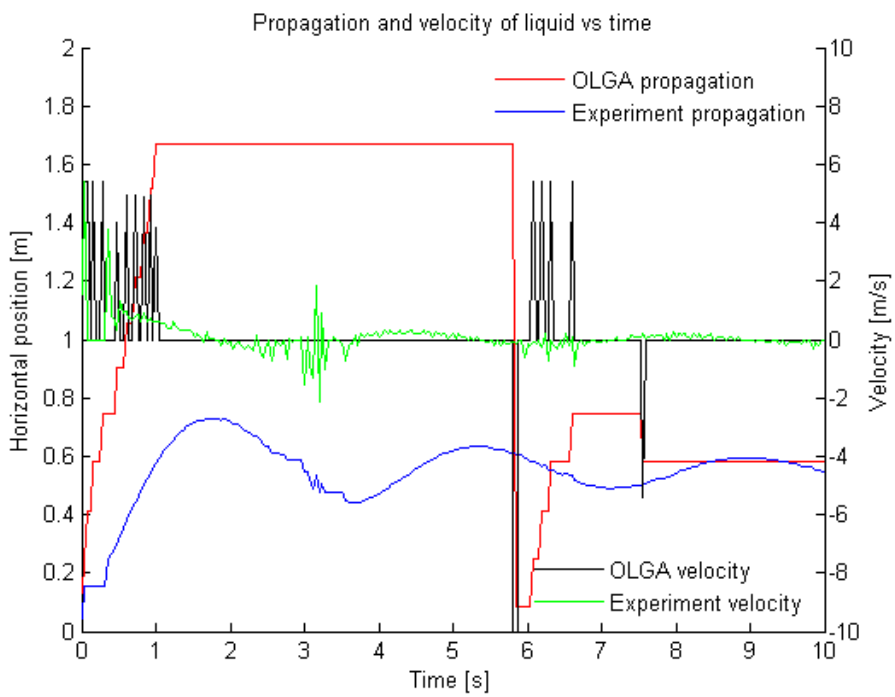


Figure 6.1: Experimental and simulation data for water surface 0.450 m, PVT AirWater2

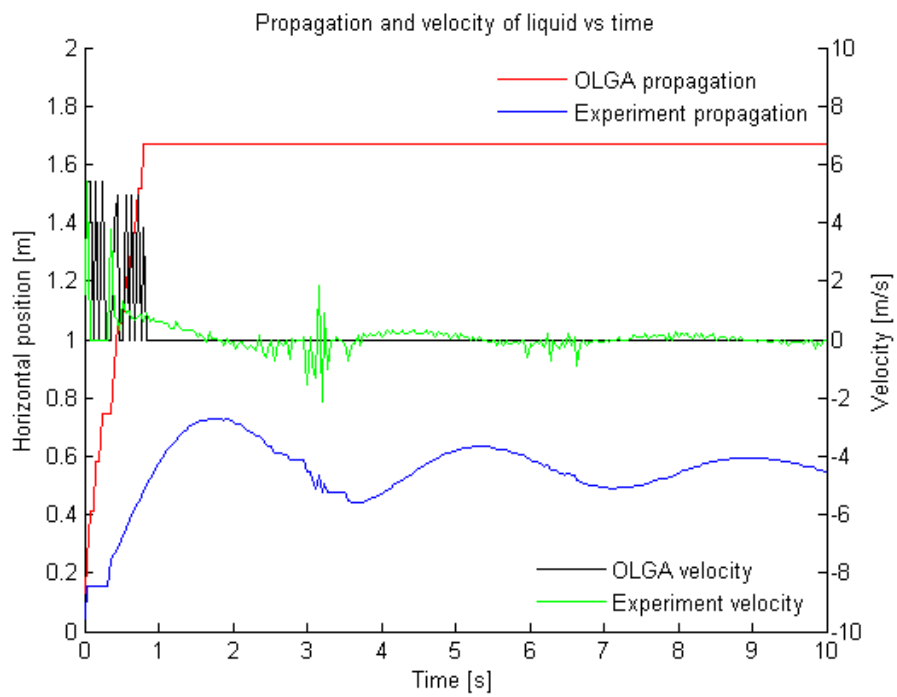


Figure 6.2: Experimental and simulation data for water surface 0.450 m, PVT AirWater4

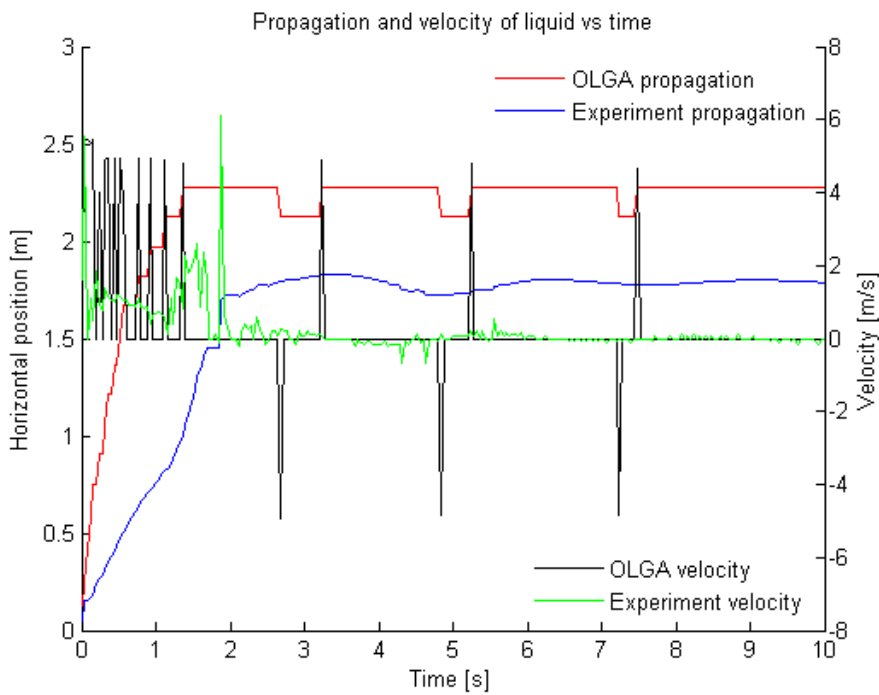


Figure 6.3: Experimental and simulation data for water surface 0.675 m

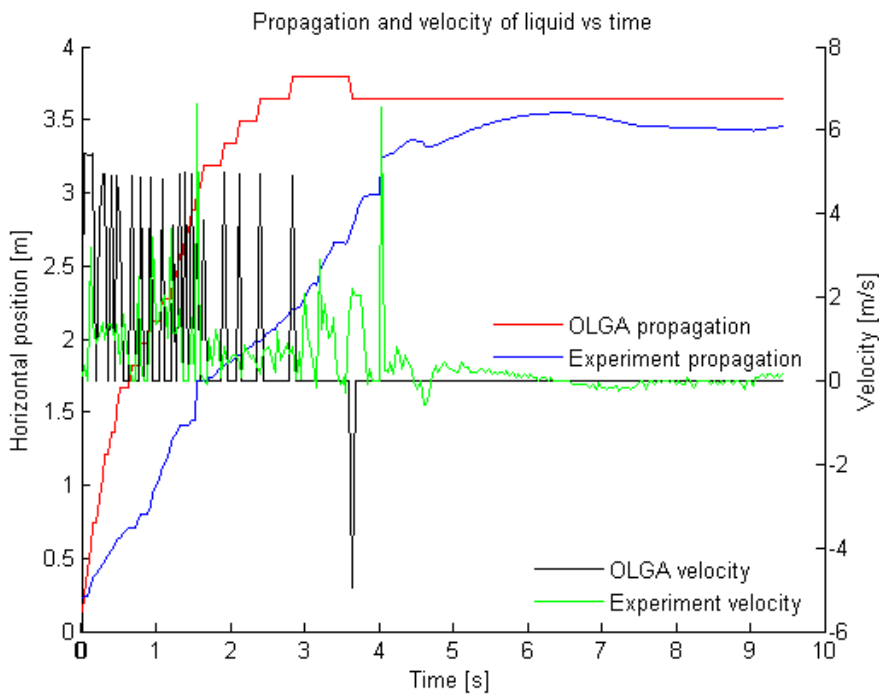


Figure 6.4: Experimental and simulation data for water surface 0.750 m

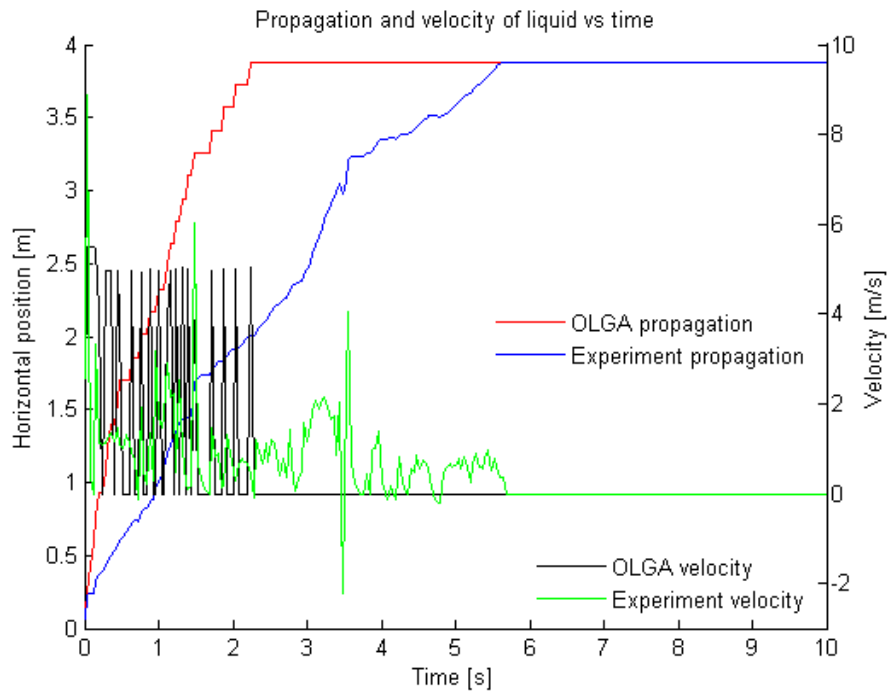


Figure 6.5: Experimental and simulation data for water surface 0.825 m

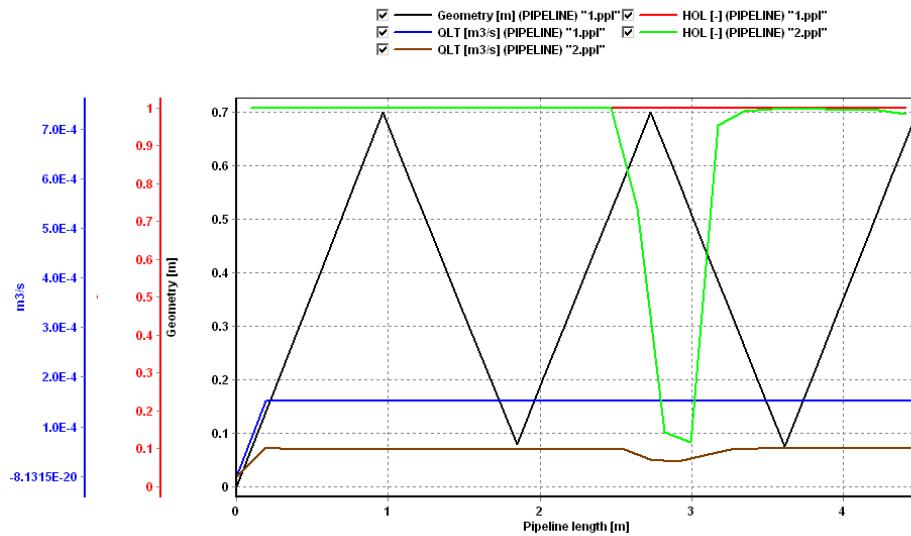


Figure 6.6: Parametric study of the inlet pressure vs flushing. Y-axes: dark; vertical height, red; holdup, blue; total volumetric flow

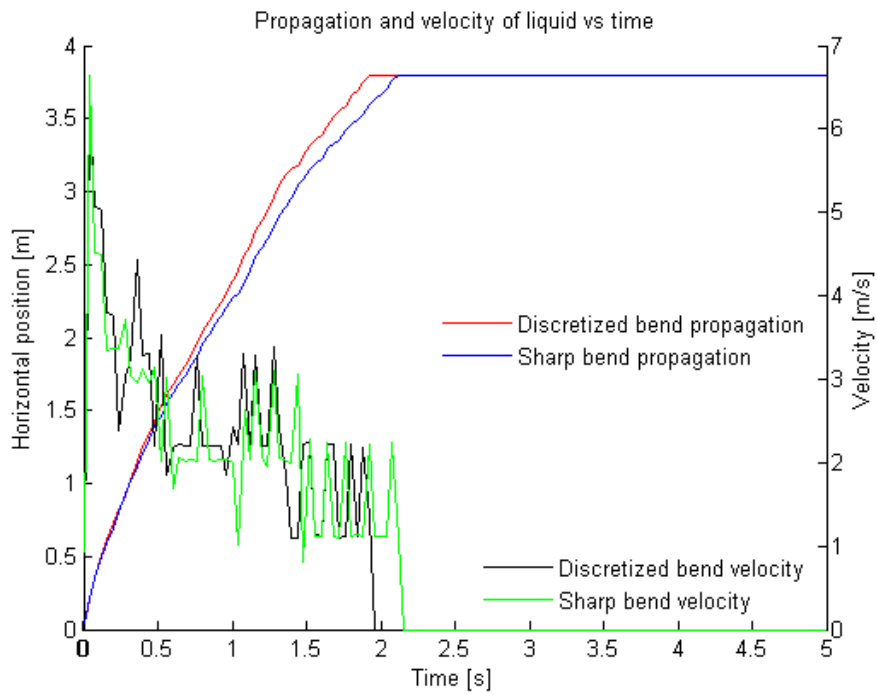


Figure 6.7: Comparison of sharp and discretized bends

6.3 Discretized bends vs sharp bends

Figure 6.7 displays the position and velocity for the first liquid particle versus time. Both simulations had the same number of sections (118). The source pressure was set to 1.1 bara to avoid numerical instabilities.

