NTNU – Trondheim
Norwegian University of
Science and Technology

Simulation of Flow Around an Oil Boom

Henning Bjørvik

Marine Technology
Submission date: July 2015
Supervisor: Bjørnar Pettersen, IMT
Co-supervisor: Håvard Holm, IMT

Norwegian University of Science and Technology
Department of Marine Technology

NTNU
Norwegian University of Science and Technology
Department of Marine Technology

MASTER THESIS IN MARINE HYDRODYNAMICS

SPRING 2015

FOR

Stud.techn. Henning Bjørvik

SIMULERING AV STRØMNING OMKRING EN OLJELENSE
(Simulation of flow around an oilboom.)

The candidate is going to simulate viscous flow around a rectangular section. This can be a simplified element of an oil boom.
Initially, a literature study shall be performed with regard to challenges with oil boom operation and functionality.
Actual parameters for the study shall be chosen in collaboration with the supervisors. OpenFOAM will be used for the numerical simulations. This means that the candidate must thoroughly document all the cases simulated.
If time allows for it, two-fluid simulations shall be investigated with OpenFOAM. All the experience the candidate get from exploring OpenFOAM for two-fluid flow must be documented.

In the thesis the candidate shall present his personal contribution to the resolution of the problem within the scope of the thesis work. Theories and conclusions should be based on mathematical derivation and logic reasoning identifying the various steps in the deduction. The original contribution of the candidate and material taken from other sources shall be clearly defined. Work from other sources shall be properly referenced. The candidate should utilize the existing possibilities for obtaining relevant literature.
The thesis should be organized in a rational manner to give a clear exposition of results, assessments and conclusions. The text should be brief and to the point, with a clear language.

The thesis shall contain the following elements: A text defining the scope, preface, list of contents, summary, main body of thesis, conclusions with recommendations for further work, list of symbols and acronyms, references and appendices. All figures, tables and equations shall be numerated.
It is supposed that Department of Marine Technology, NTNU, can use the results freely in its research work by referring to the student's thesis.
The thesis shall be submitted July 23, 2015, in two copies.

Bjørnar Pettersen
Professor/supervisor

Co-supervisor: Associate professor Håvard Holm

Preface

This thesis concludes my Master of Science (MSc) in Marine Technology, at Norwegian University of Science and Technology (NTNU). During my studies I have had the pleasure of studying at the prestigious universities UC Berkeley and Korea Advanced Institute of Science and Technology (KAIST), at which I have learned a lot, both academically and about myself.

I want to thank my academic supervisors, both at NTNU and KAIST, respectively Professor Bjørnar Pettersen, Associate Professor Håvard Holm and Professor Hyun Chung for great assistance on this thesis. The discussions with my supervisors have been of greatest importance for my work.

I also want to thank Professor Daejun Chang, at KAIST, for great help on settling down in South Korea during this semester.

This research was supported in part with computational resources at NTNU provided by NOTUR, http://www.notur.no. I have been dependent on the computer Vilje during my work. A big thanks to the HPC group at NTNU for keeping the systems up and running, and for technical support.

Daejeon, South Korea, 10th July 2015

Henning Bjørvik

Abstract

Oil booms are used to control an oil spill, i.e. preventing it from spreading or reflecting it away from sensitive areas. As of today, oil booms are not efficient in open water. Failure modes are entrainment, drainage, splash-over, boom submergence and boom planning. In this thesis we focus on drainage failure, i.e. accumulation of oil at the face of the boom leads to a build-up until the oil escapes beneath the boom.

The open-source C++ library OpenFOAM is used in this work. The two-phase solver interFoam is chosen. The solver applies the Volume of Fluids method for interface capturing, i.e. the interface is calculated implicitly by volume fractions. The solver compresses and smears the interface over several cells (usually two or three). This is to prevent the interface from diffusing, which is a problem with the VoF method, especially when unstructured grids are applied. This is shown in this thesis.

This study contains of two parts, a one-phase study and a two-phase study. In order to contain oil, it is essential to simulate circulation on the front face of the oil boom. This is done by running one-phase simulations (with water). A convergence study is done showing that 54 000 cells are necessary in a small area in front of the oil boom (an area measuring 0.4 in horizontal dir. and 0.5 vertical dir., with oil boom dimension 0.5x0.5) to preserve the circulation. A study of increasing Reynolds number and angling the boom is also performed. The effect of increasing Reynolds number is smaller area of circulation, while increasing angle enlarges the area significantly.

A channel flow case is studied in order to understand how grid resolution, grid configuration, density ratio, Froude number and CFL number affects the flow. It is found that a skewed mesh easily introduce numerical diffusion, especially if the mesh is skewed near the inlet. The same result is found for cells with aspect ratio far from 1. The studies done in this thesis shows that a CFL number equal to 1 is not sufficient near the interface region, and a CFL number below 0.5 is advisable.

Applying an oil layer to the studies done in the one-phase section makes difficulties appear. The interface layer easily diffuse in this case. InterFoam uses the MULES algorithm to ensure boundedness of the phase fraction. A study of this shows that the results are highly affected

by the conditions given in this algorithm. The results in this thesis shows that it is advisable to increase number of α-corrections to 10, number of maximum iterations to 20 and error limit to 1e-20 for best solution. Also, increasing number of α-corrections within one time-step improves the results significantly, but at a higher computation cost. In addition to the solving algorithm MULES, different discretisation schemes are tested for the different terms in the equations, e.g. Crank-Nicolson for temporal discretisation, Gauss vanLeer for gradients etc.

Experience in this thesis shows that is highly recommended to apply a structured grid in the area of the interface. Handling a skewed mesh in this area while the interface is compressed towards the object is shown to be a very tough task. The main reason for this is how the VOF method works, i.e. that the solver always try to align the flow with the cell lines. This is shown in this thesis. A longer oil layer demands a lot more computation time. Since the vertical interface has to be sufficiently refined, this will demand an extensive amount of cells throughout the grid for a longer layer.

Introducing an oil layer to the one-phase study changes the area of circulation significantly. Introducing oil with kinematic viscosity 3e-6 $\frac{m^2}{s}$ makes the area of circulation larger. As with the one-phase case, increasing Re decrease the area of circulation. At some point, for a high enough Re, the oil will stop circulate and be drained under the object. For higher Re the circulation stops faster. In this thesis we find that circulation stops at t=9 for Re=100, at t=17 for Re=75, at t=50 for Re=50 and for Re=25 the circulation is still preserved after t=170.

A larger viscosity difference between the fluids are found to be harder to model. This is documented both by running channel cases and by running the circulation study. Numerical errors occurs more easily, especially when the interface is compressed in orthogonal direction, which occurs more easily with a higher viscosity fluid. The experience made in this thesis is that simulations with smaller viscosity difference is much easier to model with the VOF method.

Sammendrag

Oljelenser brukes i dag til å kontrollere oljeutslipp, dette kan være i form av å legge lensene rundt oljen for så å samle opp det tykkere laget, eller det kan være i form av å lede oljen unna sensitive områder. Dagens status for oljelenser er at de ikke fungerer optimalt i åpent vann. En oljelense kan feile ved entrainment (dråper av olje brytes løs fra oljelaget), drainage (kontinuerlig strøming), splash-over, neddykning av oljelense og helning av oljelense. I denne oppgaven studeres drainage. Under drainage akkumuleres oljen foran oljelensen. Når dette laget bygger seg tettere og tettere vil oljen etterhvert unslippe under oljelensen.

I dette arbeidet er OpenFOAM brukt. Dette er en c++ toolbox, og er distribuert som en åpen kildekode. To-fase løseren interFoam er brukt. Denne benytter Volume of Fluids metoden for å løse grenselaget mellom to fluider. Dette blir gjort implisitt ved at lokasjonen av de to fluidene blir beregnet og lokasjonen til grenselaget kan dermed fastslås der hvor det ikke er 100% av et fluid. InterFoam legger til et ledd i ligningen for volume fractions for å komprimere grenselaget. Dette gjøres for å unngå numeriske feil i grenselaget, noe som ofte skjer i VoF-metoden. Grenselaget er veldig sensitivt å modellere, spesielt med tanke på mesh. Ustrukturerte mesh fører ofte til ustabile løsninger av grenselaget, dette er vist i denne oppgaven.

Analysene i denne oppgaven kan deles i to deler, henholdsvis èn-fase og to-fase. Essensielt for at oljen ikke skal unslippe under oljelensen er å få oljen til å sirkulere foran oljelensen i motsetning til å flyte rett nedover og under oljelensen. Denne sirkulasjonen er simulert ved èn-fase strømning (med vann). En konvergensstudie er gjort som viser at 54 000 celler er nødvendig i et lite område foran oljelensen (området måler 0.4 i horisontal retning og 0.5 i vertikal retning, hvor oljelensens dimensjon er 0.5x0.5). Det er også gjort et studie av Reynolds nummer og vinkling av oljelensen. Dette viser at sirkulasjonsområdet blir mindre med økende Reynolds nummer, og større med økende vinkel på oljelensen.

En studie av en kanalstrømning er også gjort for å bedre forstå effekten av forfining av mesh, konfigurering av mesh, tetthetsforskjell mellom fluidene, Froude tall og CFL nummer. Simuleringene viser at spesielt et ustrukturert mesh, ikke orientert med strømningen, lett introduserer numeriske feil. Det samme gjelder celler med aspektforhold ikke nært 1. Simuleringene viser

også at CFL nummer må være betraktelig lavere enn 1 nær grenselaget, og helst lavere enn 0.5.

Å simulere strømning av olje rundt oljelensen medfører mange utfordringer. Grenselaget vil bevege seg over et større område vertikalt i motsetning til kanalstrømningen, og cellene vil i tillegg ikke være rette i forhold til strømningen. InterFoam benytter en algoritme kalt MULES for løse volume fractions, og for å korrigere disse. Ved å simulere flere caser er det i denne oppgaven funnet at resultatet avhenger sterkt av betingelsene gitt i MULES. Resultatene i denne studien viser at antall α-korreksjoner bør økes til 10, antall maksimum iterasjoner til 20 og errorgrense til 1e-20. Antall α-korreksjoner i et tidssteg bør også økes til betraktelig mer enn 1, men dette øker datatiden betraktelig. I tillegg til å sette kriterier for algoritmen MULES, er det i denne oppgaven også tested forskjellig diskretiseringsmetoder for de forskjellige leddene i ligningene som løses. For eksempel er forskjellen på Crank-Nicolson og Euler testet.

Erfaringene gjort i denne oppgaven viser at det er anbefalt å anvende et strukturert grid i området i og rundt grenselaget. Å kontrollere strømningen med et skewed grid imens grenselaget blir presset mot objektet viser seg å være en veldig vanskelig oppgave. Hovedgrunnen til dette er måten VOF metoden løser problemet. Metoden prøver alltid å rette strømningen langs cellelinjene. Dette er vist i denne oppgaven. Et lengre oljelag krever veldig mye mer datatid. Ettersom det vertikale grenselaget må være tilstrekkelig forfinet vil det kreve veldig mange celler horisontalt for et lengre oljelag.

Å legge til et oljelag til èn-fase studiene forandrer sirkulasjonsområdet betraktelig. Å legge til olje med viskositet 3e-6 $\frac{m^2}{s}$ gjør sirkulasjonsområdet større. Som for èn-fase studiene, vil høyere Re redusere sirkulasjonsområdet. På et tidspunkt, gitt at Re er høy nok, vil sirkulasjonen i oljelaget opphøre og oljen vil bli dratt ned og under objektet. For høyere Re skjer dette raskere. I denne oppgaven er det funnet at sirkulasjonen stopper på t=9 for Re=100, på t=17 for Re=75, på t=50 for Re=50 og for Re=25 er sirkulasjonen fortsatt tilstede på t=170.

Erfaringer gjort i denne oppgaven viser at større viskositetsforskjell mellom fluidene er vanskeligere å modellere. Numeriske feil i grenselaget ser ut til å inntreffe lettere, spesielt hvis grenselaget blir presset sammen ortogonalt (noe som skjer lettere for et fluid med høy viskositet). Erfaringene fra denne oppgaven tilsier at det er lettere å simulere strømninger med mindre forskjell i viskositet mellom fluidene.

Contents

List of Figures

Nomenclature

α	Phase fraction
δ	Dirac-delta function
κ	Curvature of interface between fluids
λ	Wave length
μ	Dynamic viscosity
ν	Kinematic viscosity
$\overrightarrow{g}$	Gravitational acceleration
$\overrightarrow{n}$	Unit normal vector for arbitrary shaped surface
$\overrightarrow{V}$	Fluid velocity
ρ	Density of fluid
σ_s	Surface tension coefficient
σ_{ij}	Stress tensor
τ	Shear stress
ζ	Wave amplitude

C	Convective fluxes
C_D	Drag coefficient
C_L	Lift coefficient
CFD	Computational Fluid Dynamics
CFL	Courant Frederich Levy
CPU	Central Processing Unit
CSF	Continuum Surface volumetric force
CV	Control Volume
dA	Finitesimal surface area
dV	Finitesimal volume
f	Body forces
F_D	Drag force
F_L	Lift force
f_v	Vortex shedding frequency
Fr	Froude number
FVM	Finite Volume Method
H	Wave height
HPC	High Performance Computing
i	Cell index
$interFoam$	OpenFOAM solver

j	Neighbour cell index
L	Characteristic length
$MEGA$	Mesh generator
$MULES$	Multidimensional Universal Limiter for Explicit Solution
$OpenFOAM$	Open Field Object and Manipulation, C++ library toolbox
P	Pressure stresses
p	Pressure
p^*	Modified pressure
PDE	Partial Differential Equation
Re	Reynolds number
St	Strouhal number
T	Viscous fluxes
T_v	Vortex shedding period
u	Fluid velocity x-dir. cartesian coordinates
v	Fluid velocity y-dir. cartesian coordinates
V_r	Relative velocity
VOF	Volume of Fluid
w	Fluid velocity z-dir. cartesian coordinates
WL	Water line

Chapter 1

Introduction

Oil spills in the ocean or coastal waters cause severe damage to the environment and economy. The world has witnessed several large oil spills during the history of oil extraction, both on land and in the ocean.

March 24, 1989, an oil tanker headed for Long Beach, California, hit Prince William Sound's Bligh Reef in the Gulf of Alaska. The name of the oil tanker was Exxon Valdez. Over the next days Exxon Valdez spills over 11 million US gallons of crude oil into the ocean according to Holleman (2014). Holleman (2014) also states that the oil damaged over 1300 miles (2100 km) of coastline, and Keim (2009) claims that the oil extracted over 11 000 square miles (28 000 km^2) of ocean. Several maps, made by Federal on scene coordinator's, showing the severity of the oil spill can be found in United States Coast Guard (1993).

According to Exxon Valdez Oil Spill Trustee Council (2015), tens of thousands of sea birds (estimated 100 000 - 250 000), thousands of sea otters (estimated around 2 800), hundreds of harbour seals and eagles died in short term after the accident. Also, billions of algeas and eggs were destroyed.

The economic impact on the region was huge, tourism was heavily impacted, with predicted losses of hundreds of millions of dollars. Another consequence of the oil spill was that commercial fishing was closed. Herring fishery in Prince William Sound is still, up to this date, closed according to Coil et al. (2010).

Still, 25 years after, the effects of the oil spill are seen. According to Holleman (2014), as of 2010, still 19 of 32 monitored wildlife habitats or populations are still not recovered. A pod of orcas (killer whales), lost 15 out of 22 members, and still have not produced a single calf since the accident.

The Exxon Valdez oil spill was very severe in its nature, but by far a smaller oil spill compared to the largest oil spill in history. In January 1991 the Iraqi military forces set fire to several hundreds of oil wells, resulting in an oil spill of an estimated amount of around 380-520 million US gallons, this is by far the largest oil spill in history. However, the largest official accidental oil spill is the Deepwater horizon oil spill in the Gulf of Mexico.

A blow out of a well caused an explosion on the oil rig. Numerous attempts to plug the well was unsuccessful, until it was capped about three months after the blow out. This resulted in an estimated oil spill of over 200 million US gallons. As of September 14th, 2010 (well was officially sealed September 19th, 2010), approximately 684 miles of Gulf coast shoreline was experiencing oil impacts, distributed over Louisiana, Mississippi, Alabama and Florida, as stated by Gulf Coast Ecosystem Restoration Council (2010). In total, approximately 1000 miles of shoreline was eventually impacted according to Warren (2011). The oil spill extracted over 68 000 square miles of ocean, according to satellite pictures obtained from Skytruth (2010).