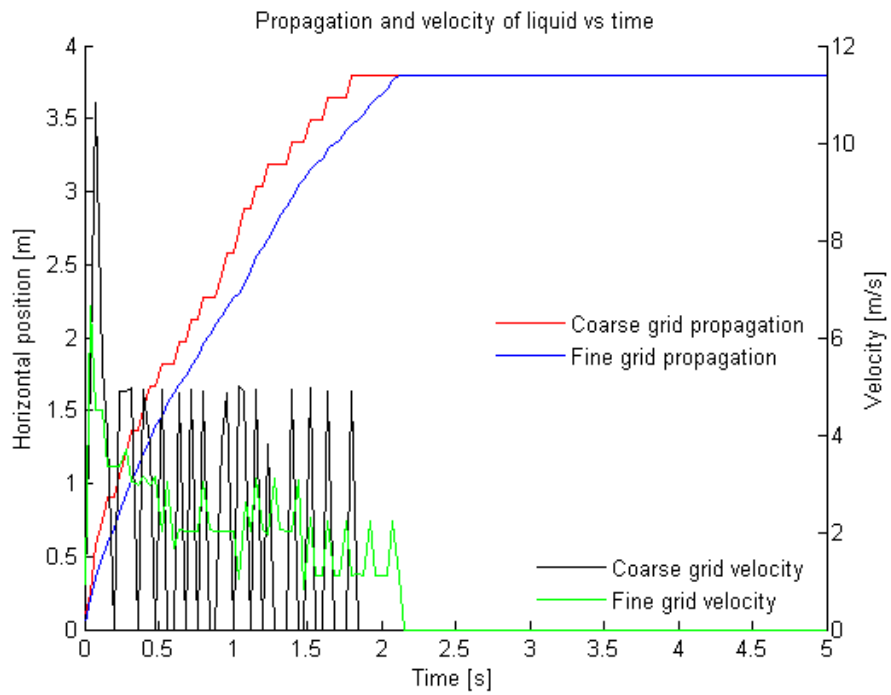


Figure 6.8: Comparison of coarse and fine grid

6.4 Coarse vs fine grid

Figure 6.8 displays the different results between a fine grid (118 sections) and a coarse grid (25 sections). The pressure in the source was set to 1.1 bara and sharp bends were used.

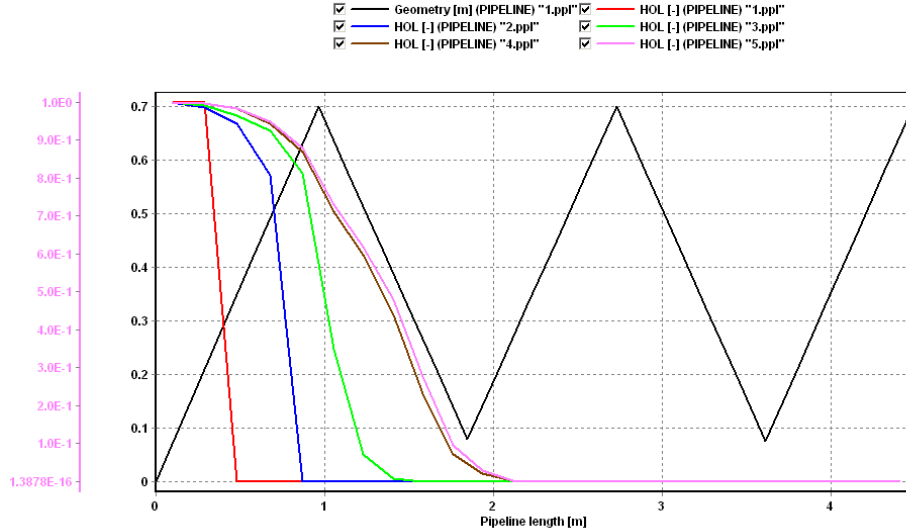


Figure 6.9: Parametric study C_d , 0.04 seconds. Y-axes; dark; height, pink; holdup

6.5 Parametric study of C_d

Figure 6.9 and 6.10 displays the holdup with varying discharge coefficient, C_d in a pipe with sharp bends and an inlet pressure corresponding to an elevation of the tank at 0.75 meters. Figure 6.9 displays the holdup after 0.44 seconds and figure 6.10 after 20 seconds. The black line displays the pipe. The corresponding y-axis shows the elevation. The x-axis displays the distance along the pipeline, not the horizontal. The various colored lines in the plot denotes the different C_d coefficients. The file extension "1.ppl" is $C_d = 0.2$, "2.ppl" is $C_d = 0.4$, "3.ppl" is $C_d = 0.6$, "4.ppl" is $C_d = 0.8$ and "5.ppl" is $C_d = 0.9$.

6.6 Study of velocity profile in OLGA

Figure 6.11 and 6.12 displays the velocity, holdup and volumetric flux profile for a time step in OLGA. The inlet pressure was set to 1.0724677 bara corresponding to an elevation of the tank to 0.75 meters. The configuration used was with sharp bends and 25 sections.

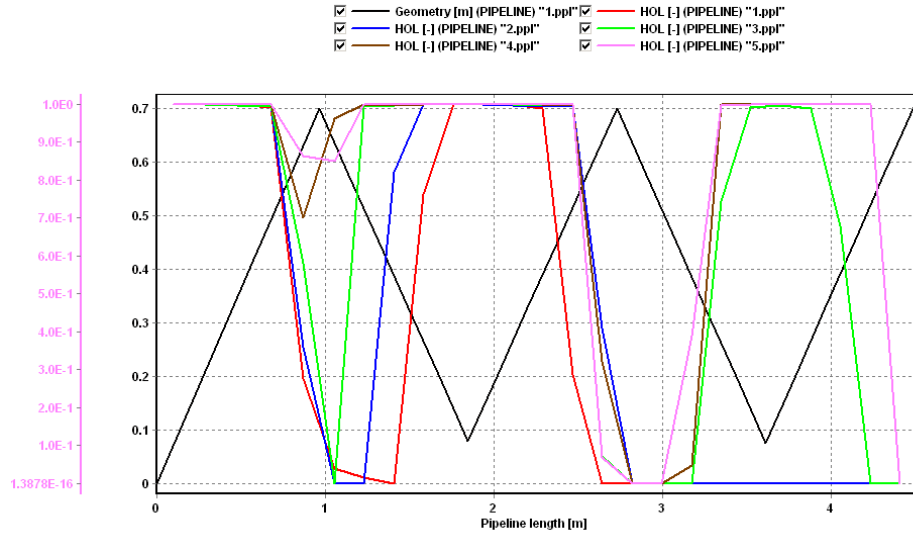


Figure 6.10: Parametric study C_d , 20 seconds. Y-axes; dark; height, pink; holdup

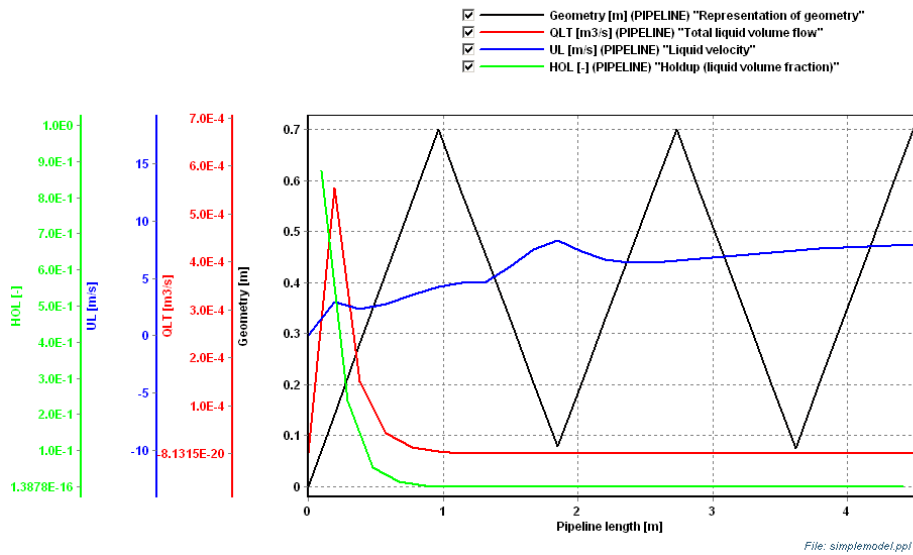


Figure 6.11: Profile after 0.04 seconds

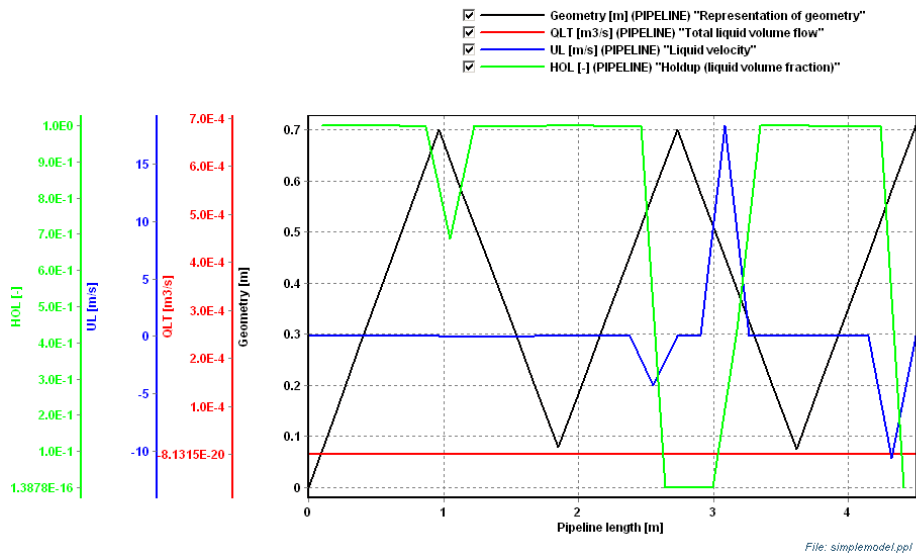


Figure 6.12: Profile after 10 seconds

Chapter 7

Discussion

7.1 Comparison of OLGA and experimental results

7.1.1 End state

For the experiments and simulations where the water did not displace the air, the water came to rest. The water column in the experiments and simulations is given in table 6.1. Due to discretization of the pipe in the simulations, the node with the highest elevation containing water (holdup ≈ 1) was deemed as the column height. For the experiments, the water column was measured on the rig after the water had come to rest.

When PVT tables are generated in PVTsim, water specification must be given. In the water specification there is an option to choose the amount of either % water cut, mol spec water/mol feed, mol% of feed+spec water or weight% of feed+spec water. mol spec water/mol feed was chosen and the amount of water was set to a very low number, eg 0.0001. When PVT tables with a number of water added that was not very low, water appeared far downstream long before the water coming from the source had reach that position. It is thought that this unrealistic accumulation of water is a result of setting a number which is too high for the amount of water in the PVT tables. This accumulation of water might have been due to that OLGA calculates the water in the air to condensate. However the quantity of liquid accumulated from condensation is unrealistically high.

For the experiment with a tank level of 0.45 meters the momentum of the liquid column almost made the liquid pass the first peak. The elevation of the peaks on the rig is approximately 0.68 m above the inlet. It was expected that the liquid would come to rest at approximately the same elevation as the water surface in the tank when no liquid passed the first peak. The experi-

ment confirmed this with an elevation of the liquid column to 0.448 meters. The OLGA simulations displayed some curious results for this experiment. For Sim1 the liquid that passed the first peak figure 6.1 disappeared just before 6 seconds. Then the liquid column in pipe 1 was 0.49 m at end state. A liquid column that is higher than the elevation of the tank should not be at rest. The water column is not at equilibrium since the water column is higher on one side and the liquid should flow towards the low pressure side. Another simulation with a different PVT table was carried out due to the mysterious disappearing fluid. In this simulation the water barely passed the first peak and the very bottom of the subsequent pipes were filled with water. For this simulation the first pipe had a holdup very close to 1 at 0.49 meters elevation. A water column of 0.49 meters elevation indicates that the calculated pressure is higher than 0.45 meters. However, it is not possible to set the source at the inlet in OLGA. The absolute position of the source in the configuration used is 0.0963 meters. This corresponds to a vertical elevation of 0.07 meters. Then the calculated end pressure in OLGA corresponds to a water column of 0.42 meters. It is not known what this discrepancy comes from.

In all simulations with an end state with fluid at rest, OLGA predicted higher water columns downstream than what was observed in the experiments. This was consistent with the transient progress that is discussed below. However, it is not very different from the experiments. For the 0.45 m and 0.675 m cases where the water barely passes the next peak in the experiments, only a very small amount of water propagated to the next valley in the simulations. This shows surprisingly good correspondence between the end state of the experiments and the simulations, even for a coarse grid in OLGA. There might be three reasons for the slight discrepancy for the end state. First the end state of simulation with 0.45 m elevation of the tank displayed a water column in the first pipe of 0.49. Thus the pressure in the simulations must have been higher than in the experiments. Secondly, the total length of the pipe is less in the simulations, since the bends have not been included in the configuration. Then there is a smaller pressure loss due to friction in the OLGA case. Thirdly, there is no model for surface tension between the wall and the fluid in OLGA. In a small diameter pipe, as is the case, this effect is considerable and could be the main reason for the discrepancy with regards to end state between the simulations and the experiments.

7.1.2 Transient progress

Generally, the liquid propagated a lot quicker through the pipe in the simulations than in the experiments. That is seen by that the graphs for the position vs time comes to rest sooner for the simulations and has a faster

propagation through the pipe. Also, the velocity profile lies to the left and is generally higher than for the experiments. Curiously, the end state in the simulations and experiments was surprisingly close given such a large difference in velocity and propagation rate. Higher velocity of a liquid column also means higher momentum. Then a higher frictional work has to take place to brake up the flow till rest. A reason for the end state to be so close between simulations with higher velocity and propagation rate versus the experiments has been identified. Slugs die at the top of an upwards pipe section followed by a downward pipe section if stratified flow is calculated in the downward section, see section 4.6.1. Then terrain slugging will not be calculated in OLGA. As was described in 2.2.4, terrain slugging was very important in the experiments. Certainly, terrain slugging in the uphill pipes was a key factor for the liquid to propagate as far as it did for the experiments, except of course the experiment with elevation 0.45 meters where the liquid did not pass the first peak. Also, the experiment with elevation 0.45 meters showed that momentum was very important for the experiments. The liquid column accelerated rapidly in the first part of the first uphill pipe. This can be seen in the plots of the various experiments and simulations as a very steep slope of the position. The velocity profile should also have a steep slope. This is difficult to observe in the plots for two reasons. For the experiments, the lighting at the inlet was poorer than the rest of the pipe. The result was that the video analysis script did not catch the liquid front before it had propagated a small distance. The result is that the position profile for the experiments is almost straight up from $time = 0$ to $time = 0 + dt$. dt is 0.04 seconds in the simulations and experiments since this is the frame rate of the video camera. For the experiments the coarse grid had a very similar effect. With the coarse grid, the first node is at a distance of 0.1926 meters from the inlet. Due to the flow rate, the holdup was larger than 0.1 (the criteria set for water is present) after 0.04 seconds. Dividing the distance over time the velocity is 4.815 m/s, see figure 6.1. After the initial high peak, the velocity drops to zero since the holdup in the next node is lower than 0.1 and thus water is not deemed to be present. These more or less discrete peaks follow the stepping in the propagation rate in space. For simulations with higher propagation rates, the holdup increases to 0.1 in several control volumes at a time step. Then the velocity will be the number of control volumes (CV) with holdup larger than 0.1 times the length of the CV divided by 0.04 seconds. When the water is deemed as present in the control volume furthest down stream with a holdup larger than 0.1, at many time steps it takes some time before any subsequent CV's are filled up with a holdup larger than 0.1. That is the reason for the distinguished steps in the propagation profile and peaks followed by zero velocity in the velocity profile for the simulations.

After the initial very high calculation of velocity and displacement for the experiments, both curves flatten out considerably except for some distin-

guished points. When the water propagates through the pipe, there are two reasons for this. First the image analysis script has a color filter in order to find the liquid, see section 5.2.4. If there is a mixture of water and air in the pipe, and there is too much air, the filter will deem the color as a shade of gray and ignores the pixel. Also if the number of neighboring pixels that is deemed as water is too low, the liquid in the region is deemed as a remaining droplet (from a previous experiment) and ignored. A good example of a bubble that propagates through a slug and causes the water to flow back and leave some ignored water droplets that are ignored is seen in figure 6.5 at around 3 meters. The propagation profile for the experiment will also tend to flatten out at valleys and peaks. At peaks because a very thin sheet of liquid flowing downhill, the front of the water is at some times caught and some times deemed as remaining droplets and ignored. A good example is the experiment in figure 6.4. In the propagation profile there is a distinguished step at about 1.5 meters when the liquid passes a peak accelerates in the downhill pipe and retards in the next valley. After the liquid had reached its maximum displacement downstream, back flow would occur if the liquid did not reach the end of the pipe. When back flow occurred, droplets of water would somewhat stick to the pipe wall if the velocity was high as for the experiment with a water surface of 0.45 meters, figure 6.1 and 6.2. At $time = 1.5s$ back flow starts. At around $time = 3s$ there some droplets that are large enough not to be deemed as remaining droplets from a previous experiment. These droplets are large enough not to remain on the wall and flow back. Then the next large droplet is found and so on until the water column propagates up again due to the pressure gradient between the water surface in the tank and the height of the water column.

In the simulations the water column oscillated longer than the experiments after reaching the furthest point downstream. It is thought that the reason for the water column in the simulations to oscillate a relatively long time is due to that there is no model for surface tension between the fluid and the pipe wall in OLGA. The pipe has an inner diameter of 0.016 meters and then the surface tension effect is of importance, see section 2.2.3.

For the transient progress of the experiments versus simulations the correspondence is quite good even for a coarse grid for the case with a water surface of 0.45 meters and the correspondence with regards to time decreases with higher water surface and velocity. This being said, it is important to ignore holdup with very low values downstream. This is thought to be a result of PVT tables and numerical diffusion. Taking the slope of the curve for the experiment with water surface 0.825, the factor between OLGA and the simulations is the inverse of the fraction of the time when the liquid reaches the pipe outlet, $time\ factor = \frac{\Delta t_{exp}}{\Delta t_{OLGA}} = \frac{5.64}{2.24} \approx 2.5$. This factor, when the water has reached the furthest point downstream, gets smaller for the experiments with lower velocities and water that propagates a shorter

distance, i.e. better correspondence for low inlet pressure and velocity. It is thought that the main reason for this is that OLGA does not account for surface tension between the fluid and the wall and possibly that the valve coefficient is wrong.

Even though OLGA predicts faster propagation of liquid in the pipe and consequently higher velocity and momentum, OLGA did not predict that the pipe was flushed in the simulation with a pressure corresponding to a water surface of 0.825 meters, see figure 6.5. In the experiment, this elevation of the water surface was sufficient to flush the pipe. The liquid did, as is illustrated reach the end of the pipe.

7.1.3 Flushing

In order to find the height difference between the water surface in the tank and the inlet to flush the pipe, the elevation of the tank was shifted a considerable number of times around the elevation, 0.825 meters. This was found as the lowest elevation of which it was possible to repeatedly flush the pipe. Surprisingly, OLGA did not predict flushing with the corresponding pressure although OLGA has in the cases without flushing in the experiments, predicted more filling of the pipe and higher propagation rates. A parametric study of the inlet pressure was carried out in order to find at what inlet pressure OLGA would predict flushing of the pipe. A plot of the case that barely flushed and barely did not flush the pipe is found in figure 6.6. The inlet pressure of the case that flushed the pipe was 1.086875 bara and the inlet pressure for the case that did not flush the pipe was 1.086750 bara. The case that did flush the pipe has the file extension 1.ppl and the one that did not 2.ppl. The corresponding water surface for the two cases are 0.9807 meters and 0.9792 meters respectfully. Consequently, OLGA predicts flushing for a water surface of 0.9792 + a maximum of 1.5 mm. For the case that did not flush, the volumetric flow, Q , is not zero. The velocity however is too slow to push the bubble down the pipe. The average of the volumetric liquid total flow, Q , was found for the last 5 seconds to $2.61766E-05 \text{ m}^3/s$. This corresponds to a velocity of approximately 0.13 m/s. The critical celerity for the pipe in question was found to be 0.2 m/s, see 2.2.3. After 40 seconds the pipe flow has reached a quasi steady state and the median of the velocity is well below the critical celerity preventing the air intrusion to propagate out of the pipe. The quasi steady state is due to that the volumetric flow fluctuates. The highest value of Q the last 5 seconds is about $5.5E-10 \text{ m}^3/s$. This corresponds to a velocity of 0.27 m/s and is above the critical celerity and causes the bubble to be dragged downwards, co current with the flow. The pressure loss due to the "stratification" under the bubble is obviously enough to retard the flow and prevent the bubble from propagating out. The

volumetric flow of the filled pipe is approximately $1.52\text{E-}4 \text{ m}^3/\text{s}$. When comparing the volumetric flow with the figure 2.4, it is seen that the volumetric flow (flux) corresponding to a head of 0.98 meters is about $0.3\text{E-}4$. That is about a factor 2 in discrepancy with the OLGA flow rate prediction. This discrepancy is thought to come mostly from the head loss factors added for the plot of the marginal cases, that are not accounted for in OLGA, see section 2.3. Also, the valve coefficient was set to a constant. An improvement of the marginal cases script should be made to take calculate the valve sizing coefficient as described in section 2.4.

The factor in the head needed to flush the pipe in the experiments and OLGA is $\frac{h_{exp}}{h_{OLGA}} = \frac{0.825}{0.9807} \approx 0.84$. This is a surprisingly good result given the uncertainty regarding the head loss in the valve and the pressure drop in the pipe with multi-phase flow. Since OLGA does not take into account surface tension, that is deemed as the largest contributor to the pressure drop for this case with a small diameter that is not possible to include in the OLGA model, it was expected that OLGA would predict a lower head needed to flush the pipe than in the experiments. The result from this analysis is that OLGA over predicted the head needed. Two reasons for this result is suspected. First that OLGA under predicts the gas lift effect that occurs due to terrain slugging. Second that the pressure drop in the valve is lower in the experiments than in the simulations. The latter suspicion is a bit difficult to assess since it was not possible to find the discharge coefficient for the valve, that OLGA requires, see section 2.4 and that the pressure drop due to surface tension is unknown.