On the most demanding day of operation, where over 181 miles of coast were impacted heavy or moderate, United States Coast Guard (2011) reported the use of:

- over 6000 vessels
- 82 helicopters
- 20 air-crafts
- over 47 000 personnel
- 3 795 985 feet of containment boom

In total, 4.2 million feet (1.3 million metres) of traditional containment booms were deployed, and 9.1 million feet (2.8 million metres) of a one-time sorbent boom (which absorbs the oil), as stated by Butler (2011). The problem is that these booms does not operate well in open sea.

As stated in Holleman (2014): "Whether it's Prince William Sound or the Gulf of Mexico, seldom is more than 10 % of the spilled oil recovered."

Oil containment booms are typical devices to control an oil spill. Surrounding the oil with an oil boom prevents oil from spreading over a larger area. To gather the oil in a smaller area, and thus making the oil layer thicker, is suitable for recovering the oil. Other uses are deflecting the oil to a desired location or diverting the oil away from sensitive areas (e.g. harbours, water intakes etc.).

Booms are usually efficient in sheltered water, but in open sea they are usually inefficient. This is because they are subjected to waves, current and wind, all at the same time.

1.1 Outline of thesis

- Chapter 2 gives an introduction to oil booms, i.e. the design, operating conditions and failure modes.
- Chapter 3 is an introduction to the open-source CFD software package OpenFOAM, which is used in this thesis.
- Chapter 4 gives the theoretical background for this thesis. The equations used are derived in this chapter, as well as the discretisation procedure and a discussion of boundary conditions. Typical parameters to describe the flow, e.g. Reynolds number and Froude number, are also described in this chapter.
- Chapter 5 describes in brief the Volume of Fluid (VOF) method, i.e. a common method to solve the interface between two fluids. The solver interFoam is used in this work. This solver is described and discussed in this chapter.
- Chapter 6 describes the numerical set-up for the analysis in this thesis, mainly the mesh configuration and defining patches.
- Chapter 7 contains the one-phase study performed in this thesis.
- Chapter 8 contains the two-phase study performed in this thesis.
- Chapter 9 gives a brief conclusion and recommendations for further work on this topic.

Chapter 2

Oil booms

2.1 Design

A typical Oil boom features the following elements, as presented in figure 2.1 and figure 2.2.

- A sub-surface skirt
- Freeboard
- Flotation element(s)
- Longitudinal tension member
- Ballast

Most boom designs are classified in two categories; Curtain Booms and Fence Booms.

- Curtain Boom: Such booms are typically supported by air or foam-filled flotation element, with a subsurface skirt. The flotation element has a circular cross section. An example can be seen in figure 2.1. Where WL is water line.
- Fence Boom: This type of booms are typically designed with internal or external flotation elements, and with a skirt (both subsurface and above the surface) with a flat cross section. An example can be seen in figure 2.2.

There are numerous other oil boom categories, e.g. Shore Sealing Boom, PermaGuard Boom and one time use booms. As it is not the scope of this thesis to discuss the different oil booms, these will not be explained here.

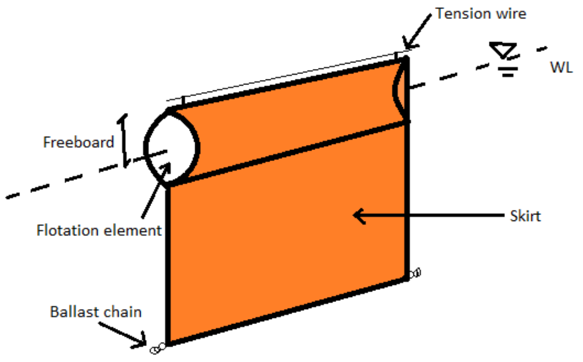

Figure 2.1: Illustration of a typical curtain boom

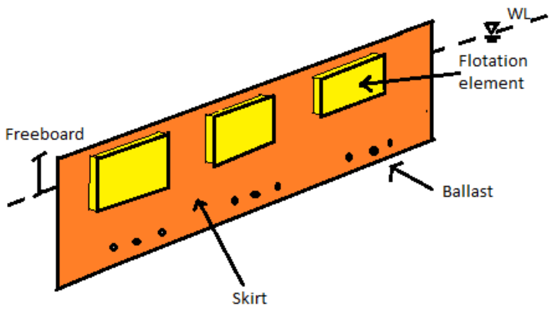

Figure 2.2: Illustration of a typical fence boom

There are currently no oil booms capable of operating in sea state 3 or larger, claimed by Kwak (2012). The sea states are defined according to Douglas sea and swell scale by Met office (2010) as shown below, where the most severe conditions are left out. To avoid having large flotation elements, which will make deployment and handling of the boom difficult, the boom should in best way follow the wave motions.

Table 2.1: Douglas sea and swell scale

State of the sea			Swell	
Code figure	Height [m]	Description	Code figure	Description
0	0	Calm (glassy)	0	No swell
1	0-0.1	Calm (ripped)	1	Very low (short, low waves)
2	0.1-0.5	Smooth (wavelets)	2	Low (long, low waves)
3	0.5-1.25	Slight	3	Light (short, moderate waves)
4	1.25-2.5	Moderate	4	Moderate (average, moderate waves)

Wave length:
Short wave: < 100 [m]
Average wave: 100 [m] - 200 [m]
Long wave: > 200 [m]

Wave height:
Low wave: < 2 [m]
Moderate wave: 2 [m] - 4 [m]
High wave: > 4 [m]

2.2 Failure modes

The efficiency of an oil boom depends on numerous factors, e.g. design, sea state of operation, the properties of the oil (viscosity and density). The typical failure modes are described briefly below, and shown in figure 2.3.

- **Entrainment:** Currents may lead to building of an upstream wave on the oil layer. Turbulence due to this phenomenon may break away droplets from the oil layer which can escape under the boom.
- **Boom submergence:** In high currents the boom might experience submergence, and therefore let the oil escape more easily. This is especially something to consider if the boom is moored.
- **Boom planning:** High currents may also cause the boom to angle. If current acts in one direction and wind in the other, this condition is highly likely to occur.
- **Splash-over:** In severe sea states, especially with high wind velocities, oil can be splashed over the freeboard. Low wave length to wave height ratio, λ/H, and high wave amplitudes, ζ, easily generates this condition.
- **Drainage:** High density oil easier accumulate at the face of the boom. As the oil layer gets thicker, oil easier escape under the boom. Low viscosity oil has higher possibility of drainage. The critical current speed for this failure mode is generally higher than for entrainment, as stated in Violeau et al. (2007).

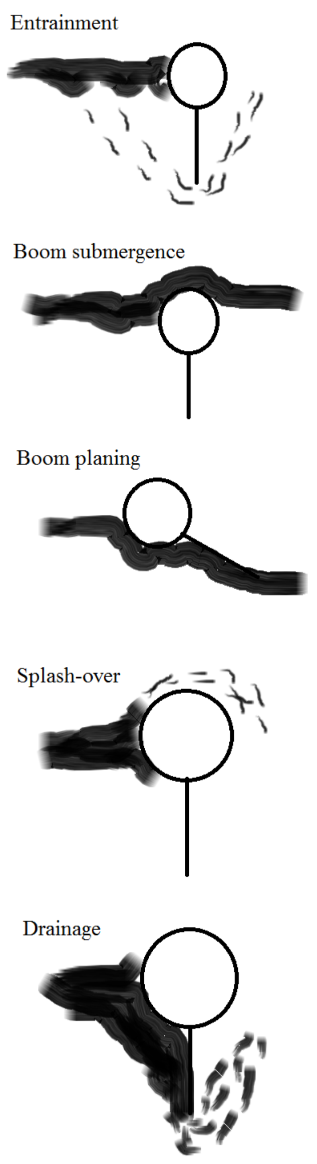

Figure 2.3: Typical failure modes for oil booms

2.3 Status today

In addition to the mentioned problems in section 2.1 and section 2.2 there are some other issues to think about.

A common way to try to control an oil spill is to deploy oil booms, and thereby use chemical dispersant underwater to break down the oil. A problem is that the oil booms are made of flexible materials. Therefore they may deform under waves and high currents. The tension at the components holding the oil boom together, e.g. tension wires, might be exerted to large forces. A study done by Ruifeng and Xun (2014) shows how the towing direction plays a significant role for the stress values in different sections of the oil boom. In Ruifeng and Xun (2014) it is also found that the maximum stress is concentrated around the middle section of the boom (when towed in a U-shape) and at the connections to the tugboats.

A problem with oil spills are the complex nature of them. Ornitz and Champ (2002) emphasises that every oil spill is different, i.e. the nature of material spilled, local environmental conditions, sensitivity of impacted natural resources, and effectiveness of response/clean-up technologies. How an oil spill is reacted to should also consider change in oil characteristics over time, e.g. by evaporation. According to Ornitz and Champ (2002), there are four main response categories to recover the oil:

- Chemical dispersant, as previously mentioned
- In-situ burning
- Mechanical recovery, mainly oil booms and skimmers (often in collaboration)
- Bioremediation

If an oil boom is towed this will also put restrictions on the towing vessels. Conventional oil boom systems normally operate around 0.7-1.0 knots, as stated in Ivshina et al. (2015). This makes the recovery vessel sail at very low speed, which is harmful for the engine. If pulled by two vessels, e.g. in an U-shape, the oil will gather in the rear end of the U-formation. A recovery pump can be placed here to recover the oil. According to Ivshina et al. (2015) this method usually have an operation speed around 5 knots.

A widely used method for containing the oil has been to use three vessels, two for towing and one for oil recovery. This is also problematic as operating multiple vessels close to each other enhances the possibility for accidents, e.g. broken booms or lines encountering the propellers.

In figure 2.4, a U-shape towing and a V-shape towing is shown. In the V-shape formation the skimmers are most naturally placed in the vessel at the V-tip, while for the U-shape the skimmers might as well be put at the two towing vessels.

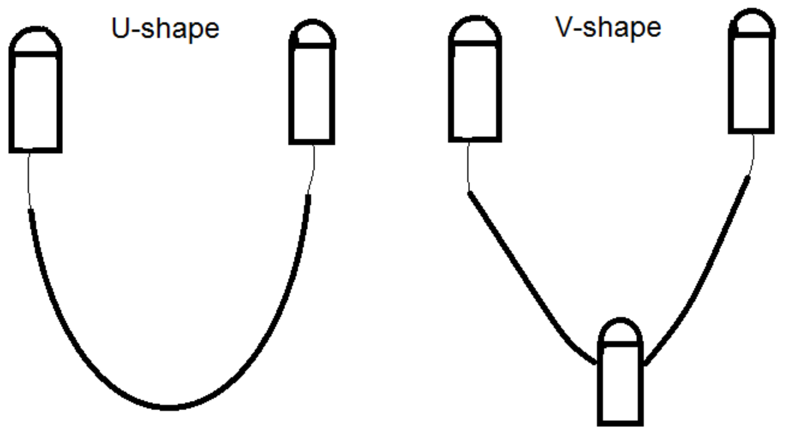

Figure 2.4: U-shape and V-shape towing configuration

Another issue with conventional oil booms are the cleaning costs. Large and complex oil booms might be difficult to clean after use. And are also naturally much harder to handle at shore.

To summarize, we want a simple but effective oil boom design. This design should be able to handle forces from both currents, wind and waves as this is essential in open sea. An oil boom design working at a higher speed is desirable both for fast prevention of spreading and the operation of the vessels in use.

Chapter 3

OpenFOAM

In this study, OpenFOAM (Open Field Operation and Manipulation) (OpenFOAM Foundation, 2015c) is used. This is an open-source C++ library, mostly used for CFD-problems, developed by OpenCFD Ltd. Even though OpenFOAM is mainly used for CFD problems, it can be used for almost all continuum mechanics problems where Finite volume method (FVM) can be applied. The program runs on Linux systems.

The main advantages of OpenFOAM are:

- open-source
- use of C++ code which is easily readable
- object oriented

An example of an equation representation in OpenFOAM is shown below, as in Greenshields (2015b). This shows the easy readable language.

```
solve
(
fvm::ddt(rho, U)
+ fvm::div(phi, U)
− fvm::laplacian(mu, U)
==
− fvc::grad(p)
);
```

$$\frac{\delta \rho U}{\delta t} + \nabla \cdot \phi U - \nabla \cdot \mu \nabla U = -\nabla p \qquad (3.1)$$

OpenFOAM comes with tools for meshing, e.g. BlockMesh, and for pre- and post-processing, e.g. ParaView. As an open-source library, the user is free to extend and alternate the library as wished. Also all the processes are run in parallel, so the user can maximize the usage of computer power.

3.1 Historical background

Originally, the toolbox was developed at Imperial College, London, in the late 1980s. The idea was to find a more flexible and powerful simulation platform than the recent ones, which at that time usually was written in FORTRAN. As C++ also was in development at that time, the development of the toolbox took large steps. After a while, Nabla Ltd. commercialized the toolbox under the name FOAM. In 2004 the package was released, under the new name OpenFOAM, as an open-source package under the GNU General Public License by OpenCFD Ltd. This was done, according to OpenFOAM Foundation (2015c), to benefit needs in research, development and consultancy.

OpenFOAM is now developed and distributed by The OpenFOAM Foundation, which holds the copyright to the source code, still as an open-source toolbox.

3.2 Computation time and capacity

CFD problems usually have a high data cost, which requires the user to run the simulation on several cores simultaneously. As previous mentioned, OpenFOAM allows the user to have full control of data usage, i.e. number of cores used etc. In this work, NTNU's supercomputer Vilje is used. Vilje, is a High Performance Computing (HPC) system procured by NTNU in cooperation with met.no and UNINETT Sigma. It contains 19.5 racks, consisting of 1404 nodes. Each node has 16 CPU cores (dual eight core), and therefore the system consists of in total 22464 cores.

3.3 OpenFOAM structure

In OpenFOAM there are three essential folders, i.e. '0' (or another preferred starting time), 'constant' and 'system'.

- **'0':** Contains initial/starting conditions for the flow.
- **'constant':** Contains the mesh and its properties. Also contains physical properties, i.e. g, ν and ρ. Where g is gravity, ν is kinematic viscosity and ρ is density of the respective fluid.
- **'system':** Contains the solving algorithm (numerical methods), starting time, timestep control, writeintervals etc. Also contains control of decomposition to several processors.

In this thesis the two-phase solver interFoam is used. Figure 3.1 shows the standard file structure for the interFoam solver. This solver is discussed in section 5.1.

```
       '0'                    'constant'                            'system'
  |__ alpha.water.org         |__dynamicMeshDict              |__  controlDict
  |__       U                 |__g                            |__ decomposeParDict
  |__     p_rgh               |__turbulenceProperties         |__    fvSchemes
                              |__transportProperties          |__    fvSolution
                              |__ 'PolyMesh'                  |__   setFieldsDict
                                  |__blockMeshDict
                                  |__boundary
                                  |__faces
                                  |__neighbour
                                  |__owner
                                  |__points
```

Figure 3.1: File structure of standard interFoam solver

When solved, OpenFOAM creates a folder for each timestep where the flow properties, i.e. velocity, pressure and phase fraction are stored in different files. These files contains the respective values at each grid point. Additional properties, e.g. vorticity can be calculated based on these values. There are many different solvers in OpenFOAM, which is suitable for different types of problems, e.g interFoam for two-phase flow, pisoFoam for one-phase incompressible flow, dieselFoam for diesel spray etc.

3.3.1 User-guide

This section is a brief user-guide for using the interFoam solver.

With a generated mesh the user should export the mesh to OpenFOAM (if the built-in mesh generator blockMesh is not used). The boundaries will be exported as patches, see section 6.2 for illustration. To these patches the user must assign boundary conditions in the respective files, i.e. 'alpha.water.org', 'U' and 'p_rgh' in the '0' folder.