7.2 Discretized bends vs sharp bends

In figure 6.7 it is seen that the liquid propagates faster in the pipe with discretized bends than the pipe with sharp bends. This is seen from that the position of the first liquid particle in the pipe with discretized bends lies to the left indicating a quicker propagation. The velocity profile confirms this as the velocity profile for the pipe with discretized bends lies to the left and is slightly higher than the pipe with sharp bends. When the liquid reaches the end of the pipe, the calculated velocity profile drops to zero. This is due to that the velocity is calculated from the the change of x- and y-coordinate of the last position of the liquid with time. When the liquid has reached the end of the pipe, the change in position is zero and thus the velocity is calculated as zero. Using volumetric flux or velocity parameters from OLGA produced unrealistic results and was disregarded. The difference in the results from using discretized bends or sharp bends is very small for this simulation. It is thought that it would be corresponding results for a simulations with lower pressure. Since OLGA produces instabilities with lower pressure for

both a configuration with discretized bends and sharp bends this has not been possible to investigate. However, the comparison presented figure 6.7 strongly indicates that discretization of the bends is unnecessary, especially when other sources of discrepancy with experimental data are as large as described in section 7.1.

7.3 Coarse vs fine grid

To investigate the influence of using a coarse vs a fine grid the simulations plotted in figure 6.8 was carried out. Due to the effect of smearing out of results and numerical diffusion (see section 4.5.2) it was thought that the results from the coarse grid would have a lower resolution. That assumption was correct. Investigating the position and velocity profile of the fine and coarse grid in figure 6.8 one should notice the distinguished steps in position in the coarse grid vs the fine grid. The result on the velocity profile for the coarse grid is that it jumps from zero to a high value for every step. In the fine grid this is less prominent. A refining of the grid eliminates stepping of the front propagation and oscillations of the velocity. However, due to the configuration and flow phenomena described in 7.1 there are at times sudden changes of velocity and holdup. Thus, smooth graphs is not unconditionally a correct result.

Using than approximately 25 sections lead to instability and failure to simulate the case in OLGA. The object of this project was to investigate to what degree OLGA is capable of predicting whether or not the pipe would be flushed. In this scope the essential outcome of the simulations are the end state, to what degree OLGA predicts the same end state as in the experiments. Investigating the propagation of the water in figure 6.8 there is relatively little discrepancy between using a fine or a coarse grid. The discrepancy for using a finer grid lies on the conservative side. When flushing a real pipeline, it is important to know that the pipeline will actually be flushed. It is deemed conservative that lower velocity and slower filling of the pipe is predicted as this indicates that a higher pressure is needed to flush the pipeline.

7.4 Parametric study of C_d

As mentioned in section 2.4 the valve coefficient, C_d is used in OLGA to calculate the pressure drop over the valve. There has not been found any method for converting C_d to C_v for an arbitrary flow rate. To get some

understanding of the influence and sensitivity of the OLGA results due to C_d a parametric study has been performed.

The parametric study is displayed in figure 6.9 and 6.10. In figure 6.9 the fluid is propagating with relatively high velocity. It is evident that the case with the highest C_d , $C_d = 0.9$ propagates quickest, however not much quicker than $C_d = 0.8$ and correspondingly the case with the lowest C_d propagates slowest. $C_d = 0$ is omitted as this will cause infinite pressure drop over the valve resulting in zero flow see section 2.4 for theory. $C_d = 1$ was also omitted since this case caused numerical instability. In figure 6.10 it is seen that $C_d \geq 0.4$ has approximately the same end state when investigating the holdup profile. It is tempting to choose a C_d for the simulations that reproduces an end state that matches the experiments and probably matches the experiments better. However, it has been argued in section 7.1 that most of the discrepancy between the simulations and the experiments is due to that OLGA does not take into account surface tension. It is the authors opinion that the simulations should be ran with the assumed input from an operator, not to tune the simulation output to experimental results. Thus it has been chosen to run the simulations with the default $C_d = 0.86$.

7.5 Study of velocity profile in OLGA

In order to compare velocity from the experiments and the simulations it was thought that the velocity to the corresponding control volume that was deemed as liquid by setting a low number for the holdup, see section 5.4 would be reasonable to compare with the experimental velocity calculations, see equation 5.1. However the OLGA results were very strange and thus a study of the velocity profile was carried out. It was found that the velocity profile in OLGA was unrealistic. The holdup, geometry, velocity and volumetric flux for the water is presented in figure 6.11 after 0.04 seconds. In this figure, there is a velocity component for the water although there is no water present downstream. An extremely small number (E-16 for holdup or E-20 for total liquid flux) indicates equal to zero. Using computers, the value exact 0 is never found as there is some deviation on a very low scale. Thus an extremely small number compared with other results should not uncritically be deemed as "not zero". The results in figure 6.11 are not problematic since finding the last liquid node may (and has been) correlated with other parameters, such as holdup. Then one may state that when holdup is smaller than a reasonable number, liquid is deemed as zero. Even doing this, problems arise when results as in figure 6.12 arises. Here OLGA produces a large velocity for the liquid when the volumetric flux is extremely small. Also holdup in these sections is not extremely small. When velocity from the experiments was compared to the velocity calculated by OLGA

corresponding to the last liquid node (holdup > 0.1), the OLGA velocities would be very strange fluctuating extensively between large negative and positive numbers in time. Correlating holdup to volumetric flux, Q , also produced unreasonable results. It was deemed as of no value for comparison with the experimental output. Instead, the velocity for the OLGA output was calculated as the velocity from the experiments calculating the change in position vs time.

Chapter 8

Conclusions

A small scale experimental set-up with two peaks and two valleys has been prepared in the laboratory to carry out experiments on flushing of pipelines. It was possible to flush the pipeline in the experimental set-up and the experiments were repeatable. By keeping the same elevation and level of water in the tank, experiments with the same outcome were produced. The rig had three important modifications. First the hand operated valve was changed to a magnetic valve. The magnetic valve has a short stroke time and the variation in stroke time is assumed neglectable. Using a magnetic valve instead of a hand operated valve ensured repeatability of the experiments. Second, a vent was set at the outlet to avoid siphoning. Thirdly, a black sheet was mounted behind the rig. This created good contrast to the fluorescent water and was critical to get good performance of the video analysis.

One camera was used to record the experiments. This was a high definition camera with a frame rate of 25 frames per second. For the experiments carried out, this was sufficient. The resolution of the video analysis was much better than the resolution in the experiments. In performing experiments with higher velocity, it is strongly recommended to use a camera with a higher frame rate in order to have a smaller propagation of the liquid from frame to frame. If careful investigation of the front is to be performed, it is recommended to have more than one camera. This is to have a better resolution. A camera with a shorter distance to the rig, will produce more pixels that is capturing the water propagation and front. The camera was carefully set up for the experiments. A leveling device was mounted on the camera tripod in order to have the camera level.

A script was developed in Matlab for image processing. The work to develop the image analysis script was extensive. One major part was to set up a good color filter. The other was to filter out droplets of water that remained from the previous experiment. This script has been tweaked to catch the

first area in the pipe with a continued liquid column. When the flow is uphill this is the same as the liquid front due to counter current gravity effects. In the downhill pipes, the program tracks the front of the thin film of water that propagates rapidly due to co current gravity. When printing out images using this script, it is confirmed visually that the script is catching the propagation of the water column. The assumptions and methods in the script has been documented thoroughly to facilitate further work on this script and modifications to be used with other experiments.

A model of the experimental set-up has been developed in OLGA. Due to numerical issues, some simplifications had to be made. Two phenomena that is thought to have an important influence on the outcome of the simulations have been discussed. These are the valve discharge coefficient and the surface tension between the fluid and the wall. The OLGA program does not have a model to calculate the pressure drop due to surface tension between the fluid and the wall. For small diameter pipes, as in the rig, this has been confirmed by extensive research to have a significant effect. Through parametric study of the valve coefficient, the influence of this parameter has been investigated. It is recommended that the magnetic valve in the rig is replaced with a magnetic valve for which the discharge coefficient is known or that experiments are carried out to find the discharge coefficient of the valve that is used. The effect of surface tension has not been found. This is due to the uncertainty of the pressure drop due to the discharge coefficient. However, it is strongly suspected that the surface tension is the main reason for a large part of the discrepancy in the propagation rate of the simulations and experiments. For the experiment that flushed the pipe, the OLGA simulation predicted that the water would reach the end of the pipe by a factor 2.5. This factor decreases with inlet pressure and velocity of the water.

In the experiments and simulations that did not flush the pipe, there was a surprisingly good correspondence in the end state. The end state was quantified by measuring the vertical height of the liquid columns in the pipes where liquid was present. Corresponding heights was found by studying holdup in the OLGA output.

The simulations in OLGA to predict flushing has been found to be conservative. Surprisingly, OLGA did not predict that the pipe would be flushed with the equivalent inlet pressure as to the experiment in which the pipe was flushed. This was remarkable, since the propagation rate in the OLGA simulation was higher than in the experiment by a factor 2.5. Also, surface tension was thought to cause the inlet pressure needed to flush the pipe in the experiments to be higher. The factor between the head needed in the experiments and OLGA to flush the pipeline was found to be 0.84. Two possible reasons for this outcome has been discussed. First, the discharge coefficient in the valve might be too high, creating a larger pressure drop

in the simulations than in the experiments. The higher propagation rate in OLGA contradicts this, when one does not take into account that OLGA does not have a model for the pressure drop due to surface tension. The other reason is that OLGA does to a small extent take into account the flow history. In effect, slugs are killed at the end of an upwards or downwards pipe if OLGA has predicted the flow regime in the pipe to be stratified. In the experiments, siphoning is important as the water column displacing all the gas propagates in the downwards pipes. Also, the gas lift effect due to terrain slugging in the upwards pipes significantly reduces the weight of the water column and causes the water to flush the pipe at surprisingly a surprisingly low head, 0.825 meters.

Due to stability considerations, the grid in the OLGA simulations had to be very coarse. Through a comparison with a fine grid and a coarse grid, it is found that the coarse grid has a good performance since the discrepancy between the two grids is very small. Also, the bends in OLGA cannot be modeled. A program to model the bend with discrete straight sections. The discrepancy between a model with discrete sections in the bend and a model with sharp bends has been found to be insignificant.

For future reference a list of further work is presented:

1. Video analysis to find liquid front in the downhill pipes, $H \approx 1$
2. Post processing to find liquid front in the downhill pipes in OLGA, $H \approx 1$
3. Experiments with larger pipe diameter.
4. Smaller angles
5. Varying peak heights
6. Assess valve coefficient and perform new simulations to find discrepancy due to surface tension

Due to limitations in time and very large uncertainty of the outcome, the image analysis script was set up to catch the front of the continuous water column. In the uphill pipes this coincides with the water front since gravity is counter current. In order to find the liquid front in the downhill pipes, a method has to be developed to find the furthest downstream position in the pipe where holdup is very close to 1. A correlation of the pixel values and holdup must be found. This may be done by investigating pixel values over a cross section of the pipe and correlating it to output from a multiphase meter. This is assumed to be an extensive project and has not been considered a part of this thesis. Finding the corresponding holdup in OLGA output is straightforward by using the post processing script that has been developed.

Experiments with an undulating pipeline with a larger pipe diameter should be carried out. Using a larger pipe diameter, e.g. >0.2 meters, surface tension effects between the wall and the fluid are small. Also large pipe diameters is used in the oil and gas industry for transportation of hydrocarbons.

The angles between the straight sections is quite large and in a real case there are usually considerably smaller changes in inclination between two pipe sections. Experiments and simulations should be carried out to investigate the performance of OLGA.

It is more realistic to have a case with varying elevation of the peaks and valleys. This is also thought to produce a wider range of flow phenomena and will be an interesting case for OLGA to simulate.

A valve with known discharge coefficient should be used in order to assess the influence of surface tension in a small diameter pipe, that OLGA does not account for.

Bibliography

- [1] K. Bendiksen. An experimental investigation of the motion of long bubbles in inclined tubes. *Int. J. Mult. Flow*, 10(4):467–483, 1984.
- [2] Member ASCE Chyr Pyng Liou and Member ASCE William A. Hunt. Filling of pipelines with undulating elevation profiles. *Journal of hydraulic engineering*, 122(10):534–539, October 1996.
- [3] Monika Johansen. *An experimental study of the bubble propagation velocity in 3-phase slug flow*. Tapir Uttrykk, 2006.
- [4] Nydal O.J. Johansen M. Bubble drift velocity in horizontal liquid/liquid systems. Submitted to *Int. J. Mult. Flow*.
- [5] O.J. Nydal. Experiments in downwards flow on stability of slug fronts. *Third International Conference on Multiphase Flow, ICMF'98*, June 8-12 1998. Lyon, France.
- [6] Ole Jørgen Nydal. Lecture notes for multi-phase flow. Handouts, 2008.
- [7] G. Das P.K. Das T.K. Mandal, M.K. Bhuyan. Effect of undulation on a gas-liquid two-phase flow through a horizontal pipeline. *Chemical Engineering Research and Design*, 86(3):269–278, 2008.
- [8] J. Townson. *Free-surface hydraulics*, pages 206–208. Unwin Hyman, 1st edition, 1991.
- [9] H K Versteeg and W Malalasekera. *An Introduction to Computational Fluid Dynamics The Finite Volume Method*. Pearson, 2007.
- [10] Frank M. White. *Fluid Mechanics, fifth edition*. McGraw-Hill, 2003.
- [11] E. E. Zukoski. Influence of viscosity, surface tension, and inclination angle on motion of long bubbles in closed tubes. *J. Fluid. Mech.*, 25(4):821–837, 1966. Submitted to *Int. J. Mult. Flow*.

Appendix A

Matlab script for the marginal cases

```
%Script to calculate marginal cases for the undulating pipeline used in
%experiments for the master thesis of Andreas N Winnem spring 2009. This is
%an extension of the original script that plots dp and corresponding head
%for a range of volume fluxes

clear all;
clc;
clf;

%Input parameters

%volume flux – only input parameter to be set by the user with the given
%experimental set-up in the laboratory
Q_min = 1e-6;
Q_max = 1e-3;
Q_diff = Q_max - Q_min;
%number of points to plot
n = 100;
Q_step = Q_diff/(n-1);
%vector of the volume fluxes
Q = (Q_min:Q_step:Q_max);
%gravtitational acceleration
g = 9.81;
%density of water
rho = 998.2071;
%kinematic viscosity of water
mu = 0.001;
%vertical height at peaks and bottoms between straight pipe segments
z = [0 0.7 0.08 0.7 0.075 0.708];
%length of straight pipesegments
l = [0.91 0.832 0.83 0.83 0.832];
%length of bend
```

```

bend = 0.0155;
%feed pipe diameter
D = 0.04;
%Length feed pipe
L = 3;
%pipe diameter
d = 0.02;
%feedpipe roughness
epsilonD = 1e-5;
%pipe roughness
epsilonD = 1e-5;

%head loss coefficients
Kelbow90 = 0.72;
Kelbow45 = 0.21;
Ksc = 0.105;
Kvalve = 6.9;
Kinlet = 0.5;
Koutlet = 1;

%declaring a velocity vector and vector for reynoldsnumbers for the pipe
%with large and small diameter
uD = zeros(1,n);
ud = uD;
ReD = uD;
Red = uD;

%calculating the flow regime in the pipes

%velocity in the feed pipe:
uD(1,:) = 4*Q(1,)/(pi*D^2);
%Reynoldsnumber in the feed pipe:
ReD(1,:) = (rho*uD(1,)*D)/mu;
%velocity in the pipe:
ud(1,:) = 4*Q(1,)/(pi*d^2);
%Reynoldsnumber in the feed pipe:
Red(1,:) = (rho*ud(1,)*d)/mu;

%declering vectors for the friction factors
fD = zeros(1,n);
fd = fD;

%friction factor based on the reynolds number
%feed pipe:

for i=1:1:n

    if ReD(i)>2300

        %Haaland friction factor
        fD(i) = (-1.8*log(6.9/ReD(i) + ((epsilonD/D)/3.7)^(1.11)))^(-2);

    else

```

```

        %Poisuille friction factor
        fd(i) = 64 / ReD(i);

    end

    %pipe:
    if Red(i)>2300

        fd(i) = (-1.8*log(6.9/Red(i) + ((epsilond/d)/3.7)^(1.11)))^(-2);

    else

        fd(i) = 64 / Red(i);

    end

end

%head loss for a filled pipe

%summing up head loss factors
KD = Kinlet + Kvalve;
Kd = Ksc + 4*Kelbow90 + Kelbow45 + Koutlet;

%initial calculations
Δ_z = z(length(z))- z(1);
pipelength = sum(l);
Afeed = pi*D^2/4;
Apipe = pi*d^2/4;

%declaring vector for dp_filled
dp_filled = zeros(1,n);

%calculating the head loss in a filled pipe
dp_filled(1,:) = -rho*g*( Δ_z + ((Q(1,:).^2)/(2*g)).*...
    ( ( fd(1,:)*L/D + KD )/Afeed^2 +...
    ( fd(1,:)*pipelength/d + Kd )/Apipe^2 ) );

%declaring vector for the hydrostatic head
head_filled = zeros(1,n);

%corresponding hydrostatic head
head_filled(1,:) = -dp_filled(1,:)./(rho*g);

%head loss for a pipe with stratified flow in the downhill sections

%initial calculations
Kd = Ksc + 5*Kelbow45 + Koutlet;
sum_z = z(2) + z(4) + z(6);

%declaring vector for dp_stratified
dp_stratified = zeros(1,n);

```

```

%calculating the head loss in a pipe with stratified flow in the downward
%pipes

dp_stratified(1,:) = -rho*g*( sum_z + ((Q(1,:).^2)/(2*g)).*...
    ( ( fd(1,:)*L/D + KD )/Afeed^2 +...
    ( fd(1,:)*pipelength/d + Kd )/Apipe^2 ) );

%declaring vector for the hydrostatic head
head_stratified = zeros(1,n);
head_stratified(1,:) = -dp_stratified(1,:)/(rho*g);

%plotting
figure(1)
hold on

plot(Q,head_stratified, 'r')
plot(Q,head_filled, 'b')

legend('Stratified downhill', ...
    'Filled pipe', ...
    'Location', 'BestOutside')

XLABEL('Volumetric flux, Q[m^3/s]')
YLABEL('Head, h [m]')
TITLE('Head vs volumetric flux')

hold off

```


Appendix B

Matlab script for discretization of the pipe

```
%Script to discretize a curved bend into straight sections and storing the
%nodes as coordinates for OLGA input. This is an extension of the simpler
%script bend that only discretizes a bend.
```

```
clc;
clear all;
clf;

%input
%number of nodes to discretize the bend, use even numbers to reduce error
n = 4;
%bend radius
r = 0.122831704;
%arch length
s = 0.155100638;
%depth of arc in bend
arc_depth = 0.023678401;
%tangent angle with horizontal of straight pipe (radians)
theta_straight = 0.631354254;
%angle of curve section
phi = 2*theta_straight;
%vertical height at peaks and bottoms between straight pipe segments
z = [0.7 0.08 0.7 0.075 0.708];
%length of straight pipesegments
l = [0.91 0.832 0.83 0.83 0.832];

%calculating the length of bend section (one ghost node in each end)
 $\Delta_s = s/(n+2);$ 

%initializing vector for the bend coordinates (x,y)
bend_coordinate = zeros(n,2)';
```

```

%need a vector for the bends in the valleys
bend_valley = bend_coordinate;

%initializing temporary angles to be used in the discretized equation
% theta_0 = theta_straight;
% theta_1 = theta_0;
theta_0 = theta_straight;

%initializing the vector for the angles for each pipe segment
theta = zeros(n+2,1);

%initial calculation
theta(1) = theta_straight;
theta(2) = theta_0 - 2*Δ_s/r;

%calculating the angles
for i = 2:1:n+1

    %second order central difference
    theta(i+1) = theta(i-1) - 2*Δ_s/r;

end

%calculating x,y coordinates
for i = 2:1:n+1

    bend_coordinate(1,i) = bend_coordinate(1,i-1) + Δ_s*cos(theta(i));
    bend_valley(1,i) = bend_coordinate(1,i-1) + Δ_s*cos(theta(i));
    bend_coordinate(2,i) = bend_coordinate(2,i-1) + Δ_s*sin(theta(i));
    bend_valley(2,i) = -(bend_coordinate(2,i-1) + Δ_s*sin(theta(i)));

end

bend_coordinate = bend_coordinate';
bend_valley = bend_valley';

%writing coordinates to file
filename = ['bend',num2str(n)];
save(filename, 'bend_coordinate','-ascii','-tabs')

%calculating input parameters to OLGA

%initializing coordinate vector
xy_OLGA = zeros(1,2);

%elevation profile of straight sections of the pipe including the bends
%length of elevation vector z
k = length(z);
y_straight_prof = zeros(1,k);
y_straight = zeros(1,k);

y_straight(1) = z(1) - arc_depth;

for i = 2:1:k

```

```

    %elevation profile
    y_temp = z(i) - z(i-1);
    y_straight(i) = z(i) - z(i-1) - sign(y_temp)*arc_depth;

end

%elevation profile of the straight sections excluding the bends
y_straight_prof(1) = z(1) - arc_depth;
y_straight_prof(2:k) = z(2:k)-....
    sign(y_straight(2:k)).*arc_depth;

%calculating the horizontal length of the straight sections
x_straight = zeros(1,k);
x_straight(1,1) = sqrt(l(1,1).^2-...
    (y_straight(1,1)).^2);
x_straight(1,2:k) = sqrt(l(1,2:k).^2-...
    (y_straight(1,2:k)).^2);

%combining calculations for straight pipe sections and the discretized
%bends
bend_start = [0 0];
bend_temp = zeros(size(bend_coordinate));
bend_length_x = bend_coordinate(end,1);
x_length = 0;
y_length = 0;
slope = 0;

for i = 1:1:k-1

    slope = sign(y_straight(i));
    bend_start = [(x_straight(i) + x_length) y_straight_prof(i)];
    if slope > 0
        bend_temp(:,1) = bend_coordinate(:,1) + bend_start(1);
        bend_temp(:,2) = bend_coordinate(:,2) + bend_start(2);
    else
        bend_temp(:,1) = bend_valley(:,1) + bend_start(1);
        bend_temp(:,2) = bend_valley(:,2) + bend_start(2);
    end
    xy_OLGA = [xy_OLGA; bend_temp];
    x_length = x_length + x_straight(i) + bend_length_x;

end

%have to reset bend_temp since it will be shorter than above and otherwise
%contain old values
bend_temp = [0 0];

%special treatment of the last bend since it is only half a bend.
slope = sign(y_straight(k));
bend_start = [(x_straight(k) + x_length) y_straight_prof(k)];
bend_temp(1:round(n/2+1),1) = bend_coordinate(1:round(n/2+1),1) + bend_start(1);
bend_temp(1:round(n/2+1),2) = bend_coordinate(1:round(n/2+1),2) + bend_start(2);
xy_OLGA = [xy_OLGA; bend_temp];

```

```

%need to add an extra row of zeros for OLGA to understand that [0 0] is the
%starting point
xy_OLGA = [0 0;xy_OLGA];

%writing the OLGA input to file
filename = ['pipe',num2str(n)];
save(filename, 'xy_OLGA', '-ascii', '-tabs')

%plotting
figure(1)
hold on

plot(bend_coordinate(:,1),bend_coordinate(:,2), 'b')

legend('Discretized bend', ...
       'Location', 'BestOutside')

XLABEL('Horizontal length [m]')
YLABEL('Vertical height [m]')
TITLE('Discretized bend')

hold off

figure(2)
hold on

plot(xy_OLGA(:,1),xy_OLGA(:,2), 'b')

legend('Pipeline', ...
       'Location', 'BestOutside')

XLABEL('Horizontal length [m]')
YLABEL('Vertical length [m]')
TITLE('Discretized pipeline')

hold off