The user should then move to the 'constant' folder. First, move to the 'polyMesh' folder. This folder contains all the information about the mesh generated. In the file 'boundary' the user should assign the boundaries chosen to the patch numbers generated. Also the type of patch should be assigned, e.g. for a 2D solution the patch type should be empty, this is explained in section 6.2. Then, move out of the 'polyMesh' folder. The folder 'g' contains the gravity acceleration, which can be alternated to adjust Froude number. 'transportProperties' contains the properties of the two fluids. By choosing transportModel Newtonian, the user do not need to alternate the properties under 'CrossPowerLawCoeffs' and 'BirdCarreauCoeffs' as these are ignored when Newtonian fluids are chosen. In this file the surface tension coefficient is also set.

The 'system' folder contains the setting for how the analysis is performed. The initial location of the fluids should be set in 'setFieldsDict', while 'decomposeParDict' contains the information on how the mesh is divided on several processors. These two functions have to be activated with the commands 'setFields' and 'decomposePar' in the terminal window. In 'fvSchemes' and 'fvSolution' the user should set the chosen discretisation schemes for the operators, and solver preferences (number of iterations, error limit etc.). StartTime and EndTime for the simulation should be set in 'controlDict'. WriteInterval sets when the results are written to file. As long as the user sets adjustTimeStep to yes, deltaT can be ignored. In this file Courant Frederich Levy number (CFL number) is also set, under maxCo and maxAlphaCo (for the interface region).

Chapter 4

Theoretical background

In a differential analysis we are interested in the fluid motion at every point in the fluid domain. The domain is split up in cells, which can vary in size and shapes. In each cell we solve a set of partial differential equations (PDEs) to find velocity, pressure etc. As we make these cells smaller and increase the number towards infinity, we will be able to solve the PDEs at any point in the fluid.

4.1 Fluid properties

In developing the governing equations, we will focus on Newtonian and incompressible fluids. We will also assume small changes in temperature locally, so that it is no need for a energy equation. In this thesis, the fluids included are water and oil. Water is a Newtonian fluids. Oil might be up to discussion, and depends on the type of oil. According to Rønningsen (2012), reservoir oils normally are Newtonian fluids.

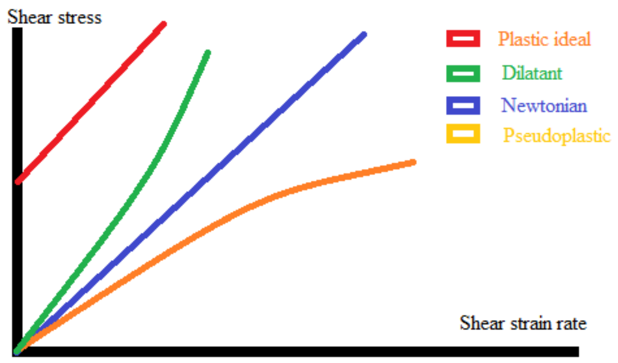

Figure 4.1: Newtonian fluid stress strain relation

As seen from figure 4.1, for a Newtonian fluid, the shear stress is linearly proportional to the shear strain rate.

4.2 Governing equations

In this section the governing equations used to solve the viscous fluid flow will be explained. In addition, boundary conditions need to be set at the boundaries. This is explained in section 4.3.

With the properties described in section 4.1, we are interested in the velocity and pressure distribution (density and temperature does not change significantly over the domain). To solve for the velocities and pressure, i.e. u(x,y,z,t), v(x,y,z,t), w(x,y,z,t) and p(x,y,z,t), as defined in figure 4.2, we need the following equations:

Conservation of mass $\longrightarrow$ Continuity equation
Newton's second law $\longrightarrow$ Euler/Navier-Stokes equations

The governing equations can be represented in various forms, e.g. integral form or discrete form. Problems might be solved using different forms of the equations. However, the form of the equations combined with the right solving technique is important for the numerical solutions.

4.2.1 Deriving the governing equations

Following the procedure described in Cengel and Cimbala (2009), we have:

Conservation of mass:

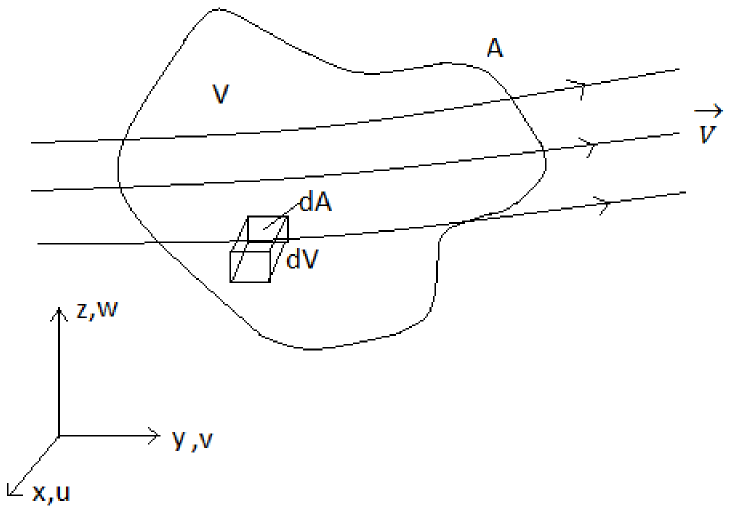

Figure 4.2: Illustration of control volume, control surface and reference frame

Figure 4.2 shows the reference frame, a control volume V with control surface A, and a flow pattern through the control volume. Also, a cubical cell dV with surface dA is showed.

Conservation of mass states that, on a control volume:

$$\left\langle \text{Net mass flow out of CV through CS} \right\rangle = \left\langle \text{Time rate of change of mass inside CV} \right\rangle$$

Therefore, we have the following expression for conservation of mass:

$$0 = \int_{CV} \frac{\delta \rho}{\delta t} dV + \int_{CS} \rho \vec{V} \cdot \vec{n} \, dA \tag{4.1}$$

where $\vec{V} = (u, v, w)$, and $\vec{n}$ is the unit vector pointing out of the volume.

Using the Divergence theorem we can derive the differential form of equation 4.1 as followed:

Divergence theorem:

$$\int_{CV} \vec{\nabla} \cdot \vec{G} \, dV = \oint_{CS} \vec{G} \cdot \vec{n} \, dA \tag{4.2}$$

As seen from equation 4.1 and equation 4.2, substituting G by $\rho \vec{V}$, equation 4.1 can be written as:

$$\int_{CV} \{\frac{\delta \rho}{\delta t} + \vec{\nabla} \cdot \left(\rho \vec{V}\right)\} dV = 0 \tag{4.3}$$

For this to be true for any shape or size of a cell, we argue that the Continuity equation must be:

$$\frac{\delta \rho}{\delta t} + \vec{\nabla} \cdot \left(\rho \vec{V}\right) = 0 \tag{4.4}$$

For the incompressible case the first term is equal to zero, and we therefore have:

$$\vec{\nabla} \cdot \vec{V} = 0 \tag{4.5}$$

or

$$\frac{\delta u}{\delta x} + \frac{\delta v}{\delta y} + \frac{\delta w}{\delta z} = 0 \tag{4.6}$$

Cauchy's equation:

The Cauchy's equation describes the total forces acting on a control volume, which is the sum of the rate at which momentum changes inside the control volume and the difference in inflow and outflow rate of momentum.

The linear momentum equation for a control volume is as given in equation 4.7.

$$\int_{CV} \rho \vec{g} \, dV + \int_{CS} \sigma_{ij} \cdot \vec{n} \, dA = \int_{CV} \frac{\delta}{\delta t} \left(\rho \vec{V}\right) dV + \int_{CS} \left(\rho \vec{V}\right) \vec{V} \cdot \vec{n} \, dA \tag{4.7}$$

where σ_{ij} is defined as in equation 4.14. $\overrightarrow{g}$ is the gravitational acceleration.

We use the same procedure as for the Continuity equation, i.e. we use the Divergence theorem. We then end up with the following differential equation:

$$\int_{CV} \{\frac{\delta}{\delta t}\left(\rho \overrightarrow{V}\right) + \overrightarrow{\nabla} \cdot \left(\rho \overrightarrow{V}\overrightarrow{V}\right) - \rho \overrightarrow{g} - \overrightarrow{\nabla} \cdot \sigma_{ij}\} dV = 0 \tag{4.8}$$

As for the Continuity equation this should hold for any cell. Therefore equation 4.8 can be written:

$$\frac{\delta}{\delta t}\left(\rho \overrightarrow{V}\right) + \overrightarrow{\nabla} \cdot \left(\rho \overrightarrow{V}\overrightarrow{V}\right) = \rho \overrightarrow{g} + \overrightarrow{\nabla} \cdot \sigma_{ij} \tag{4.9}$$

To summarize what we have this far, the continuity equation and three Cauchy's equation (one in each direction):

$$\frac{\delta u}{\delta x} + \frac{\delta v}{\delta y} + \frac{\delta w}{\delta z} = 0 \tag{4.10}$$

$$\rho \frac{Du}{Dt} = \rho g_x + \frac{\delta \sigma_{xx}}{\delta x} + \frac{\delta \sigma_{yx}}{\delta y} + \frac{\delta \sigma_{zx}}{\delta z} \tag{4.11}$$

$$\rho \frac{Dv}{Dt} = \rho g_y + \frac{\delta \sigma_{xy}}{\delta x} + \frac{\delta \sigma_{yy}}{\delta y} + \frac{\delta \sigma_{zy}}{\delta z} \tag{4.12}$$

$$\rho \frac{Dw}{Dt} = \rho g_z + \frac{\delta \sigma_{xz}}{\delta x} + \frac{\delta \sigma_{yz}}{\delta y} + \frac{\delta \sigma_{zz}}{\delta z} \tag{4.13}$$

As seen from the equations above, there are 10 unknowns and only 4 equations. To make these equations solvable we need to find a connection between the stress tensor and the fluid properties.

First, we separate friction/shear stresses and pressure stresses. A flow around a body generates friction/shear drag due to tangential stresses along the body surface. The separation of fluid causes a pressure difference which induce a pressure drag. For blunt bodies the pressure drag will dominate due to flow separation, while for streamlined bodies the friction drag will dominate. For further explanation on this, see Greco (2012).

The pressure stresses always acts inwards and normal to the face of the plane. If a body is still, in calm fluid, the only contribution to σ_{ij} would be the pressure stresses, which is the first bracket in equation 4.14.

If there is a relative velocity between body and fluid, the second bracket of equation 4.14 will be non-zero. These are the friction stresses.

Assuming the fluid properties as in section 4.1, we can write the stress tensor as:

$$\sigma_{ij} = \begin{Bmatrix} -P & 0 & 0 \\ 0 & -P & 0 \\ 0 & 0 & -P \end{Bmatrix} + \begin{Bmatrix} 2\mu\frac{\delta u}{\delta x} & \mu\left(\frac{\delta u}{\delta y}+\frac{\delta v}{\delta x}\right) & \mu\left(\frac{\delta u}{\delta z}+\frac{\delta w}{\delta x}\right) \\ \mu\left(\frac{\delta v}{\delta x}+\frac{\delta u}{\delta y}\right) & 2\mu\frac{\delta v}{\delta y} & \mu\left(\frac{\delta v}{\delta z}+\frac{\delta w}{\delta y}\right) \\ \mu\left(\frac{\delta w}{\delta x}+\frac{\delta u}{\delta z}\right) & \mu\left(\frac{\delta w}{\delta y}+\frac{\delta v}{\delta z}\right) & 2\mu\frac{\delta w}{\delta z} \end{Bmatrix} \tag{4.14}$$

where μ is the effective dynamic viscosity of the fluid, and P is the pressure stresses. This can also be written:

$$\sigma_{ij} = -\overrightarrow{P} + \tau \tag{4.15}$$

where

$$\tau = \mu(\overrightarrow{\nabla}\overrightarrow{V} + \overrightarrow{\nabla}\overrightarrow{V}^T) \tag{4.16}$$

Navier-Stokes equation:

Inserting equation 4.14 into the Cauchy's equations (in all directions) and re-writing the terms we obtain the following expression for the incompressible Navier-Stokes equation:

$$\rho\frac{D\overrightarrow{V}}{Dt} = -\overrightarrow{\nabla}P + \rho\overrightarrow{g} + \mu\nabla^2\overrightarrow{V} \tag{4.17}$$

Once again we summarize our equations, the continuity equation and three Navier-Stokes equations (one in each direction):

$$\frac{\delta u}{\delta x} + \frac{\delta v}{\delta y} + \frac{\delta w}{\delta z} = 0 \tag{4.18}$$

$$\rho\left(\frac{\delta u}{\delta t} + u\frac{\delta u}{\delta x} + v\frac{\delta u}{\delta y} + w\frac{\delta u}{\delta z}\right) = -\frac{\delta P}{\delta x} + \rho g_x + \mu\left(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} + \frac{\delta^2 u}{\delta z^2}\right) \tag{4.19}$$

$$\rho\left(\frac{\delta v}{\delta t} + u\frac{\delta v}{\delta x} + v\frac{\delta v}{\delta y} + w\frac{\delta v}{\delta z}\right) = -\frac{\delta P}{\delta y} + \rho g_y + \mu\left(\frac{\delta^2 v}{\delta x^2} + \frac{\delta^2 v}{\delta y^2} + \frac{\delta^2 v}{\delta z^2}\right) \tag{4.20}$$

$$\rho\left(\frac{\delta w}{\delta t} + u\frac{\delta w}{\delta x} + v\frac{\delta w}{\delta y} + w\frac{\delta w}{\delta z}\right) = -\frac{\delta P}{\delta z} + \rho g_z + \mu\left(\frac{\delta^2 w}{\delta x^2} + \frac{\delta^2 w}{\delta y^2} + \frac{\delta^2 w}{\delta z^2}\right) \tag{4.21}$$

We now have 4 unknowns and 4 equations, i.e. the system is solvable. The Navier-Stokes equation are non-linear partial differential equations, except from in some specific cases where the equations can be simplified to linear equations. Analytical solutions are only obtainable for simple geometries, and the equations are therefore most often solved numerically. This is what OpenFOAM does, and will be explained in section 4.4.

4.2.2 Integral form

From equation 4.1 and equation 4.7 we can further simplify these to get the desired form of the integral equations, as follows:

As the control volumes are fixed both in time and space, 4.1 reduces to:

$$0 = \int_{CS} \vec{V} \cdot \vec{n} \, dA = 0 \tag{4.22}$$

Using that the fluid is incompressible and some re-writing we obtain the following expression for 4.7:

$$\frac{d}{dt}\int_{CV} \rho \vec{V} dV + \int_{CS} \rho \vec{V}\left(\vec{V} \cdot \vec{n}\right) dA = \int_{CS} \left(-p\vec{n}\right) dA + \int_{CS} \tau \vec{n} \, dA + \int_{CV} \rho \vec{g} \, dV \tag{4.23}$$

As can be further simplified by the Gauss theorem:

$$\frac{d}{dt}\int_{CV} \rho \vec{V} dV + \int_{CS} \rho \vec{V}\left(\vec{V} \cdot \vec{n}\right) dA = \int_{CV} \left(-\nabla p + \nabla \tau + \rho \vec{g}\right) dV \tag{4.24}$$

4.2.3 Two-phase flow

Modelling multiple phase fluids might be done by adding more equations, for the respective fluids, and then match them at the interface of the fluids. In this thesis, we want to expand our equations to account for another fluid.

Equation 4.1 remains unchanged. But the momentum equation do change by accounting for forces in the interface of the fluid, expanding equation 4.9 in the following way:

$$\frac{\delta}{\delta t}\left(\rho \vec{V}\right) + \vec{\nabla} \cdot \left(\rho \vec{V} \vec{V}\right) = \rho \vec{g} + \vec{\nabla} \cdot \sigma_{ij} + \int_{S(t)_{int}} \sigma_s \kappa' n' \delta(x - x') dS_{int} \tag{4.25}$$

where σ_s is the surface tension coefficient, $\kappa' = \vec{\nabla} \cdot (\frac{\vec{\nabla}\alpha}{|\vec{\nabla}\alpha|})$ is the curvature of interface , n' is a unit vector normal to the interface and δ is a Dirac delta function. x' denotes the interface, while x is the point of evaluation. α is the phase fraction. See section 5.1 for explanation of α.

This additional last term in equation 4.25 is constructed by using a Heaviside function. This will not be further derived in this thesis, but is thoroughly explained in Tryggvason et al. (1998).

4.3 Boundary conditions

In addition to the governing equations and the discretization of these, two steps are important to simulate the problem:

- Define your computational domain by meshing
- Set appropriate boundary conditions

So, in addition to the equations of motions (which are the same for different geometries) and the computational domain, the boundary conditions are of great importance to obtain an accurate solution of the governing equations by CFD.