```

Appendix C

Matlab script for video analysis

```
%this script takes in a file containing an avi movie and calculates some
%important fluid parameters from image processing of the video frames.
%
%Input: Video file, number of frames (range) that are to be evaluated,
%color to search for – the fluid is assumed to be fluoridized and a factor
%to determine the range/ specter of colors deviating from the colorant that
%are deemed to be evidence that the color (fluid) is present.
%
%Output: Position of waterfront and velocity of the waterfront.
%
%Options:
%1) Choose to extract filtered and unfiltered images (front_data = 0) or
%the coordinate of the last liquid pixel downstream (front_data = 1).
%frontdata_range_...txt will be the filename of the output.
%2) Set the number of frames to be analyzed. firstframe is the first frame
%to analyze and lastframe the last. If firstframe is set to 1 the program
%will analyze from 2 since this frame is used as background. If firstframe
%and lastframe are set to 0, the program analyzes all frames.
%3) Setting the area of the frame that is to be analyzed to avoid analyzing
%area of the frame where the pipe is not present. Produces a rectangle that
%is being analyzed.
%4) Set the number of subsequent pixels that are to be checked vs the last
%pixel that is deemed as liquid and the tolerance in number of pixels in x
%and y direction between the last liquid pixel and the first. Eg
%check_length = 5 stores the last 5 liquid pixels. pix_tol = 10 gives a
%maximum of 10 pixel difference between the latest liquid pixel and the 5th
%pixel that was deemed as liquid before that.
%5) Set the size of the area that is being analyzed for the frames after
%the first frame. The middle of the box is at the previous liquid front.

clc;
clear all;

%name of video file
```

```

file = 'C:\Andreas\Master\Experiments\ws_45.avi';

%store front data (front_data = 1) or print out filtered images
%(front_data=0)
front_data = 1;

%set number of pixels that are to be checked to avoid finding droplets
check_length = 5;

%set maximum length in x and y direction between last and first pixel that
%is checked
pix_tol = 10;

%set length of box (pixels) that is to be analyzed after the first front
%position is found
box_length = 300;

%set fraction of coordinates in x and y direction of the video that is to
%be analyzed (to avoid analyzing pixels of a frame where the pipe is not
%present). Will produce a rectangle in which the frame is analyzed. Must be
%between 0 and 1, end must be larger than start, cannot be zero.
x_start = 0.01;
x_end = 0.97;
y_start = 0.30;
y_end = 0.75;

area = [x_start,x_end;y_start,y_end];

%number/ range of frames start - firstframe and end - lastframe, for
%manually selecting the frames to be analyzed
firstframe = 2;
lastframe = 3;
frames = [firstframe lastframe];

%converting frames to matrices that are to be manipulated
f2m(file,frames,front_data,area,check_length,pix_tol,box_length);

%function for reading frames from an avi movie and converting them to a
%matrix which is n by m by 3 that is to be analyzed with the function
%liq_position
function frame2matrix = f2m(file,frames,front_data,area,...
    check_length,pix_tol,box_length)

%loading the movieinfo
movieinfo = aviinfo(file);

%calculating the inverse of the framerate; seconds per frame
spf = 1/movieinfo.FramesPerSecond;

%declaring the range of frames that are to be manipulated. Default is to
%analyze all frames, when frames = 0 (no manual input)

```

```

if (frames == 0)

    %subtracting 1 since the first frame is used as background
    range = movieinfo.NumFrames - 1;
    start = 1;

else

    range = frames(2) - frames(1) + 1;
    %subtracting 1 to find the background as the latest image before
    %frame(1)

    if (frames(1) > 2)

        start = frames(1) - 1;

    else

        %start cannot be less than 1
        start = 1;
        range = range - 1;

    end

end

end

%declaring a matrix for storing frontposition and timestamp
fronttime = zeros(range,3);

%finding the background that is to be subtracted from the subsequent images
%returns an array MOV from the movie file 'file' containing the frames
%'frames'
MOV = AVIREAD(file,1);

%returns a matrix with the frame on the format n by m by 3
background = frame2im(MOV);

%initializing progress bar
h = waitbar(0, 'Progress');

% global prev_position;
prev_position = [0 0];

%initializing vector for droplets. coordinate (1,1) is a dummy in case
%there are no droplets found. (1,1) is assumed to be outside of the pixel
%region of the pipe
droplet = [1 1];

%writing a loop to extract the frames within the range
for j=1:1:range

    %displaying the progress
    waitbar(j/range)

```

```

%declaring the frame to be evaluated
framenumber = start + j;

%returns an array MOV from the movie file 'file' containing the frames
%'frames'
MOV = AVIREAD(file,framenumber);

%returns a matrix with the frame on the format n by m by 3
image = frame2im(MOV);

if (front_data == 0)

    %Declaring an individual frame name
    filename = ['Frameorig',num2str(framenumber),'.png'];

    %Writing the frames to file as individual jpg images
    IMWRITE(image,filename,'png')

end

%extracting the background from the current image
image = image - background;

step = j;

if step == 1;

    [frontposition prev_position droplet] = init_position(image,...
        front_data,area,check_length,pix_tol,droplet);

else

    %finding the front position
    [frontposition prev_position droplet] = liq_position(image,...
        front_data,area,check_length,pix_tol,prev_position,...
        box_length,droplet);

end

if (front_data == 0)

    %storing the filtered image
    image = frontposition;

    %Declaring an individual frame name
    filename = ['Frame',num2str(framenumber),'.png'];

    %Writing the frames to file as individual jpg images
    IMWRITE(image,filename,'png')

end

if (front_data == 1)

```



```

        %storing the frontposition with a timestamp
        fronttime(j,1) = framenumbers*spf;
        fronttime(j,2) = frontposition(1);
        fronttime(j,3) = frontposition(2);

    end

end

delete(h)

if front_data == 1

    frame2matrix = fronttime;

    %writing fronttime to file
    filename = ['frontdata_', 'range_', num2str(range), '.txt'];
    save(filename, 'fronttime', '-ascii', '-tabs')

end

%this function finds the front position in the first image that is being
%processed.
%Input: A frame on matrix form from an avi movie
%Output: A position either on (x,y) form or a relative length along the
%pipe, L

function [frontposition prev_position droplet] = init_position(image,...
    front_data, area, check_length, pix_tol, droplet)

%defining the image size
%note that pixels in the y-direction is listed first using size(image)
xmax = size(image,2);
ymax = size(image,1);

%area to be analyzed
xstart = ceil(xmax*area(1,1));
xend = round(xmax*area(1,2));
ystart = ceil(ymax*area(2,1));
yend = round(ymax*area(2,2));

%note that pixels in the y-direction is listed first using size(image)
%the color we are looking for is green, so that is the only color in the
%color vector, k, that is searched for

%green color is in the second place:
k = 2;

% %since white color includes a strong green color (high numeric value), a
% %filter to exclude the white or bright colors except green is needed
% red = 200;

```

```

% green = 150;
% blue = 200;
% black = 80;

%ratio definition between colors to see if that works better to catch more
%green fluid without getting greyscale pixels. Testing indicates that
%ration definition has a limited effect when subtracting the background.
% redgreenmax = 0.80;
% redgreenmin = 1.20;
% bluegreenmax = 0.80;
% bluegreenmin = 1.20;
redgreenmax = 1.0;
redgreenmin = 1.0;
bluegreenmax = 1.0;
bluegreenmin = 1.0;
greenmin = 50;

%setting a background color
background=255;

%setting a coordinate for the waterfront (y,x)
front = [yend,0];

front_check = zeros(check_length,2);

i = xstart;
j = ystart;

%to break off while loop
stop = 0;

while i < xend + 1

    while j < yend + 1

        %calculating the ratio between the color intensities blue and red
        %vs green and storing the green intensity as well
        imred = double(image(j,i,1));
        imgreen = double(image(j,i,k));
        imblue = double(image(j,i,3));
        rgcalc = imred / imgreen;
        bgcalc = imblue / imgreen;

        if ((imgreen > greenmin) && (rgcalc < redgreenmax ||...
            rgcalc>redgreenmin) && (bgcalc < bluegreenmax ||...
            bgcalc > bluegreenmin))
            image(j, i, :) = image(j, i, :);
            front = [j,i];
            front_check = [front_check; front];
            front_check(1,:) = [];

%           if ((image(j,i,k) > green) && (image(j,i,1)< red)...
%               && (image(j,i,3)<blue))
%                   image(j, i, :) = image(j, i, :);

```

```

else
    image(j, i, :) = background;
end

if (j == yend && i == xend &&...
    ((front_check(end,2) - front_check(1,2)) > pix_tol) ||...
    ((front_check(end,1) - front_check(1,1)) > pix_tol))

    for n = 1:1:check_length-1

        diff = front_check(n+1,:) - front_check(n,:) + n - 1;

        if diff(1)>1 || diff(2)>1
            image(front_check(n+1,1), front_check(n+1,2), :) = ...
                background;
            droplet = [droplet; front_check(n+1,1), ...
                front_check(n+1,2)];
            disp(['droplet found at ' num2str(front_check(2,end))])
            stop = stop + 1;
        end

        end

        j = ystart - 1;
        i = xstart - 1;
        stop = stop + 1;

        front_check = zeros(check_length,2);
        front_check(1:end,1) = prev_position(1);
        front_check(1:end,2) = prev_position(2);

        %         image(front_check(end,1), front_check(end,2), :) = background;
        %         j = ystart - 1;
        %         i = xstart - 1;
        %
        %         disp(['droplet found at ' num2str(front_check(2,end))])
        %         droplet = [droplet; front_check(end,1), front_check(end,2)];

        stop = stop + 1;

    end

    j = j + 1;

end

j = ystart;
i = i + 1;

if stop>20
    disp('Warning: loop broken')
    break
end

```

```

end

if (front_data == 0)

    %writing a crosshair at the image where the front is (for testing
    %purposes)
    cross = 10;

    if (front(:) ≠ 0)
        cross_x = front(2);
        cross_y = front(1);

        for i = 1:1:cross

            image(cross_y , i + cross_x - round(cross/2),:) = 0;

        end

        for j = 1:1:cross

            image(j + cross_y - round(cross/2),cross_x,:) = 0;

        end

    end

    %testing: printing the images with a crosshair on the waterfront
    frontposition = image;

    prev_position = front;

end

if (front_data == 1)

    %returning the front position in (y,x) coordinates
    frontposition = front;

    prev_position = front;

end

%this function finds the front position in the first image that is being
%processed.
%Input: A frame on matrix form from an avi movie
%Output: A position either on (x,y) form or a relative length along the
%pipe, L

function [frontposition prev_position droplet] = liq_position(image,...
    front_data,area,check_length,pix_tol,prev_position,box_length,droplet)

```

```

%defining the image size
%note that pixels in the y-direction is listed first using size(image)
xmax = size(image,2);
ymax = size(image,1);

% %area to be analyzed
% xstart = ceil(xmax*area(1,1));
% xend = round(xmax*area(1,2));
% ystart = ceil(ymax*area(2,1));
% yend = round(ymax*area(2,2));

% box_length = 100;

%shifting parameter for shifting the box to the right (downstream)
shift = 0.5;
shift_length = round(box_length*shift/2);

%finding current box to analyze
xstart = prev_position(2) - round(box_length/2) + shift_length;
xend = prev_position(2) + round(box_length/2) + shift_length;
ystart = prev_position(1) - round(box_length/2);
yend = prev_position(1) + round(box_length/2);

%checking that the box does not exceed the predefined area to analyze
if xstart < ceil(xmax*area(1,1))
    xstart = ceil(xmax*area(1,1));
    xend = xstart + box_length;
end

if xend > round(xmax*area(1,2))
    xend = round(xmax*area(1,2));
    xstart = xend - box_length;
end

if ystart < ceil(ymax*area(2,1))
    ystart = ceil(ymax*area(2,1));
    yend = ystart + box_length;
end

if yend > round(ymax*area(2,2))
    yend = round(ymax*area(2,2));
    ystart = yend - box_length;
end

%note that pixels in the y-direction is listed first using size(image)
%the color we are looking for is green, so that is the only color in the
%color vector, k, that is searched for

%green color is in the second place:
k = 2;

% %since white color includes a strong green color (high numeric value), a
% %filter to exclude the white or bright colors except green is needed
% red = 200;

```

```

% green = 150;
% blue = 200;
% black = 80;

%ratio definition between colors to see if that works better to catch more
%green fluid without getting greyscale pixels. Testing indicates that
%ration definition has a limited effect when subtracting the background.
% redgreenmax = 0.90;
% redgreenmin = 1.10;
% bluegreenmax = 0.90;
% bluegreenmin = 1.10;
redgreenmax = 1.0;
redgreenmin = 1.0;
bluegreenmax = 1.0;
bluegreenmin = 1.0;
greenmin = 50;

%setting a background color
background=255;

%setting a coordinate for the waterfront (x,y)
front = prev_position;

%vector for storing subsequent pixels that are liquid
front_check = zeros(check_length,2);
front_check(1:end,1) = front(1);
front_check(1:end,2) = front(2);

%eliminating earlier founds of droplets
length_droplet = size(droplet,1);

for i=1:1:length_droplet

    y = droplet(i,1);
    x = droplet(i,2);
    image(y,x,:) = background;

end

i = xstart;
j = ystart;

%to be used if while loop runs too many times
stop = 0;

while i < xend + 1

    while j < yend + 1

        %calculating the ratio between the color intensities blue and red
        %vs green and storing the green intensity as well
        imred = double(image(j,i,1));
        imgreen = double(image(j,i,k));
        imblue = double(image(j,i,3));

```

```

rgcalc = imred / imgreen;
bgcalc = imblue / imgreen;

if ((imgreen > greenmin) && (rgcalc < redgreenmax ||...
    rgcalc>redgreenmin) && (bgcalc < bluegreenmax ||...
    bgcalc > bluegreenmin))
    image(j, i, :) = image(j, i, :);
    front = [j,i];
    front_check = [front_check; front];
    front_check(1,:) = [];

%     if ((image(j,i,k) > green) && (image(j,i,1)< red)...
%         && (image(j,i,3)<blue))
%         image(j, i, :) = image(j, i, :);
else
    image(j, i, :) = background;

end

if (j == yend && i == xend &&...
    (((front_check(end,2) - front_check(1,2)) > pix_tol) ||...
    ((front_check(end,1) - front_check(1,1)) > pix_tol))

    for n = 1:1:check_length-1

        diff = front_check(n+1,:) - front_check(n,:) + n - 1;

        if diff(1)>1 || diff(2)>1
            image(front_check(n+1,1),front_check(n+1,2),:) = ...
                background;
            droplet = [droplet;front_check(n+1,1),...
                front_check(n+1,2)];
            disp(['droplet found at ' num2str(front_check(2,end))])
            stop = stop + 1;
        end

    end

    j = ystart - 1;
    i = xstart - 1;
    stop = stop + 1;

    front_check = zeros(check_length,2);
    front_check(1:end,1) = prev_position(1);
    front_check(1:end,2) = prev_position(2);

%     image(front_check(end,1),front_check(end,2),:) = background;
%     j = ystart - 1;
%     i = xstart - 1;
%
%     disp(['droplet found at ' num2str(front_check(end))])
%     droplet = [droplet;front_check(end,1),front_check(end,2)];
%     front_check(end,:) = [];
%

```

```

%             stop = stop + 1;

        end

        j = j + 1;

    end

    j = ystart;
    i = i + 1;

    if stop>20
        disp('Warning: loop broken!')
        break
    end

end

if (front_data == 0)

    %writing a crosshair at the image where the front is (for testing
    %purposes)
    cross = 10;

    if (front(:) ≠ 0)
        cross_x = front(2);
        cross_y = front(1);

        for i = 1:1:cross

            image(cross_y , i + cross_x - round(cross/2),:) = 0;

        end

        for j = 1:1:cross

            image(j + cross_y - round(cross/2),cross_x,:) = 0;

        end

    end

    %testing: printing the images with a crosshair on the waterfront
    frontposition = image;

end

if (front_data == 1)

    %returning the front position in (y,x) coordinates
    frontposition = front;

end

```



```
if ((front(1) == xstart) || (front(1,1) == xend))

    disp('Warning: liquid front was found at the end of the box.')
    disp('Consider calculating with a larger box length')

end

if ((front(2) == ystart) || (front(1,1) == yend))

    disp('Warning: liquid front was found at the end of the box.')
    disp('Consider calculating with a larger box length')

end

if (front(1) == prev_position(1) || front(2) == prev_position(2))

    disp('Warning: liquid front was not found.')

end

%updating previous position
prev_position = front;
```


Appendix D

Matlab script for post processing of the video analysis

```
%this script takes in a file containing the output from the script "main"
%that analyzes the video from experiments conducted for Andreas N Winnem's
%master thesis 2009.
%Input: data file containing timestamp and coordinate of front position
%Output: Coordinate of waterfront with dimension [m] and velocity with
%timestamp.

clc;
clear all;
clf;

%write filename that is going to be processed here
filename = 'frontdata_range_256.txt';

%the number of vertical pixels
y_pix = 1080;

%maximum velocity allowed, input to disregard unreasonably high velocities
%due to the emergency of eg a droplet of remaining water from a previous
%experiment moving downstream, [m/s]
v_max = 10;

%in the case of unreasonable velocity, extrapolate previous velocity to
%calculate position (extrapolate = 1)
extrapolate = 0;

%known distance measured on the rig [m]
rig_scale = 3;

%start of scale on the rig, found by read-off of image in matlab by
%checking the pixel value at the beginning of the scale
x_start = 200;
```

```

%end of scale on the rig, found by read-off of image in matlab by
%checking the pixel value at the beginning of the scale
x_end = 1000;

%scale parameter for the pixels dim pixels/ meter
scale = (x_end - x_start)/rig_scale;

%loading the data from filename into Matlab
file = (filename);
%tyx contains timestamp y-coordinate and x-coordinate of the waterfront in
%the matrix, note that y-coordinates are in the second column
tyx_image = load(file);

%dropping all timestamps with zero change in position
%first finding the non-zero position entry in x and y column
y_first = find(tyx_image(:,2), 1, 'first');
x_first = find(tyx_image(:,3), 1, 'first');

%in the unprobable case that either x or y should be non-zero whilst the
%other remains zero:
first = min(y_first,x_first);

%dropping all but the latest zero entries (deemed as start of experiment)
tyx_image(1:first-1,:) = [];

%have to reset the timestamps for the remaining entries
starttime = tyx_image(1,1);
length_tyx = size(tyx_image,1);
tyx_image(1:length_tyx,1) = tyx_image(1:length_tyx,1) - starttime;

%need to manipulate the y-values since the image has coordinate (1,1) in
%upper left corner of the matrix - want (1,1) to be the lower left corner
tyx_image(:,2) = y_pix - tyx_image(:,2);

%matrix for storing the processed values, time, x-coordinate, y-coordinate
%and velocity
txyv = zeros(length_tyx,4);

%adding the timestamp
txyv(:,1) = tyx_image(:,1);

%calculating physical x_coordinate and adding to txyv, note that
%x-coordinates are stored in the second column
txyv(:,2) = tyx_image(:,3)/scale;

%calculating physical y_coordinate and adding to txyv
txyv(:,3) = tyx_image(:,2)/scale;

%number of rows
n = size(tyx_image,1);

%calculating velocity with the formula  $v = \text{sign}(dx) * \sqrt{dx^2 + dy^2} / dt$ ,
%dimension [m/s]. sign(dx) denotes positive or negative velocity in

```

```

%x-direction which is consistent with positive or negative flow direction
%in the pipe
txyv(2:n,4) = sign(txyv(2:n,2) - txyv(1:n-1,2)).*sqrt((txyv(2:n,2) -...
    txyv(1:n-1,2)).^2 + (txyv(2:n,3) - txyv(1:n-1,3)).^2 )...
    ./ (txyv(2:n,1) - txyv(1:n-1,1));

%storing the original data for plotting purposes
txyv_original = txyv;

%checking for unreasonable velocity
[r,c]=find(abs(txyv)>v_max);

%removing the unreasonable results and recalculating
for i = 1:1:length(r)

    if (extrapolate == 1)

        txyv(r(i),2) = txyv(r(i)-1,2) + txyv(r(i)-1,2) - txyv(r(i)-2,2);
        txyv(r(i),3) = txyv(r(i)-1,3) + txyv(r(i)-1,3) - txyv(r(i)-2,3);
        %velocity is the same as for the previous point
        txyv(r(i),4) = txyv(r(i)-1,4);

    else

        txyv(r(i),2) = txyv(r(i)-1,2);
        txyv(r(i),3) = txyv(r(i)-1,3);
        txyv(r(i),4) = 0;
        %velocity for the next point needs to be updated
        txyv(r(i)+1,4) = sign(txyv(r(i)+1,2) - txyv(r(i),2))...
            .*sqrt((txyv(r(i)+1,2) - txyv(r(i),2)).^2 +...
            (txyv(r(i)+1,3) - txyv(r(i),3)).^2 )...
            ./ (txyv(r(i)+1,1) - txyv(r(i),1));

    end

end

%plotting
plot(txyv(:,2),txyv(:,3))

%writing the output to file
filename = ['txyv_', 'starttime_', num2str(starttime), '.txt'];
save(filename, 'txyv', '-ascii', '-tabs')

%plotting the corrected x,y data

figure(2)

plot(txyv(:,2),txyv(:,3), 'b')

legend('Plot of x vs y', ...
    'Location', 'BestOutside')

```

```
XLABEL('Horizontal length [m]')
YLABEL('Vertical length [m]')
TITLE('Flow with error correction')

%plotting original x,y data, no checking for unreasonable velocities

figure(1)

plot(txyv_original(:,2),txyv_original(:,3), 'r')

legend('Plot of x vs y', ...
       'Location', 'BestOutside')

XLABEL('Horizontal length [m]')
YLABEL('Vertical height [m]')
TITLE('Flow without error correction')
```

Appendix E

Matlab script for post processing of the OLGA output

```
%Script for extracting data from OLGA output. Used for Andreas N Winnem's
%Master Thesis 2009.
%
%Input: Output from "Profile plot" in OLGA. Header "TIME..." for each box
%in the OLGA output is the current time step. Check the desired parameter
%in Profile Plot and write to the file with matching file path as below in
%file. One parameter from OLGA may be processed at a time using this
%script.
%
%Output: The first column is the current time step, the two next is the
%length coordinate, x [m] and the last is the parameter chosen in
%OLGA. If more than one parameter is checked, num_parameter in this script
%has to be changed accordingly. Check the OLGA output for which column
%belongs to which parameter. The output from this script contains no
%headers. A plot to verify the output is produced. This plots the velocity
%vs time if velocity is to be calculated or the x-coordinate of the last
%downstream node containing liquid versus time.
%If velocity is to be calculated:
%Two files are generated. OLGA_full_output_end_time consist of
%simulation data for the whole pipe for all timesteps.
%OLGA_extract_end_time consist of only the last downstream control volume
%with liquid in it.

clc;
clear all;
clf;

file = 'C:\Andreas\Master\Matlab\flush_825\OLGA_geo_h_825_10s.txt';

%the number of parameters exported from OLGA, excluding, time step and
%geometry (y-coordinate), x-coordinate (horizontal length) comes out
%automatically.
```

```

num_param = 1;

%since OLGA output comes with the length along the pipe, length coordinates
%in OLGA must be recalculated. Since the inclination of the pipe segments is
%almost equal, this is done by multiplying with the fraction of the actual
%total length of the rig and the total length of the pipe in OLGA.
rig_length = 3.88;

%set the box where holdup is present
holdup_box_OLGA = 2;
holdup_column = holdup_box_OLGA + 2;

%geo_column_OLGA must be the box geometry is in. In some simulations OLGA
%extracts this to the second box. Pipe diameter is given in pipe_diameter.
geo_box_OLGA = 1;
pipe_diameter = 0.016;
geo_column = geo_box_OLGA + 2;

%minimum value for holdup. Due to numerical reasons, nodes far downstream
%will have a holdup larger than 0 long before the liquid has actually
%propagated to this node, eg Holdup = 10-6. Set min_holdup to mitigate this
min_holdup = 0.1;

%+ 2 = time, x-coordinate
total_param = num_param + 2;

%file identifier, opening the text file
fid = fopen(file);

%variable for the current time step
time = 0;

%matrix for storing the output (volume flow)
time_front_geo = zeros(1,total_param);

%matrix for storing the output (holdup)
time_front_H = zeros(1,total_param);

%counter to find odd or even number a time stamp is found
count_time = 0;

%to count the number of rows for volume flow
geo_counter = 0;

%run until end of input file
while (~feof(fid))

    %getting the current line from the text file
    s = fgetl(fid);

    %checkin for line that includes time step
    if (strfind(s,'TIME')==1)

        count_time = count_time + 1;

```



```

        %reading the time step
        time = timeread(s);

    end

    %scanning for numbers in the line and time_count is odd
    position_value = sscanf(s, '%f', Inf);

    %saving line info if number is present
    if (length(position_value)==total_param-1 && mod(count_time,2)==1)

        time_front_geo = [time_front_geo; time,...
            position_value(1), position_value(2)];

    end

    %saving line info if number is present and count_time is even
    if (length(position_value)==total_param-1 && mod(count_time,2)==0)

        time_front_H = [time_front_H; time,...
            position_value(1), position_value(2)];

    end

end

%dropping the first row of zeros since this was purely to initialize the
%time_front matrix
time_front_geo(1,:) = [];
time_front_H(1,:) = [];

time_start = time_front_geo(1,1);
time_temp = time_start;
row_counter = 0;

%finding the number of rows per timestep
while time_temp == time_start

    time_temp = time_front_geo(row_counter + 1,1);
    row_counter = row_counter + 1;

end

%subtracting 1 since the while loop stops at first position in the next
%time step
row_counter = row_counter - 1;

length_front_time = size(time_front_geo,1);

%number of timesteps, including the start, t = tstart
time_steps = length_front_time/row_counter;