Boundary conditions can be divided into 3 groups:

- Dirichlet boundary conditions: prescribing the value of a variable at the boundary, e.g. $u(x) = const.$
- Neumann boundary conditions: prescribing the gradient normal to the boundary, e.g. $\frac{\delta u(x)}{\delta n} = const.$
- Robin boundary conditions: a mix of Dirichlet and Neumann boundary conditions, e.g. $a \times u(x) + b \times \frac{\delta u(x)}{\delta n} = const.$

For an incompressible, one-fluid, laminar case, the only variables which needs to be determined are the velocity and pressure. These will be found in the folder '0', as described in section 3.3.

Boundaries might be assigned as shown in figure 4.4 depending on the problem. In the sections below, boundary conditions for the different boundaries are discussed. Several types of boundary conditions may be applied in OpenFOAM. In this section, only the relevant ones will be described. Turbulence boundary conditions will not be discussed. If a turbulence model is to be applied to the problem, the wall roughness, wall functions etc. have to be accounted for.

4.3.1 Wall boundary conditions

Fluid cannot pass through a wall, i.e. the normal component of velocity has to be zero relative to the wall. Also, the no slip condition states that the tangential velocity is zero.

In tangential direction, we have:

$$\vec{V}_{fluid} = \vec{V}_{wall} \tag{4.26}$$

What this means is that the fluid particles close to the wall will stick to the wall and follow the walls movements. So, in the case of a stationary wall, i.e. $\vec{V}_{wall} = 0$, the fluid close to the wall will experience $\vec{V}_{fluid} = 0$.

The same arguments are used on an interface between two fluids, as shown in figure 4.3. But in this case the shear stress, τ_s must be equal on the two fluids, i.e. for the interface between fluid A and fluid B the conditions $\vec{V}_A = \vec{V}_B$ and $\tau_{s,A} = \tau_{s,B}$ hold.

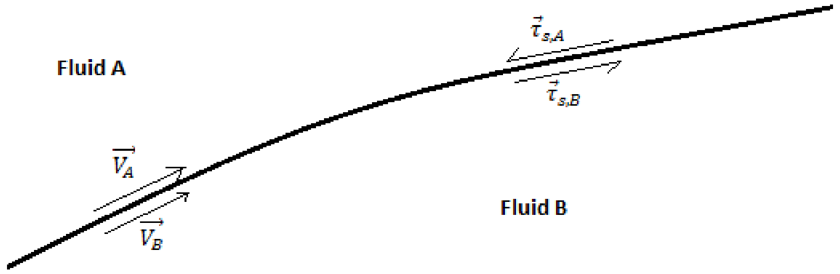

Figure 4.3: Boundary conditions on interface between two fluids

4.3.2 Inlet/Outlet boundary conditions

There are several ways to define inlet and outlet boundary conditions. In the governing equations, equation 4.18 - equation 4.21, pressure and velocity are coupled. We do not want to specify both pressure and velocity at the same place, as this will lead to an over-specification, i.e. the specified value of pressure and velocity can not hold at the same time, as stated in Bakker (2006).

We distinguish between velocity specified conditions and pressure specified conditions.

- velocity inlet/outlet: velocity specified, pressure not specified.
- pressure inlet/outlet: pressure specified, velocity not specified.

In case of a velocity inlet, we specify the inlet velocity, and the pressure adjust itself. Vice versa for a pressure inlet. In figure 4.4, an example of boundary conditions for an incompressible case is shown.

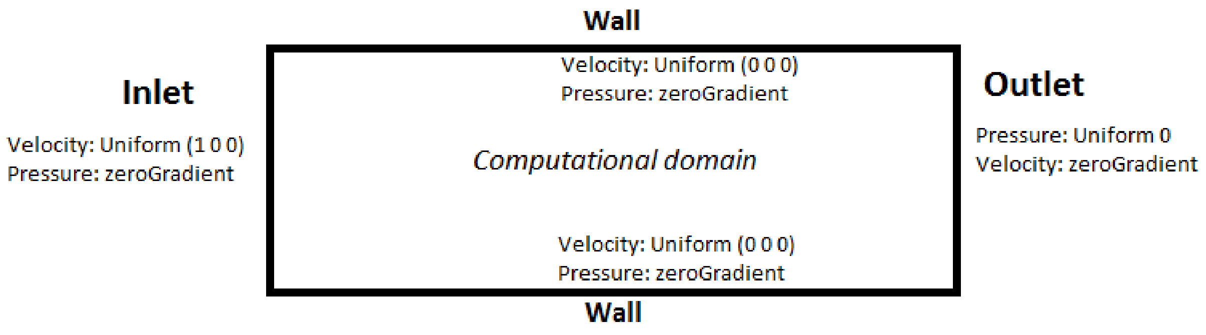

Figure 4.4: Example of boundary conditions for an incompressible case

4.3.3 Symmetryplane or slip boundary conditions

Using a symmetryplane or slip condition is practically equivalent in most cases. A slip wall does not necessarily need to be planar as a symmetryplane. A slip wall does neither necessarily impose zero normal gradient to all parameter either.

In general, a symmetryplane or slip wall imposes:

- zero normal velocity at the plane, $v = 0$.
- zero normal gradients of all variables at the plane, $\frac{\delta v}{\delta n} = 0, \frac{\delta p}{\delta n} = 0$.

4.4 Discretisation

Before solving the equations numerically we need to discretize them. The most common solvers in OpenFOAM, e.g. interFoam, uses Finite Volume Numerics to solve the PDEs. As this is the method used in this thesis, this is the only method explained in this section. The PDEs are being converted into a set of algebraic equations,

$$Ax = b \tag{4.27}$$

where x is the unknown velocities, and A and b are dependent on several factors, such as mesh, pressure etc.

OpenFOAM solves this equation by an iterative solver, which stops when a set residual is reached.

4.4.1 Discretisation of computational domain

The most common cell type in OpenFOAM are arbitrary shaped hexahedral cells, even though other shapes might be created by for example collapsing two edges. Shapes like pyramid, prism and wedge might be created this way. How these cells are built up can be seen by studying the source code of blockMeshDict, which is the built-in mesh generator of OpenFOAM. It is not the purpose of this thesis to discuss this.

OpenFoam uses co-located variables, i.e. the solution variables are stored in the cell center, and can from these values be interpolated to find the values at the surface of the cell as stated in OpenFOAM Foundation (2015a). This is shown in figure 4.5.

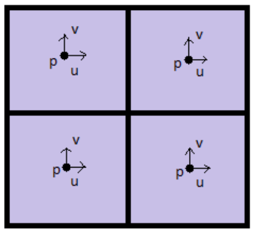

Figure 4.5: Colocated variables

For a more complex geometry than a cube, the computational point is where $\int_{V_p} (x - x_p)\, dV = 0$, which is derived from the definition of centroid position, i.e. $V_p x_p = \int_{V_p} x dV$. Where V_p is volume of the cell.

4.4.2 Discretisation of governing equations

While the Finite Difference Method (FDM) is based on discretisation of the differential operators, the Finite Volume Method (FVM) is based on discretisation of the integral form of the equations. That is, we are no longer looking at the nodes, as in the FDM, but rather on the surfaces of a control volume. OpenFOAM discretize the governing equations (both in space and time) in arbitrary polyhedral control volumes according to Jasak and Weller (1999). The flux between the control volumes is balanced. If we sum these local discrete equations contribution, the global conservation equations are retrieved. Put in another way, we will end up with a large set of discrete algebraic equations which we can solve numerically.

Our starting point are the governing equations, see section 4.2, in integral form divided by ρ. Written in a convenient way, as in Ransan (nd):

$$\frac{\delta}{\delta t}\iiint_{CV(t)} U dV + \iint_{CS(t)} C\cdot\vec{n}\, dS - \iint_{CS(t)} T\cdot\vec{n}\, dS = \iiint_{CV(t)} f dV \tag{4.28}$$

where $C\cdot n = C_1\cdot n_x + C_2\cdot n_y + C_3\cdot n_z$, and equally $T\cdot n = T_1\cdot n_x + T_2\cdot n_y + T_3\cdot n_z$.

$$U = \begin{pmatrix} 0 \\ u \\ v \\ w \end{pmatrix} \quad C_1 = \begin{pmatrix} u \\ u^2+p \\ uv \\ uw \end{pmatrix} \quad C_2 = \begin{pmatrix} v \\ uv \\ v^2+p \\ vw \end{pmatrix} \quad C_3 = \begin{pmatrix} w \\ uw \\ vw \\ w^2+p \end{pmatrix}$$

$$T_1 = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \end{pmatrix} \quad T_2 = \begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \end{pmatrix} \quad T_3 = \begin{pmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \end{pmatrix} \quad f = \begin{pmatrix} 0 \\ 0 \\ 0 \\ g \end{pmatrix}$$

Where C are the convective fluxes, T are the viscous fluxes and f are the body forces.

The FVM reformulates this integral expression to a discrete form:

$$V_i\frac{\delta U_i}{\delta t} + \sum_j C_{(i+j)/2} - \sum_j T_{(i+j)/2} = f_i \tag{4.29}$$

Where i refer to the cell index and j refer to the index of the cell next to cell i, see figure 4.6. V_i refers to the volume of the cell. As some of the values depends on the value of the neighbouring cells, we have obtained the equation system in the form of equation 4.27.

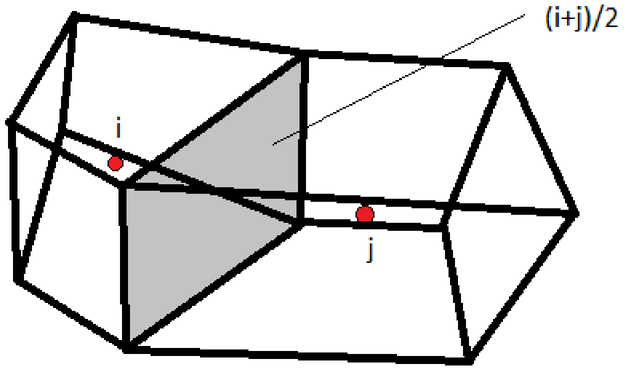

Figure 4.6: Index numbering of cells, as in (Ransan, nd)

For the mathematical derivation of the discrete form, see Ransan (nd) and Jasak and Weller (1999).

One advantage of OpenFOAM is that it allows the user to apply different schemes for each term in the equation, e.g. all gradients ($\vec{\nabla} p$, $\vec{\nabla} \vec{V}$ etc.) can be set to Gauss linear, Gauss limitedLinear etc. and time derivatives can be set to Euler, Crank-Nicolson etc. Different schemes combined with different solvers might work out more efficient than other combinations. The different schemes can be assigned in the 'system' directory.

4.4.3 Discretisation of time

As set standard in the interFoam solver, the Euler method is applied for temporal discretisation. In this thesis we apply both Euler and Crank-Nicolson. With explicit discretisation the variables are calculated based on the values from the previous time-step. To ensure stability with this method we have to limit the time-step strongly. However, with implicit method the variables are based on the next time-step. Both these systems are first order. If we combine these methods, i.e. Crank-Nicolson method, we obtain a second order solution. Therefore, the Crank-Nicholson method is a second order method, while Euler method is of first order.

As stated, the Crank-Nicolson method applies a combination of the forward Euler method and the backward Euler method.

We combine the terms in the equation to a general vector F, and let n denote current time step. For example the following PDE, in x-direction:

$$\frac{\delta \overrightarrow{V}}{\delta t} = F \tag{4.30}$$

The forward Euler is as follows:

$$\frac{\overrightarrow{V}^{n+1} - \overrightarrow{V}^{n}}{\Delta t} = F_i^n \tag{4.31}$$

And backward Euler is:

$$\frac{\overrightarrow{V}^{n+1} - \overrightarrow{V}^{n}}{\Delta t} = F_i^{n+1} \tag{4.32}$$

We see from combining these two methods, the Crank-Nicolson method then applies as:

$$\frac{\overrightarrow{V}^{n+1} - \overrightarrow{V}^{n}}{\Delta t} = \frac{1}{2}\left\{F_i^{n+1} + F_i^n\right\} \tag{4.33}$$

OpenFOAM uses a blending factor, ψ, for the Crank-Nicolson scheme as a weighting factor between the forward Euler scheme and backward Euler scheme. Setting $\psi = 0$ implies a pure Euler scheme, and setting $\psi = 1$ implies running a pure Crank-Nicolson scheme.

4.5 Description of flow

4.5.1 Reynolds number

Reynolds number is defined as the ratio between inertial force and viscous force, $\frac{Inertial force}{Viscous force}$. Deriving it from the expressions of the forces we obtain the following expression:

$$Re = \frac{\rho V L}{\mu} = \frac{V L}{\nu} \tag{4.34}$$

where:

- V is mean incident relative velocity between fluid and object
- L is characteristic length, e.g. for cylinder L is diameter

Re might be used to characterise the flow, and describe the flow regime at different Re. In general, laminar flow occurs at low Re (where viscous forces dominate) and turbulent flow at high Re (where inertial forces dominate).

- Laminar flow: flow is well organized
- Turbulent flow: flow fluctuates around a mean value.

4.5.2 Froude number

Froude number is defined as $\frac{Inertial force}{Gravitational force}$. Deriving it from the expressions of the forces we obtain the following expression:

$$Fr = \frac{V}{\sqrt{gL}} \tag{4.35}$$

where:

- V is the average liquid velocity at a cross section
- L is the characteristic length

4.5.3 Strouhal number

The Strouhal number is a dimensionless number which describes the oscillation of the fluid, or the vortex shedding frequency. For a plate it can be defined as:

$$St = f_v \frac{H}{U} \tag{4.36}$$

where f_v is the vortex shedding frequency, $\frac{1}{T_v}$, and H and U is as shown in figure 4.7.

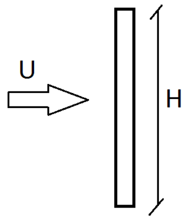

Figure 4.7: Definition of dimensions used in Strouhal number

4.5.4 Force coefficients

The drag and lift force coefficients are defined as:

$$C_D = \frac{F_D}{\frac{1}{2}\rho U^2 H} \tag{4.37}$$

$$C_L = \frac{F_L}{\frac{1}{2}\rho U^2 H} \tag{4.38}$$

As explained in section 4.2.1 and equation 4.14, the total drag force consists of contribution from the pressure forces and frictional forces.

$$F_{Dtotal} = F_{pressure} + F_{friction} \tag{4.39}$$

Due to the vortex shedding, the $F_{pressure}$ will vary in time.

4.5.5 Courant Frederich Levy number

A well known stability criteria in CFD analysis, is given by the CFL number. This term connects the time step with the cell size, which clearly can not be independent in a CFD analysis.

$$CFL = \frac{|u|\,\Delta t}{\Delta x} \leq I \tag{4.40}$$

Δt is the time step, Δx is the cell size and $|u|$ is the magnitude of velocity. I is scheme-dependent, but usually equal to unity according to Ramshaw (2011). This means that it is not always the case that the CFL number is largest at the smallest cells, as the magnitude of velocity might be higher elsewhere.

Equation 4.40 is the one dimensional form of the CFL number, in x-direction. But in general this condition holds in any direction.

Local densities of globally conserved mass, momentum and energy are transported from one place to another with the fluid motion, i.e. they are subject to convection. At a fixed point in the grid the local values will therefore vary with time.

The CFL number condition states that the distance the fluid travels in one time step must be less than the cell size in that direction.

With the interFoam solver, Δt might be adjusted automatically to keep CFL number below I. Time step control is more important when we have an interface to track, due to the time step sensibility of the surface tracking algorithm. According to Greenshields (2015b), the CFL number should not exceed 0.5 in the interface region.

Chapter 5

Two-phase solver

An important part of any multi-phase problem is to keep track of the interface at every time-step. For immiscible fluids it mainly exists two ways of doing this, as described in Emad (2014):

- Interface tracking method
- Interface capturing method

In the Interface method the interface is tracked by either assigning marker points to it, or moving the mesh with the interface. In the Interface capturing method the interface is not directly tracked, but rather calculated by fluid fractions, i.e. volume fractions of the fluid. If the calculated volume fraction shows that the cell is not completely occupied by one fluid, then this cell must be in the interface.

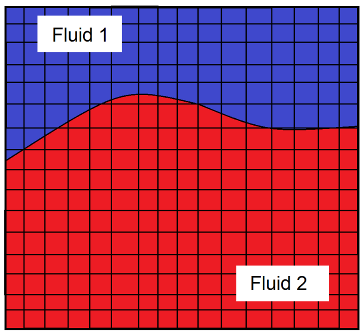

Figure 5.1: Visualisation of Interface capturing method

Only the Interface capturing method will be discussed further in this thesis. Figure 5.1 shows the principle of this method, i.e. solving the volume fractions in each cell. The method applies on a rigid grid.

In OpenFOAM the solvers interFoam, LTSinterfoam and InterDyMFoam are solvers that uses this method.

The solver used in this thesis, i.e. interFoam, is discusses in section 5.1.

5.1 interFoam

InterFoam is a solver for two-phase flow, with incompressible, isothermal and immiscible fluids. The solver uses the VOF method (No and Woodward, 1976) (Harlow and Welch, 1965) for interface capturing, and PISO (Issa, 1984) for pressure velocity coupling.

The VOF method solves the governing equations for each phase while the interface is tracked. The method uses α, volume fraction, to set the proportion of the respective cell that is occupied by a certain fluid. The interface location, its normal and curvature are therefore only implicitly known. In a post-processing application, e.g. ParaView, it can be seen that the interface is not a clear cut from one phase to the other, but instead a mixture of the two fluids over several cells. This mixture is very sensitive to the mesh, this is addressed in section 8.1. α is a function of both time and position. In each cell we have:

$$\alpha = \begin{cases} 1, & \text{for fluid 1} \\ \text{between 0 and 1}, & \text{for interface} \\ 0, & \text{for fluid 2} \end{cases} \tag{5.1}$$

In figure 5.1 we can set Fluid 1 to $\alpha = 1$ and Fluid 2 to $\alpha = 0$

By α we can write an indicator function which represents the volume fraction of one phase, as done by Rusche (2002). This is a re-writing of the conservation of mass using equation 5.1 and equation 5.3. The derivation can be seen in Deshpade et al. (2012) .

$$\frac{\delta \alpha}{\delta t} + \vec{\nabla} \cdot \left(\vec{V} \alpha \right) = 0 \tag{5.2}$$

This equation is solved simultaneously with the momentum and continuity equations.

It is important to keep the solution of this equation bounded, i.e. make the values of α not drop below 0 or above 1. The MULES algorithm is applied to preserve this, more on this algorithm can be found in section 5.1.1 and section 8.2.2.

The physical properties are defined as weighted averages in the following way:

$$\rho = \alpha \rho_1 + (1-\alpha)\rho_2 \tag{5.3}$$

$$\mu = \alpha \mu_1 + (1-\alpha)\mu_2 \tag{5.4}$$

where subscripts 1 and 2 is for the respective fluids in the case. As the α-function goes from 0 to 1 over an infinitesimal thickness we experience difficulty in approximating the gradient of the function.

A big advantage of the VOF method is that it is mass conservative, i.e. with the appropriate numerics implemented the volume of each phase is conserved exactly.

To account for the compression of the surface, OpenFOAM adds an extra term to equation 5.2. So equation 5.2 is written as in Rusche (2002):

$$\frac{\delta \alpha}{\delta t} + \vec{\nabla} \cdot \left(\vec{V} \alpha\right) + \vec{\nabla} \cdot \left(\vec{V_r} \alpha (1-\alpha)\right) = 0 \tag{5.5}$$

V_r represents a velocity field which is suitable to compress at the interface of the fluids, i.e. a relative compressive velocity between the two phases. This additional convective term, is used to compress the interface to a sharper one. Without this term, the interface would be much less refined. From equation 5.5 we can see that this term only affects the result in the interface regions, due to the term $\alpha(1-\alpha)$. In other regions $\alpha = 0$ or $\alpha = 1$ which will make this term go to zero. This term has showed some errors (e.g. loss of curvature), especially when the mesh configuration is not aligned with the flow direction. Issues with this solving technique are discussed in section 5.1.2.

Another issue with this way of tracking the interface (not knowing its exact shape and location), is that the surface integral in equation 4.25 can not be evaluated directly. To overcome this, the continuum surface force model (CSF), by Brackbill et al. (1992), is used. It solves the integral in the following way:

$$\int_{S(t)_{int}} \sigma_s \kappa' n' \delta(x - x') dS_{int} \approx \int_{CV} \sigma_s \kappa \vec{\nabla} \alpha dV \tag{5.6}$$

This equation does not take into consideration variable surface tension coefficient. Surface tension, σ_s, has to be set in the file 'transportProperties' in OpenFOAM.

The interFoam solver uses a modified pressure, according to Rusche (2002), as stated below. The reason for this is discussed in section 7.2.

$$p^* = p - \rho g \cdot x \tag{5.7}$$

This term has to be accounted for in the momentum equation, i.e. equation 4.25.

Rewriting the gradient, as in Rusche (2002):

$$\vec{\nabla} p^* = \vec{\nabla} p - \vec{\nabla}(\rho g \cdot x) = \vec{\nabla} p - \rho g - g \cdot x \vec{\nabla} \rho \tag{5.8}$$

Also, writing out the viscous stress term is more efficient for the numerical calculations:

$$\vec{\nabla} \tau = \vec{\nabla} \cdot (\mu(\vec{\nabla} \vec{V} + (\vec{\nabla} \vec{V})^T)) = \vec{\nabla} \cdot (\mu \vec{\nabla} \vec{V}) + (\vec{\nabla} \vec{V}) \cdot \vec{\nabla} \mu \tag{5.9}$$

Now we can write the final equation, as stated in Rusche (2002), in discrete form:

$$\frac{\delta}{\delta t}(\rho \vec{V}) + \vec{\nabla} \cdot (\rho \vec{V} \vec{V}) = -\vec{\nabla} p^* + \vec{\nabla} \cdot (\mu \vec{\nabla} \vec{V}) + \vec{\nabla} \vec{V} \cdot \vec{\nabla} \mu - g \cdot x \vec{\nabla} \rho + \sigma \kappa \vec{\nabla} \alpha \tag{5.10}$$

5.1.1 MULES and PISO

InterFOAM uses Multidimensional Universal Limiter for Explicit Solution (MULES) to solve the phase transport equation. This will be further discussed in section 8.2.2. It is hard to find documentation of MULES, but some information can be found in the user-guide given by Greenshields (2015b). Away from the interface the fluids are assumed incompressible. The mass and momentum equation are here solved using the PISO algorithm introduced by Issa (1984). The pressure and velocity are coupled in this area, and the PISO algorithm is well suited for this problem.

InterFoam solves the Continuity equation 4.5, the momentum equation 5.10 and the modified indicator function 5.5 using MULES and PISO. For each time step we solve the equations, according to Damian (2013), as shown below .

- Using VOF-method to find new α and ρ based on previous time-step, i.e. previous velocity field. This part is dependent on MULES.
- Calculating new momentum based on previous velocity field, old pressure and new density (calculated in the first step)
- Calculating pressure based on velocity obtained by previous step.
- The flux used to calculate the pressure is adjusted, as well as the volume in the center of the cell.
- The static pressure is found by the modified pressure $p^* = p - \rho g h$ (explained in section 7.2).

The last three steps are performed 'nCorrectors' times, as set by the user, i.e. PISO-loops.

5.1.2 Challenges with interFoam solver

According to Jasak and Weller (1995), the main disadvantage with the VOF method is that it tries to align the interface with the mesh lines. This is also the experience made in this thesis. Examples of this can be seen in section 8.1.2.

One of the challenges connected to the VOF method is the smearing of the not clearly defined interface (usually over 2-3 cells, see section 8.1). The interface has a tendency to diffuse and cause spurious currents. This usually is a result of an imbalance between pressure gradient and surface tension.

The work done in this thesis (section 8.1) shows that these spurious currents tends to be more present with an unstructured grid compared to a structured grid. This is also in accordance with the work done in Deshpade et al. (2012).

Even though we compress the surface, the interFoam VOF scheme is very sensitive and might give inaccurate interfaces. This can be seen in Case 6 and Case 7 in section 8.1.1. These formations comes from the sharp change in volume fraction gradient at the interface, as discussed by Heyns and Oxtoby (2014), or from the topics discussed above.

Another problem arise due to the discontinuous volume fraction field, namely determination of interface curvature according to Deshpade et al. (2012).

Typical numerical errors are shown in Gopala and Wachem (2008), where the method used by interFoam is compared with other schemes. This article shows the numerical errors on interface without discussing it further than the CFL number limitations. This problem is also experienced by Pringuey (2012), which also found that other CFD packages (namely Gerris (Popinet, 2010)) outperforms OpenFOAM on this issue.

Chapter 6

Numerical set-up

6.1 Meshing

As mentioned in Chapter 3, OpenFOAM comes with built-in mesh generators. However, as an open-source toolbox, several grid generators are developed to fit with OpenFOAM. In this project the grid generator MEGA is used for meshing.

To avoid numerical diffusion, there are some important things to think about when creating a mesh configuration. These are shown in figure 6.1. The larger these properties get, the larger the truncation error gets, i.e. the difference between the partial derivatives in the governing equations and in the discrete approximations.

To save computation time, we want as few cells as possible. Near the point of interest, e.g. near the object, we need a large amount of cells, i.e. the cells need to be very small here. As we want to avoid numerical diffusion, we need to minimize size jumps in areas with large gradients. To avoid small cells being transmitted out in the domain, we need to make a suitable mesh by using different blocks and increments. Meshing is a time consuming and important part of a CFD analysis. The mesh configuration used in this thesis is shown in figure 6.2.

- Size jumps between neighbouring cells
- Non-orthogonality
- Skewness

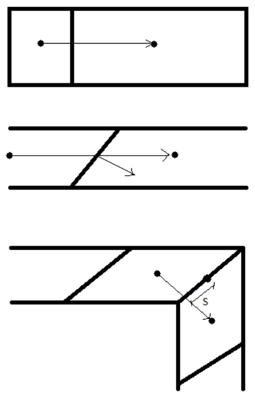

Figure 6.1: Graphical description of properties to avoid for avoiding numerical diffusion

In this thesis the main focus area is in front of the object, as this is where the oil layer should circulate and be kept in place. As the circulation of the oil originates at the top left corner, as marked in figure 6.2, it is important that the grid is well refined in this area.

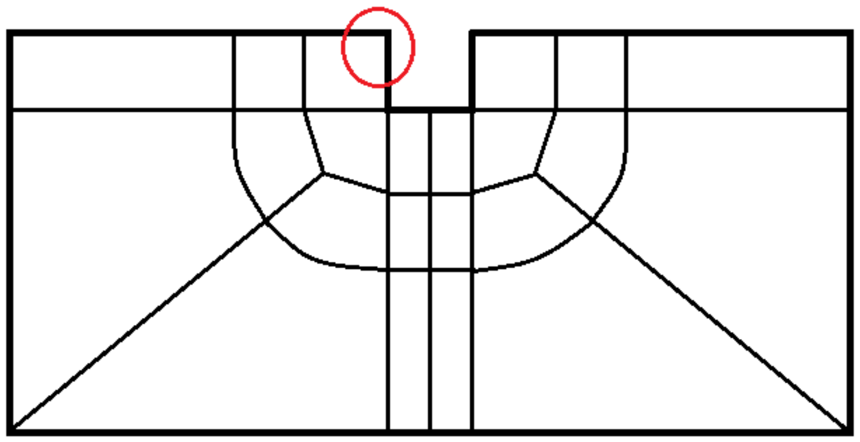

Figure 6.2: Mesh block configuration, with marked area of interest

This mesh configuration lets us refine the mesh in front of the object, without necessarily having to refine the mesh as much behind the object. This saves us a lot of computation time. Figure 6.3, figure 6.4 and figure 6.5 shows the mesh configuration used in this thesis, made in MEGA. From figure 6.5 the increments can easily be seen, as we observe small cells close to the body.

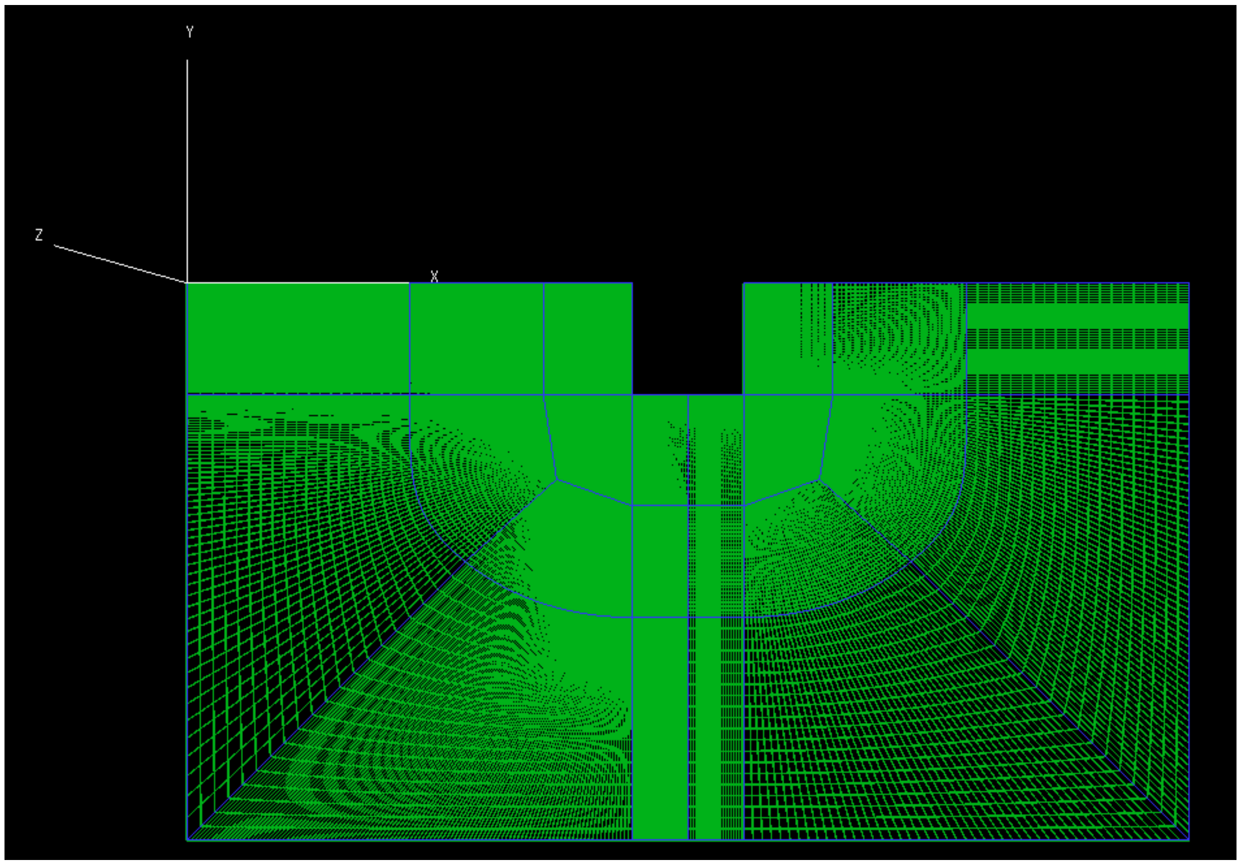

Figure 6.3: Mesh configuration

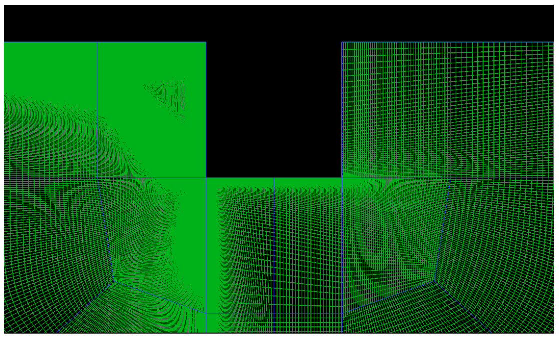

Figure 6.4: Mesh configuration, zoomed in close to the object

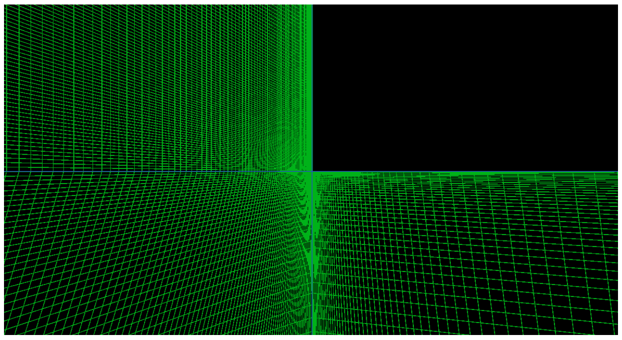

Figure 6.5: Mesh configuration, lower corner of object

6.2 Defining patches

With the generated mesh, we have to assign patches at which we can set the boundary conditions. In this thesis, the respective patches are shown in figure 6.6.