```

```

%OLGA uses staggered grid for computing velocities, ie one extra velocity
%node (row) and interpolation between the holdup nodes is done. Subtracting
%time_count/2 since time_front_H and the interpolated Q is one row less every
%time step

geo = (time_front_geo(1:row_counter-1,3) +...
      time_front_geo(2:row_counter,3))/2;

geo_vector = geo;

for i = 1:1:time_steps - 1

    geo_vector = [geo_vector; geo];

end

%building the time front matrix, time, x-coord, y-coord, holdup
time_front = [time_front_H(1:end,1) time_front_H(1:end,2) geo_vector...
             time_front_H(1:end,3)];

%finding value of the last time step
end_time = time_front(end,1);

%writing the output to file
filename = ['OLGA_full_output', 'end_time_', num2str(end_time), '_s', '.txt'];
save(filename, 'time_front', '-ascii', '-tabs')

%storing the total length along the pipe (to calculate the actual
%horizontal length scale further below)
total_length_OLGA = time_front(end,2);

%this part of the script extracts the parameters of the last downstream
%node where liquid is present. This will be the paramters compared with the
%output from video analysis

time_start = time_front(1,1);
time_temp = time_start;
row_counter = 0;

%finding the number of rows per timestep
while time_temp == time_start

    time_temp = time_front(row_counter + 1,1);
    row_counter = row_counter + 1;

end

%subtracting 1 since the while loop stops at first position in the next
%time step
row_counter = row_counter - 1;

length_front_time = size(time_front,1);

%number of timesteps, including the start, t = tstart

```

```

time_steps = length_front_time/row_counter;

%finding and storing the parameters for the last downstream node where
%liquid is present, note that present liquid is deemed as a non zero entry
%for the fluid variable
row_start = 1;
row_end = row_counter;

OLGA_width = size(time_front,2);
OLGA_extract = zeros(time_steps,OLGA_width);

%checking for discontinuity. OLGA has in some simulations displayed
%accumulation of liquid at the end of the pipe before any liquid had been
%displaced to this section (ie after very short time).
continuity_check = 1;
discontinuity_step = OLGA_width;
old_check = 0;

%tolerance for zero entries between the previous last zero entry and the
%current, eg last = row 10, current = 100 leaves 90 entries with the
%parameter equal to zero. tol < 89 will then give a discontinuity warning
tol = 10;

for i = 1:1:time_steps

    time_prop = time_front(row_start:row_end,:);

    %updating row_start and row_end
    row_start = row_start + row_counter;
    row_end = row_end + row_counter;

    %finding the last liquid node downstream
    non_zero = find(time_prop(:,4), row_counter, 'first');

    node_prop = 0;

    if (length(non_zero) < OLGA_width)

        node_prop = time_prop(1,:);

    else

        for j = 1:1:length(time_prop)

            if time_prop(j,4)>min_holdup

                node_prop = time_prop(j,:);

            end

        end

        if node_prop == 0

```

```

        node_prop = time_prop(1,:);

    end

end

%storing properties for the current time_step
OLGA_extract(i,:) = node_prop(:);

%checking if there is a discontinuity
if (continuity_check == 1)

    for j=1:length(non_zero)-1

        check_temp = non_zero(j+1) - non_zero(j);
        position = time_prop(non_zero(j+1),:);
        diff = check_temp - old_check;

        if (check_temp ≠ 1 && abs(diff) > tol)

            disp(['Discontinuity was found at ' num2str(position)])
            continuity_check = 0;

        end

        old_check = check_temp;

    end

end

end

%recalculating length scale in OLGA output
OLGA_factor = rig_length/total_length_OLGA;
OLGA_extract(:,2) = OLGA_extract(:,2)*OLGA_factor;

%number of rows
n = size(OLGA_extract,1);

%adding extra column for velocity
OLGA_extract = [OLGA_extract zeros(n,1)];

%calculating velocity with the formula v=sign(dx)*sqrt(dx^2 + dy^2)/dt,
%dimension [m/s]. sign(dx) denotes positive or negative velocity in
%x-direction which is consistent with positive or negative flow direction
%in the pipe
OLGA_extract(2:n,end) = sign(OLGA_extract(2:n,2) -...
    OLGA_extract(1:n-1,2))...
    .*sqrt((OLGA_extract(2:n,2) - OLGA_extract(1:n-1,2)).^2 + ...
    (OLGA_extract(2:n,3) - OLGA_extract(1:n-1,3)).^2 )...
    ./ (OLGA_extract(2:n,1) - OLGA_extract(1:n-1,1));

% %Velocity calculation based on the previous and the next position

```

```

% OLGA_extract(2:n-1,end) = sign(OLGA_extract(3:n,2) - OLGA_extract(1:n-2,2))...
%     .*sqrt((OLGA_extract(3:n,2) - OLGA_extract(1:n-2,2)).^2 + ...
%     (OLGA_extract(3:n,3) - OLGA_extract(1:n-2,3)).^2 )...
%     ./ (OLGA_extract(3:n,1) - OLGA_extract(1:n-2,1));

%plotting the x-coordinate if not velocity is calculated
plot_vector = 2;

%writing the output to file
filename = ['OLGA_extract_', 'end_time_', num2str(end_time), '_s', '.txt'];
save(filename, 'OLGA_extract', '-ascii', '-tabs')

%plotting velocity
figure(1)

plot(OLGA_extract(:,1),OLGA_extract(:,5), 'b')

legend('Plot of time vs velocity', ...
       'Location', 'BestOutside')

XLABEL('Time [s]')
YLABEL('Velocity [m/s]')
TITLE('Verification plot')

%plotting the position of the furthest downstream node with holdup larger
%than than min_holdup (deemed as liquid is present)
figure(2)

plot(OLGA_extract(:,1),OLGA_extract(:,2), 'r')

legend('Plot of time vs position', ...
       'Location', 'BestOutside')

XLABEL('Time [s]')
YLABEL('Liquid front [m]')
TITLE('Verification plot')

%identfy number with arbitrary number of decimals within a string and
%converting to number

function time = timeread(str)

% The locations of the numbers:
idx = regexp(str, '\d');
% The numbers themselves:
nums = regexp(str, '\d', 'match');

length_idx = length(idx);

dot_position = 0;

%finding the position of the dot seperator in a number eg 0.001

```

```

for i = 2:1:length_idx
    if (idx(i)-idx(i-1))>1
        dot_position = i;
    end
end

%inserting dot in the number string of characters, each integer in eg 0.001
%is per now a character in an array, eg '0' '0' '0' '1'
nums = [nums(1:dot_position-1) , '.' , nums(dot_position:length_idx)];

%collecting the "characters"
num_collect = char(nums)';

%converting to number
numn = str2double(num_collect);

time = numn;

```

Appendix F

Matlab script for comparing the video analysis and the OLGA output

```
%This script is to compare processed output from OLGAfor Andreas N Winnem's
%master thesis 2009.
%
%Input: Extract from OLGA processed data (OLGA_extract_end_time... files)
%and video analysis data (video_extract_starttime files). Pipe diameter d
%must be set for plotting the latest liquid position vs time.
%
%Output: Comparison of position (x/d) and velocity of last liquid partical
%downstream between simulations (OLGA) and experiments (video)
%
%Options:
%1) To plot the data the vectors to be plotted must be of the same
%length. In order to achieve this with datasets with unequal length there
%is either an option to cut the extra rows in the longest data set or to
%extrapolate the last value of the shortest data set. See variables cut and
%extrapolate.
%2) To plot latest liquid position vs time set position = 1. To
%plot velocity vs time, set position = 0. Note that the end user
%must load the correct OLGA file (containing either velocity or a parameter
%indicating that liquid is present eg holdup.

clear all;
clc;
clf;

d = 0.02;

%loading data from file
OLGA_file_1 = 'C:\Andreas\Master\Matlab\ws_75\OLGA_extract_10_s.txt';
```

```

OLGA_file_2 = 'C:\Andreas\Master\Matlab\ws_75\video_extract_0.52.txt';

OLGA_data_1 = load(OLGA_file_1);
OLGA_data_2 = load(OLGA_file_2);

%set name for parameters from data set 1
name_1 = 'OLGA';
name_2 = 'Experiment';

%set length of x-axis [s]
xmax = 5;
xmin = 0;

%length of y-axis [m]
ymax_m = 4;
ymin_m = 0;

%length of y-axis [m/s]
ymax_ms = 12;
ymin_ms = 0;

%state the column of velocity in dataset 1
velo_column_1 = 5;

%state the column of velocity in dataset 2
velo_column_2 = 4;

%in the case that the size (number of rows) is unequal choose to cut or
%extrapolate, cut = 1 is cutting of rows to have the same length. This will
%cut a number of rows at the end of the longest datase, ie cutting off in
%time.
cut = 1;
extrapolate = 1 - cut;

%need to store the time vector for the longest data set if extrapolation is
%used
time = 0;

%option to choose between position of velocity input/ output
position = 1;
velocity = 1 - position;

%checking the length of data sets
OLGA_length_1 = size(OLGA_data_1,1);
OLGA_length_2 = size(OLGA_data_2,1);
length_diff = abs(OLGA_length_1 - OLGA_length_2);

%cutting
if cut == 1

    if OLGA_length_1 > OLGA_length_2

        cut_start = OLGA_length_1 - length_diff + 1;
        OLGA_data_1(cut_start:OLGA_length_1,:) = [];
    end
end

```



```

else
    cut_start = OLGA_length_2 - length_diff + 1;
    OLGA_data_2(cut_start:OLGA_length_2,:) = [];

end

time = OLGA_data_2(:,1);

end

if extrapolate == 1

    if OLGA_length_1 > OLGA_length_2

        extrapolate_start = OLGA_length_1 - length_diff + 1;
        last_line = OLGA_data_2(end,:);
        width = size(OLGA_data_2,2);
        zeros_matrix = zeros(length_diff,width);
        OLGA_data_2 = [OLGA_data_2; zeros_matrix];
        time = OLGA_data_1(:,1);

        for i=1:1:width

            OLGA_data_2(extrapolate_start:end,i) = last_line(i);

        end

    else

        extrapolate_start = OLGA_length_2 - length_diff + 1;
        last_line = OLGA_data_1(end,:);
        width = size(OLGA_data_1,2);
        zeros_matrix = zeros(length_diff,width);
        OLGA_data_1 = [OLGA_data_1; zeros_matrix];
        time = OLGA_data_2(:,1);

        for i = 1:1:width

            OLGA_data_1(extrapolate_start:end,i) = last_line(i);

        end

    end

end

end

% %writing the output to file
% filename = ['Comparison_',parameter,'.txt'];
% save(filename, 'OLGA_extract','-ascii','-tabs')

```

```

%storing the horizontal position
OLGA_position_1 = OLGA_data_1(:,2);
OLGA_position_2 = OLGA_data_2(:,2);

%storing velocity
OLGA_velocity_1 = OLGA_data_1(:,velo_column_1);
OLGA_velocity_2 = OLGA_data_2(:,velo_column_2);

%plotting all in one figure, left y-axis horizontal length, right y-axis
%velocity, bottom x-axis time

figure(1)

h11 = line(OLGA_data_1(:,1),OLGA_data_1(:,2),'Color','r');
h12 = line(OLGA_data_2(:,1),OLGA_data_2(:,2),'Color','b');
ax1 = gca;
set(ax1,'XColor','k','YColor','k')
xlim([xmin xmax])
ylim([ymin_m ymax_m])

name_1m = [name_1 ' propagation'];
name_2m = [name_2 ' propagation'];

LEGEND(name_1m , name_2m , 'Location', 'East')
LEGEND BOXOFF

ax2 = axes('Position',get(ax1,'Position'),...
           'YAxisLocation','right',...
           'Color','none',...
           'XColor','k','YColor','k');

h13 = line(OLGA_data_1(:,1),OLGA_data_1(:,velo_column_1),...
           'Color','k','Parent',ax2);
h14 = line(OLGA_data_2(:,1),OLGA_data_2(:,velo_column_2),...
           'Color','g','Parent',ax2);

set(get(ax1(1),'Xlabel'),'String','Time [s]')

set(get(ax1(1),'Ylabel'),'String','Horizontal position [m]')
set(get(ax2,'Ylabel'),'String','Velocity [m/s]')

xlim([xmin xmax])
ylim([ymin_ms ymax_ms])

name_1ms = [name_1 ' velocity'];
name_2ms = [name_2 ' velocity'];

LEGEND( name_1ms , name_2ms , 'Location', 'SouthEast')
LEGEND BOXOFF

TITLE('Propagation and velocity of liquid vs time')

```