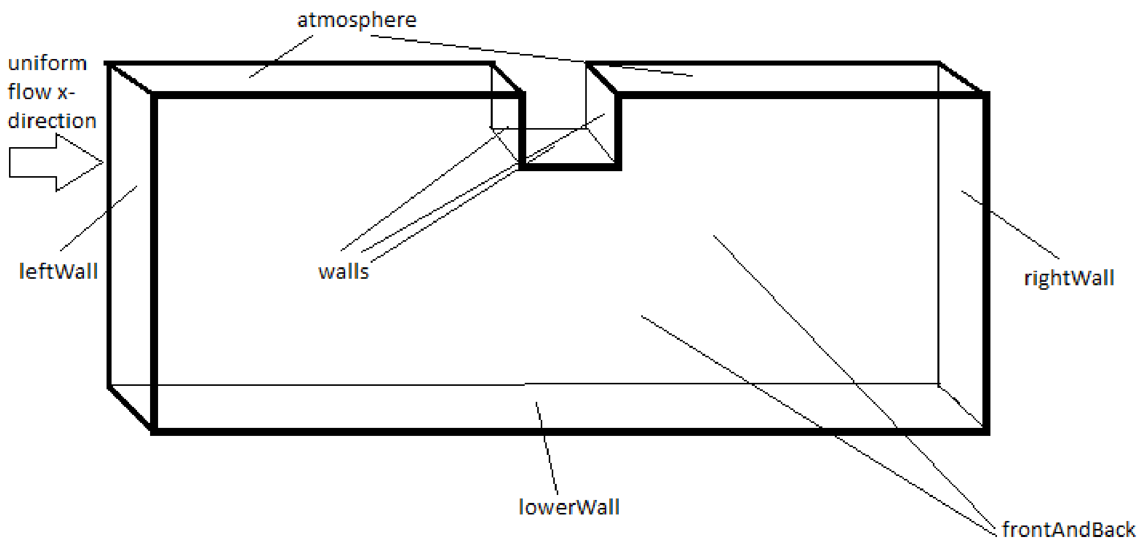

Figure 6.6: Assigned patches in computational domain

For use in OpenFOAM the mesh has to be in 3 dimensions, 3D (for 2D cases the mesh has 1 cell in z-dir. as defined in figure 6.3). We set the front and back planes as type empty, i.e. no solution is generated on these planes.

Chapter 7

One-phase

7.1 Dimensions of domain and boundary conditions

In the upcoming sections we use the following dimensions and boundary conditions (for interFoam solver), except when alternating Re as we then change inlet velocity. α boundary conditions are set in accordance with Rusche (2002).

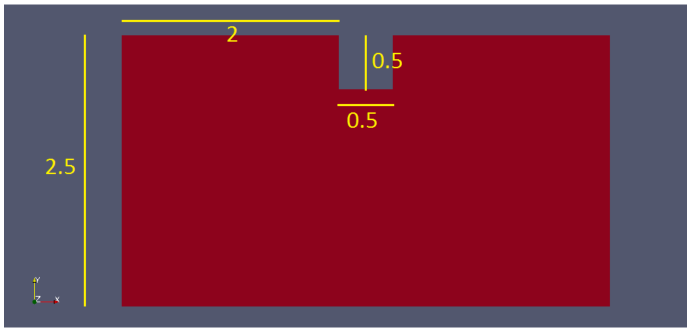

Figure 7.1: Dimensions of computational domain

- leftWall:
  U: fixedValue uniform (1 0 0)
  p_rgh: zeroGradient
- rightWall:
  U: zeroGradient
  p_rgh: fixedValue uniform 0
- atmosphere:
  U: slip
  p_rgh: zeroGradient
- walls:
  U: fixedValue uniform (0 0 0)
  p_rgh: zeroGradient
- lowerWall:
  U: slip
  p_rgh: zeroGradient
- frontAndBack:
  U: empty
  p_rgh: empty

7.2 Verification of solver, interFoam vs. pisoFoam

To give an introduction to the interFoam solver (two-phase) it is interesting to compare it with the pisoFoam solver (one-phase). This is to see that the different solvers give the same solution under the same conditions.

As mentioned in section 5.1 we have to specify the two different phases, i.e. the α-value, either 0 or 1, to the respective phases. This, as well as the initial locations of these fluids are assigned in the file 'setFieldsDict' in the 'system' directory.

For testing the solver we want to keep the conditions equal for both test cases (pisoFoam and interFoam). So in the two-phase setup we set a very small layer of different fluid at the bottom far away from the object, to make the environment as equal to one-phase flow as possible. This is shown in figure 7.2, where the thin layer of additional fluid can be seen at the bottom.

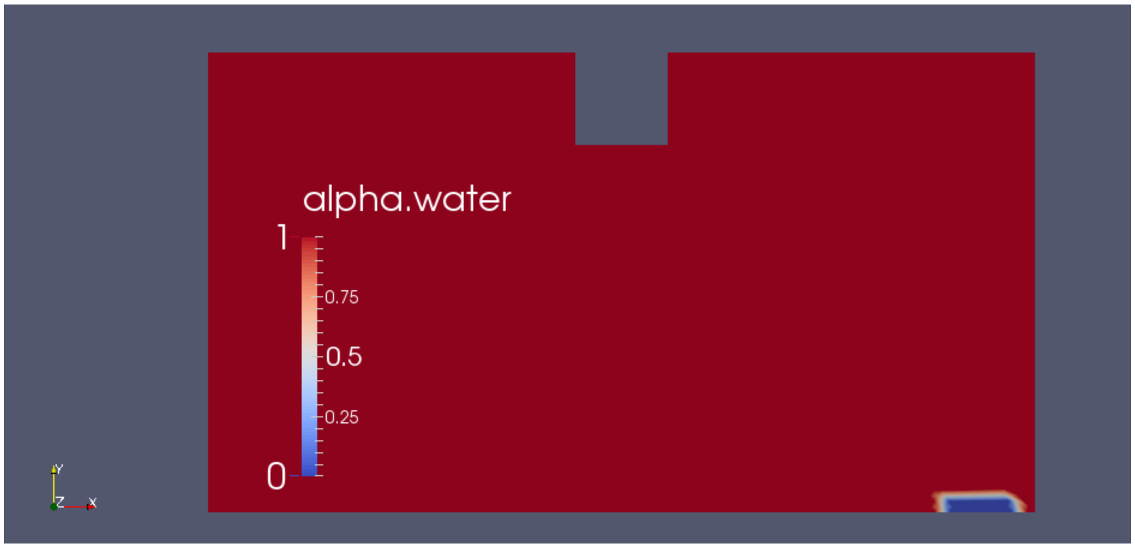

Figure 7.2: Simulation of onephase flow using interFoam solver

In this test case we solve for $Re = 50$. We control Re by kinematic viscosity, ν. Setting $\nu_{water} = 0.01$ for water will therefore give us the desired Re, as long as we keep the inlet velocity, $U = 1$, and the characteristic length, $L = 0.5$. In this case, $\nu_{otherfluid} = 0.01$ and density is set the same as for water, in order to trick the solver to be a one-phase solver.

- **Initially:**
- water: $\nu = 1 \cdot 10^{-6}$

- **Adjusted:**
- water: $\nu = 0.01$

InterFoam and pisoFoam also solves for pressure differently, as stated in section 5.1. InterFoam solves for $p^* = p - \rho g h$. This is, according to OpenFOAM Foundation (2015b), to avoid deficiencies in handling of pressure / buoyant force balance on distorted meshes. To avoid differences in pressure and force coefficients, $\rho g h$ is added in the boundary conditions for the two-phase case. The pisoFoam solver solves pressure as $\frac{m}{s^2}$, so we can compare with interFoam solver by multiplying by ρ in the pisoFoam case.

As seen from the results presented below, the interFoam solver is reliable. Both solvers give the same results.

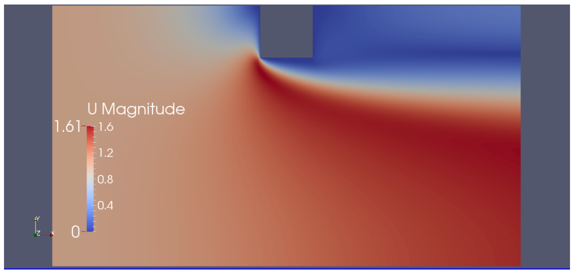

Figure 7.3: Velocity distribution with pisoFoam

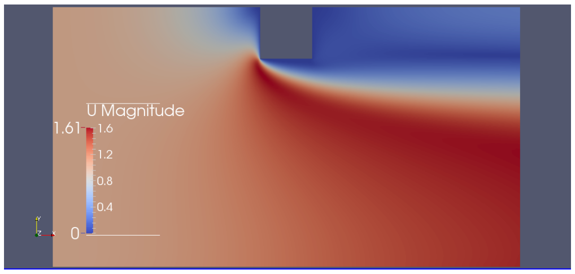

Figure 7.4: Velocity distribution with interFoam

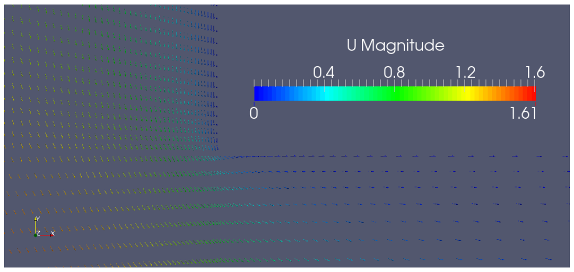

Figure 7.5: Velocity vectors with pisoFoam, lower left corner

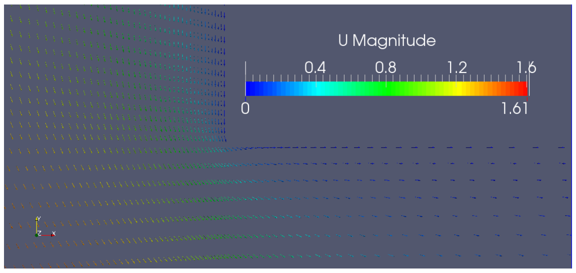

Figure 7.6: Velocity vectors with interFoam, lower left corner

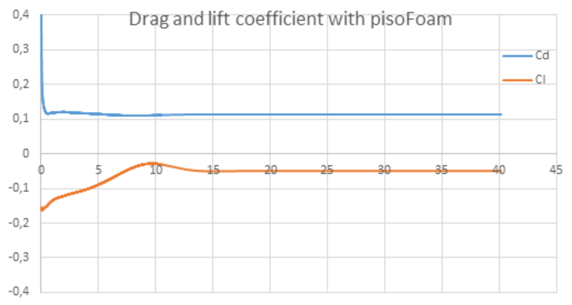

Figure 7.7: Drag and lift coefficients with pisoFoam

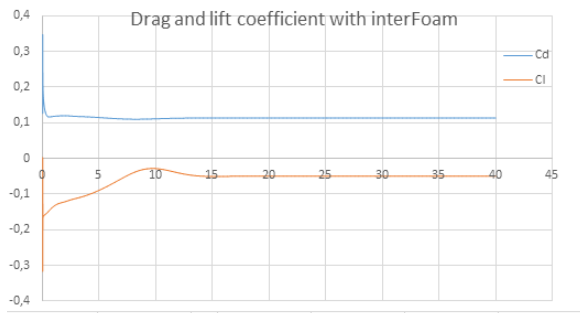

Figure 7.8: Drag and lift coefficients with interFoam

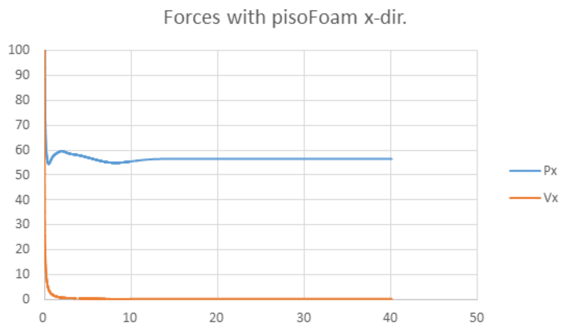

Figure 7.9: Pressure and viscous forces with pisoFoam, in x-dir.

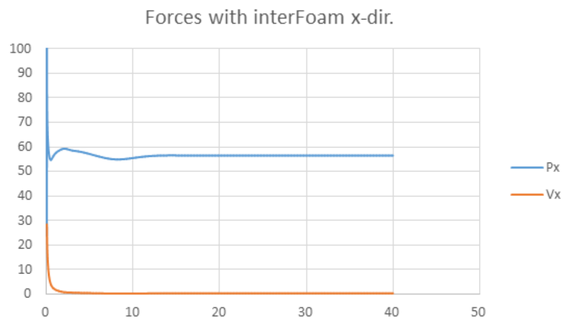

Figure 7.10: Pressure and viscous forces with interFoam, in x-dir.

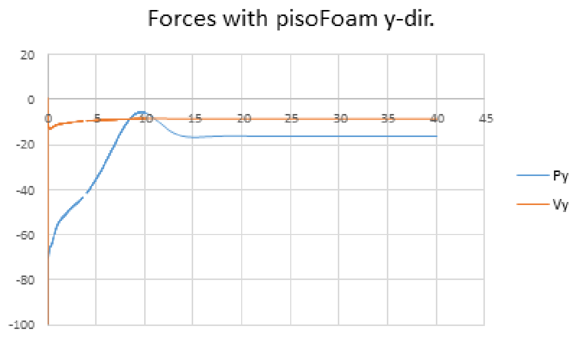

Figure 7.11: Pressure and viscous forces with pisoFoam, in y-dir.

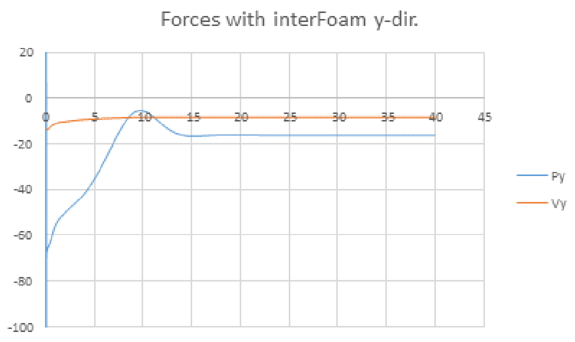

Figure 7.12: Pressure and viscous forces with interFoam, in y-dir.

7.2.1 Comparison with previous literature

A simulation, using FLUENT, is done by Amini and Schleiss (2009). To compare the result we set up the same domain. In Amini and Schleiss (2009) they gradually increase the inflow velocity over the inlet, with a mean velocity 0.2 cm/s. In our case we set the inlet velocity 0.2 cm/s over the inlet.

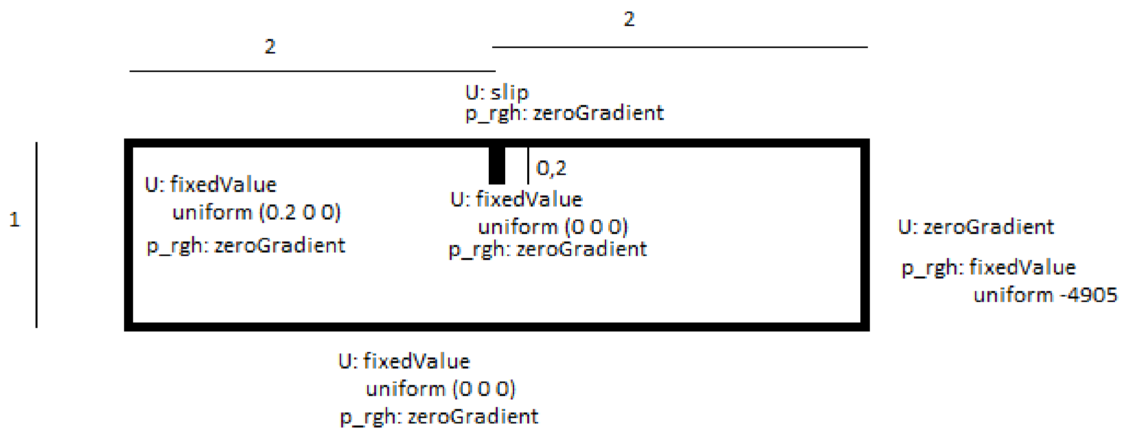

Figure 7.13: Numerical setup for comparison with previous literature

The results presented below fit well with the results from Amini and Schleiss (2009).

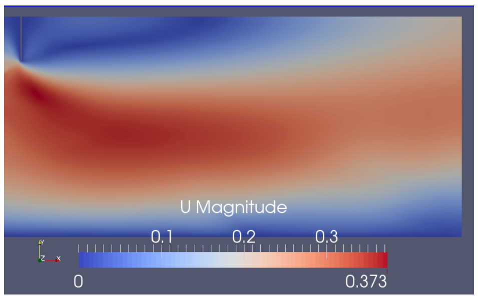

Figure 7.14: Velocity for comparison with previous literature

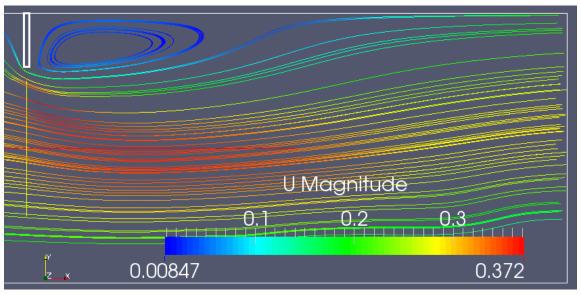

Figure 7.15: Streamlines for comparison with previous literature

7.3 circulation of fluid

In order to contain the oil we need a well refined grid that preserve the circulation ahead of the object. We check this by running simulations with only water. The area of interest is marked in figure 7.16.

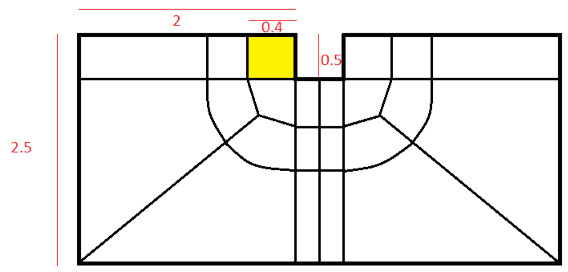

Figure 7.16: Area of interest for containing oil

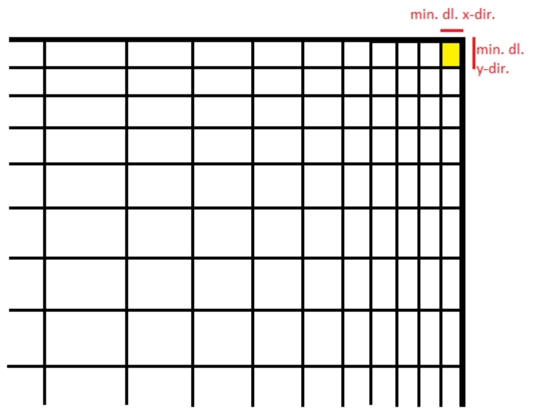

Figure 7.17: Mesh refinement top left corner of object

We run the interFoam solver as in section 7.2, i.e. with a small layer of other fluid at the bottom far away from the surface and object, so that it does not influence the flow. If we zoom in at the top left corner of the object, the mesh looks as in in figure 7.17. We do a convergence study to find a sufficient grid refinement. We run the following cases, for the box in front of object, i.e. the marked box in figure 7.16:

- **Grid 1**
- horizontal:180 cells, min dl.=0.000014
- vertical:300 cells, min dl.=0.000028
- *SUFFICIENT*
- **Grid 2**
- horizontal:160 cells, min dl.=0.00003
- vertical:300 cells, min dl.=0.000028
- *INSUFFICIENT*
- **Grid 3**
- horizontal:144 cells, min dl.=0.000057
- vertical:300 cells, min dl.=0.000028
- *INSUFFICIENT*
- **Grid 4**
- horizontal:124 cells, min dl.=0.000125
- vertical:300 cells, min dl.=0.000028
- *INSUFFICIENT*
- **Grid 5**
- horizontal:112 cells, min dl.=0.0002
- vertical:300 cells, min dl.=0.000028
- *INSUFFICIENT*
- **Grid 6**
- horizontal:80 cells, min dl.=0.000726
- vertical:300 cells, min dl.=0.000028
- *INSUFFICIENT*

From the convergence study we see that we need 180 cells (min dl.=0.000014) in horizontal direction to preserve the circulation of fluid ahead of the object.

To reach time 42 with Grid 1, the computation time was 80 hours on 4 nodes. Adding more nodes only makes the computation time longer, as there will be few cells per node.

Below, the velocity vectors for Grid 1, Grid 2 and Grid 3 (with a reference box, i.e. the pink box), in the upper left corner of the object, are shown. It is clearly seen that only Grid 1 preserve the circulation due to to the no-slip condition at the wall. The other cases are not well enough refined close to the wall.

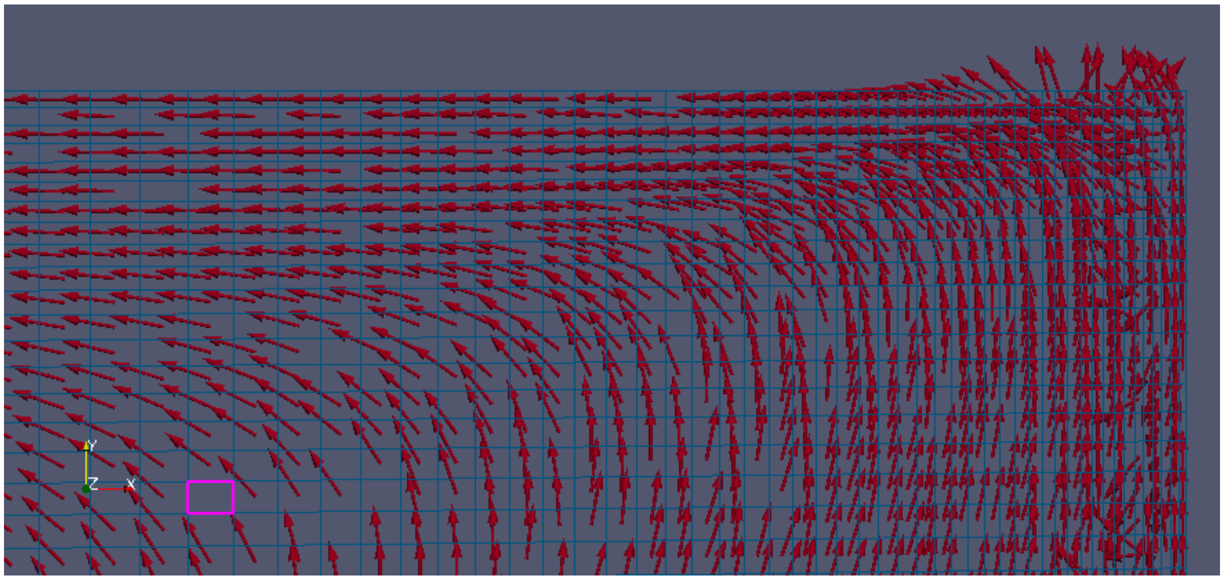

Figure 7.18: Velocity vectors for Grid 1

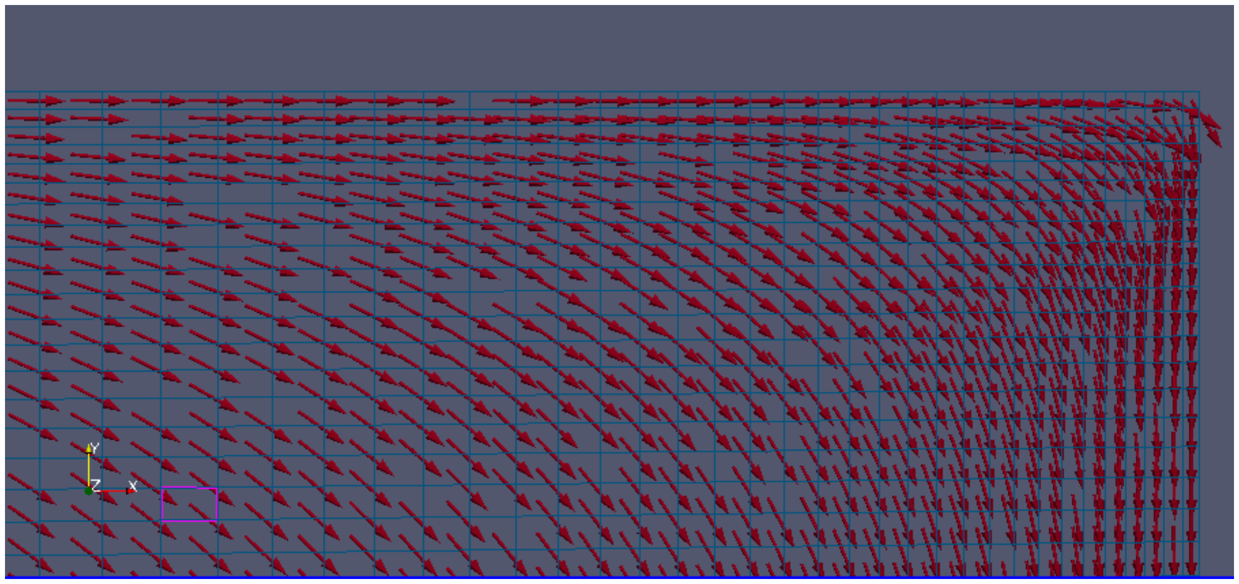

Figure 7.19: Velocity vectors for Grid 2

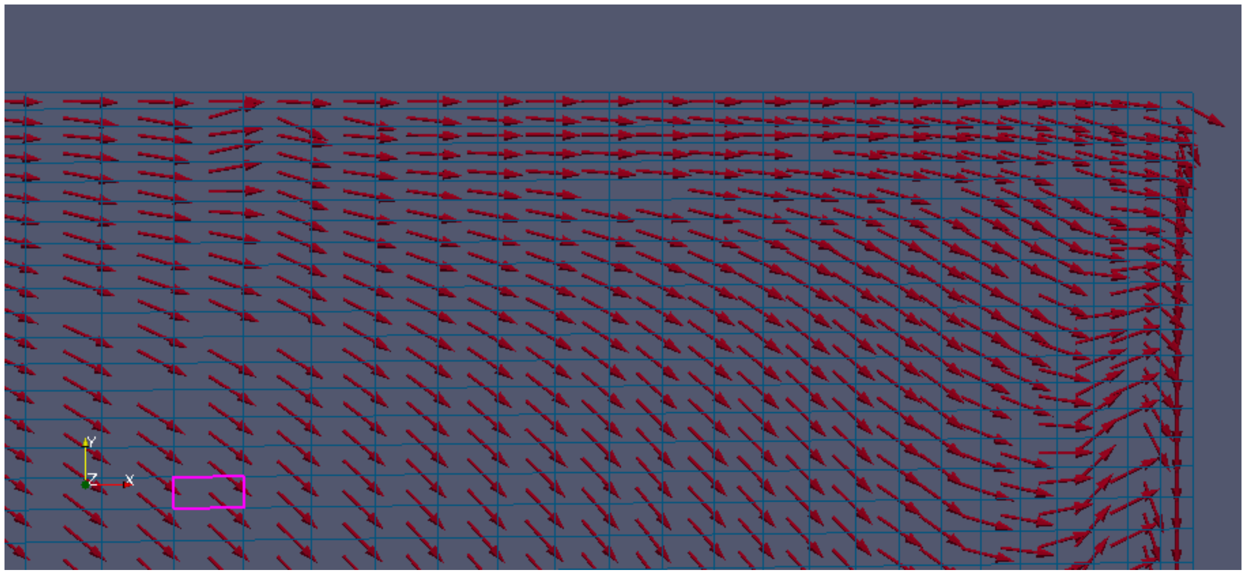

Figure 7.20: Velocity vectors for Grid 3

In Grid 4 we can see the circulation, but it does not seem to be well enough refined, as seen in picture 7.22. The center of the circulation is not well refined, so the velocity vectors act strange in this area.

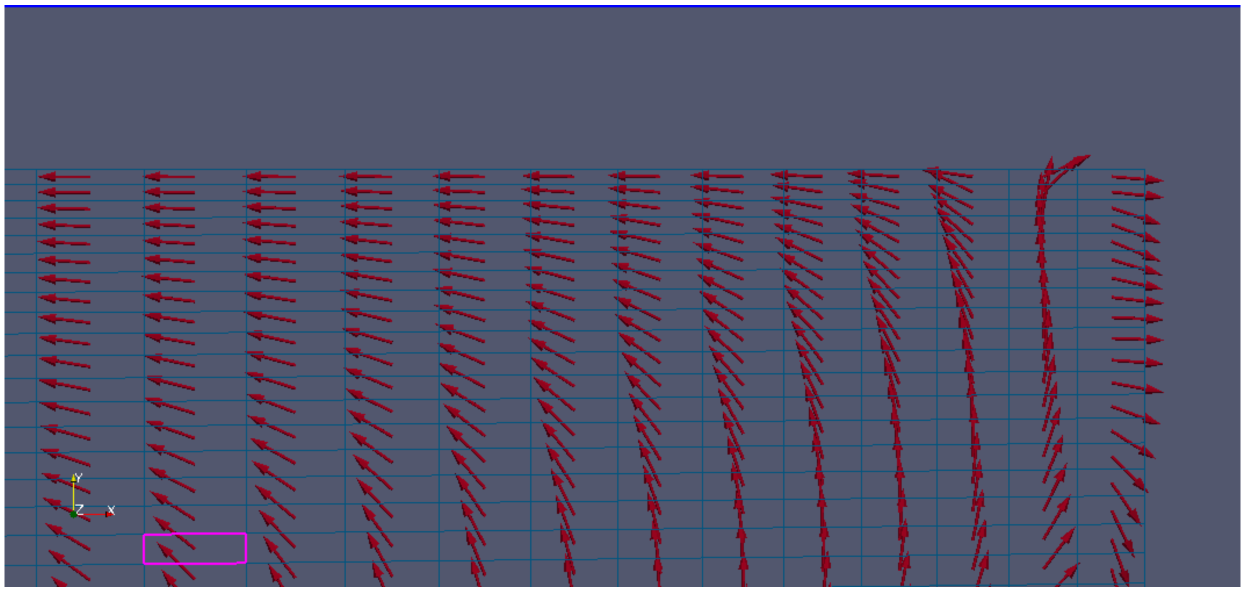

Figure 7.21: Velocity vectors for Grid 4

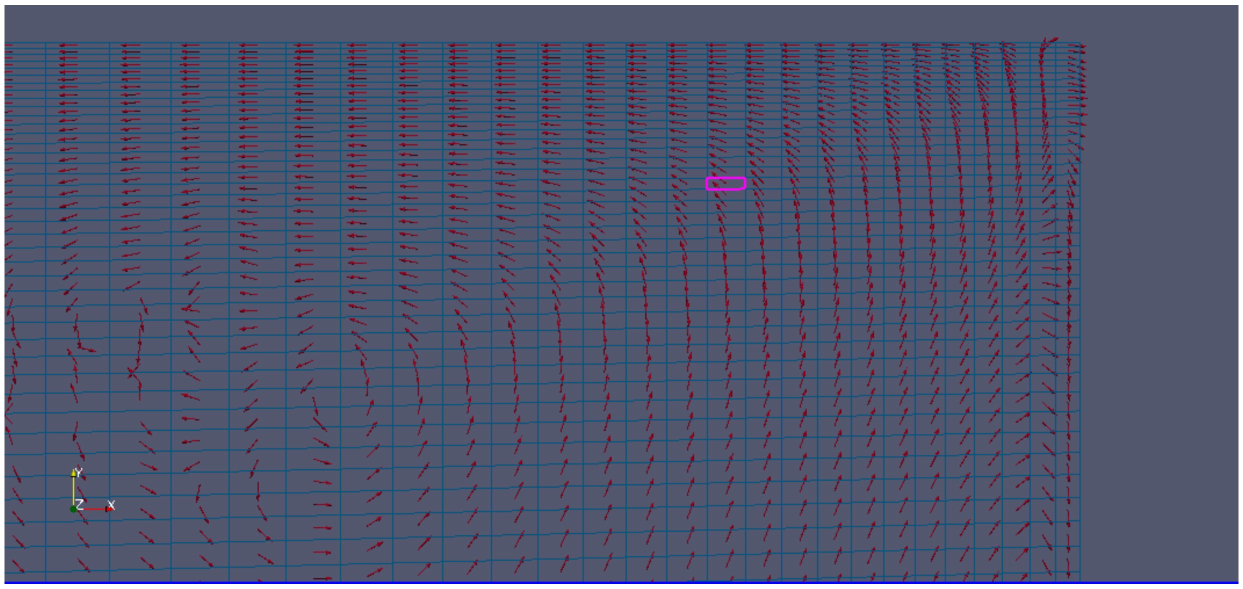

Figure 7.22: Velocity vector for Grid4, shows not well enough refined vortices

Grid 5 and Grid 6 acts in the same manner as Grid 2 and Grid 3.

The pictures below show the circulation in front of the object. It exists in every time-step for Grid 1, but seems to move around. The times chosen below are representable for the first time steps also. Some other time steps can be seen in Appendix, (A.1).

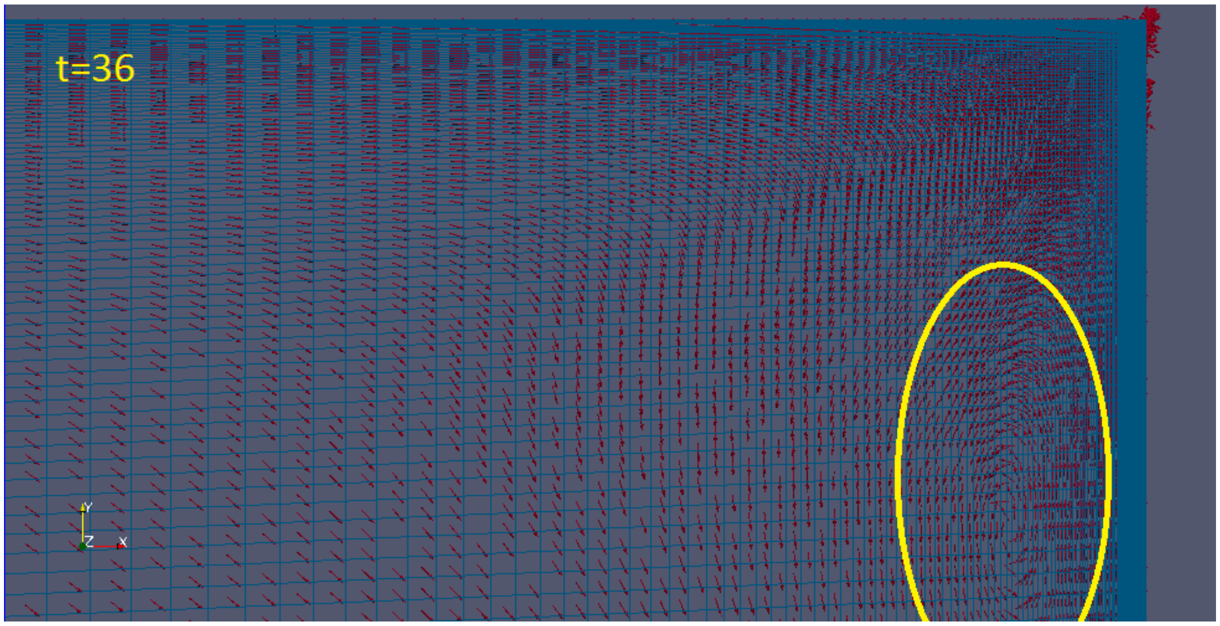

Figure 7.23: Grid 1 velocity vectors at time 36

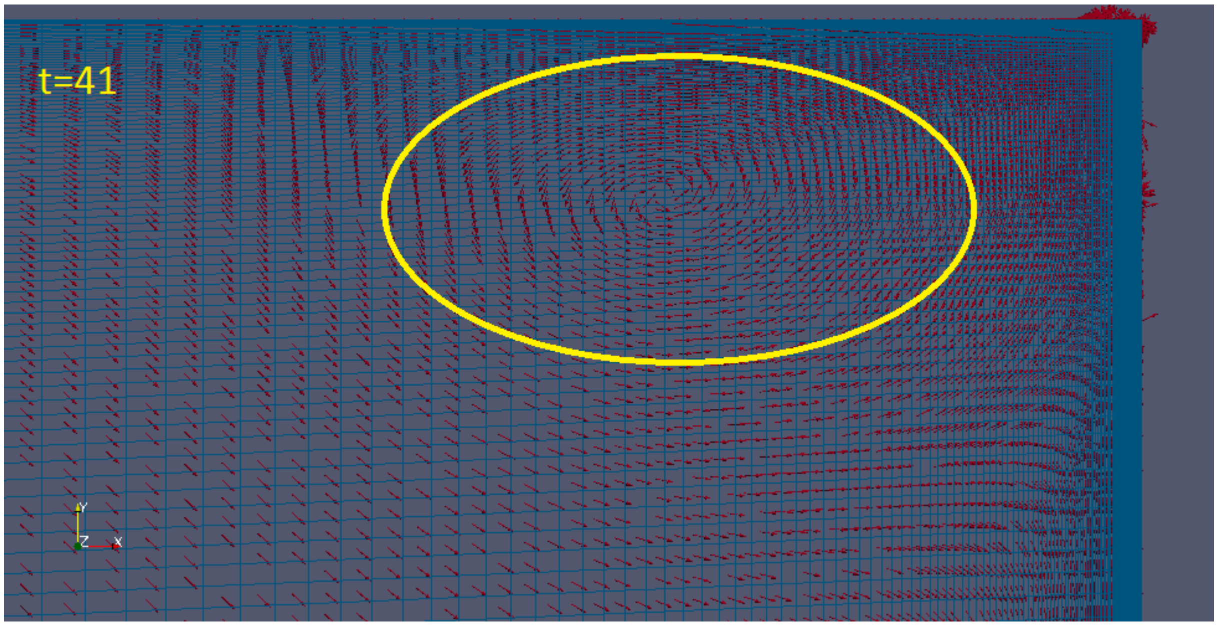

Figure 7.24: Grid 1 velocity vectors at time 41

7.3.1 Circulation at different Reynolds number

In this section we test how Reynolds number influence the vortices ahead of the object. The results are presented below, for Re=50, Re=100 and Re=150.

The pictures below are zoomed in. To see the relevance between them the same cell is marked in the pictures. The marked cell has coordinates (upper right corner) (1.98003 -0.0189668 0).

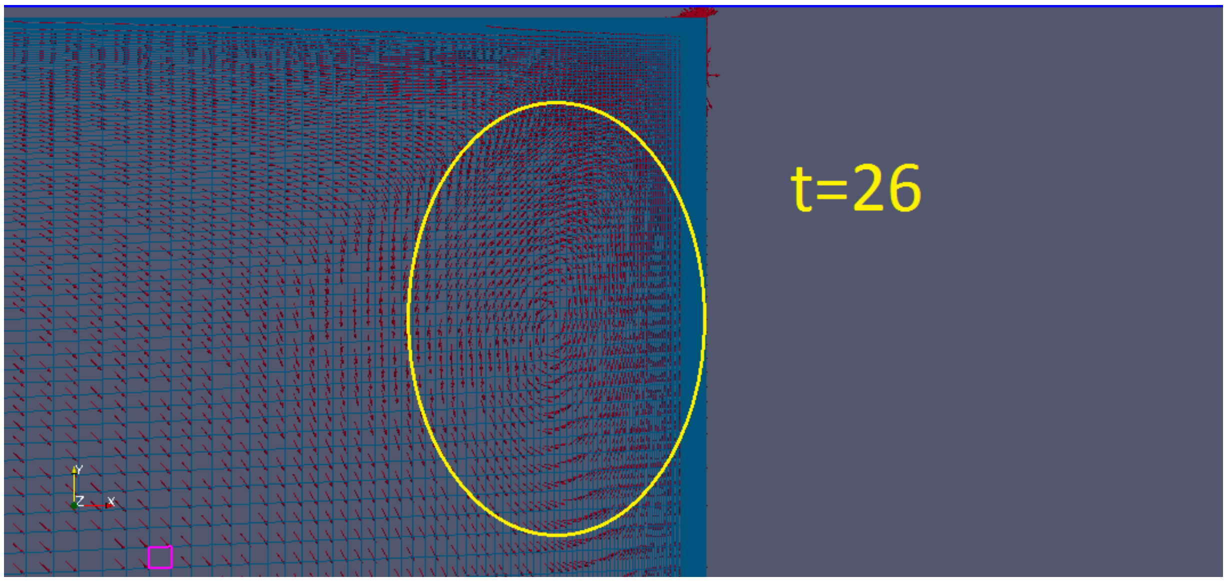

Figure 7.25: Re=50 velocity vectors at time 26

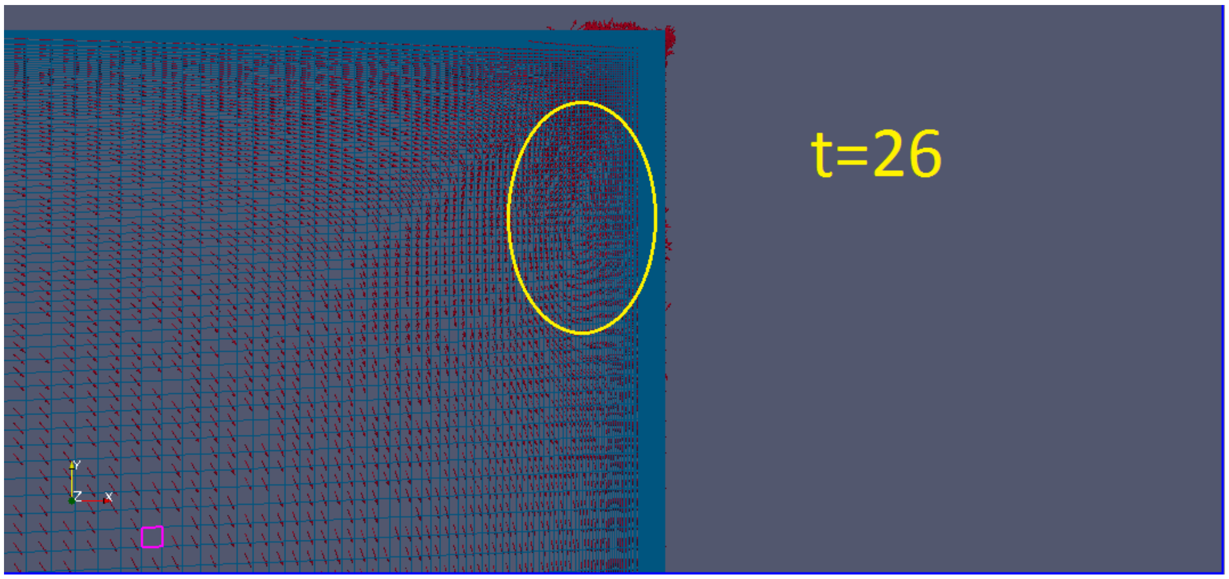

Figure 7.26: Re=100 velocity vectors at time 26

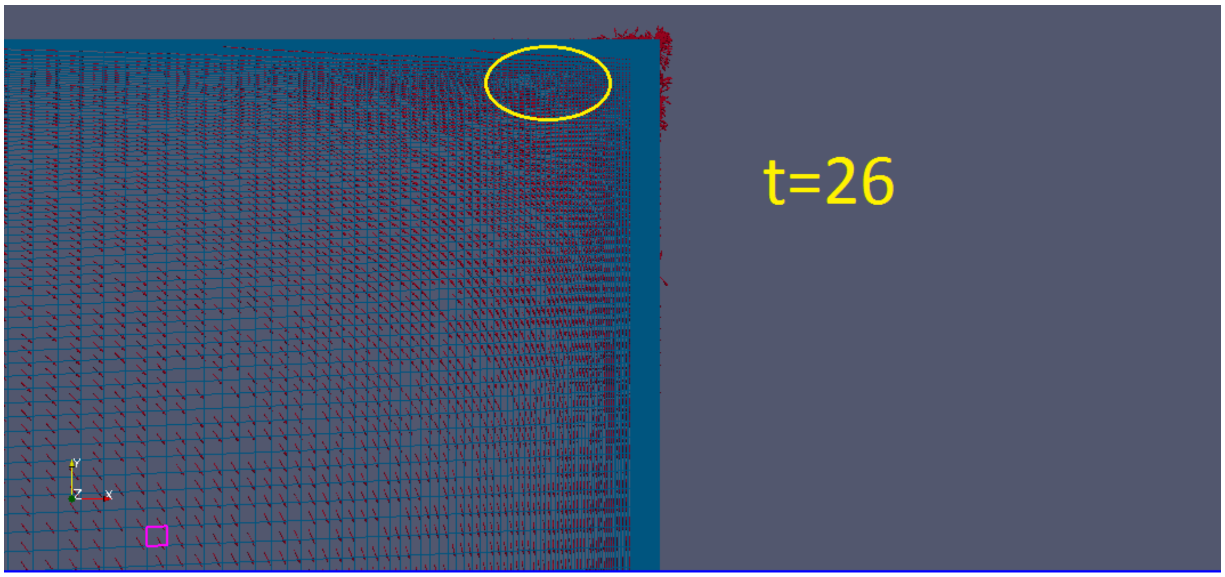

Figure 7.27: Re=150 velocity vectors at time 26

As seen from the pictures above, a higher Reynolds number suppress the vortices ahead of the object. At Reynolds number 300, these vortices are totally vanished with this mesh. Circulation at more time steps can be found in Appendix, (A.2).

7.3.2 Circulation at different angles

We test the influence on the circulation of angling the object. The coordinates of the edges can be calculated by geometric considerations, as shown in figure 7.28.

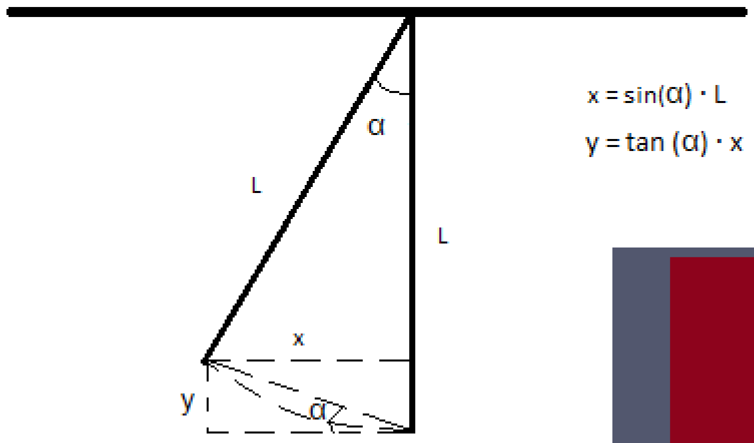

Figure 7.28: Geometrical considerations for angling the boom

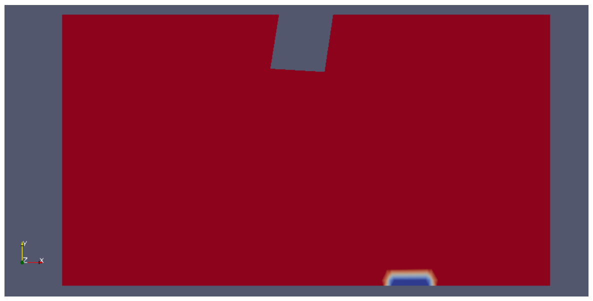

Figure 7.29: Domain with angle=10 degrees

We use the same grid as Grid 1 in section 7.3, and test 3 additional cases, with angle defined as in figure 7.28 and reference box coordinate (1.98 , -0.00556);

- 0 degree
- 5 degrees
- 10 degrees
- -5 degrees

The velocity vectors for the different cases are presented in figure 7.30, 7.31, 7.32 and 7.33. The streamlines are presented in section 7.3.2.

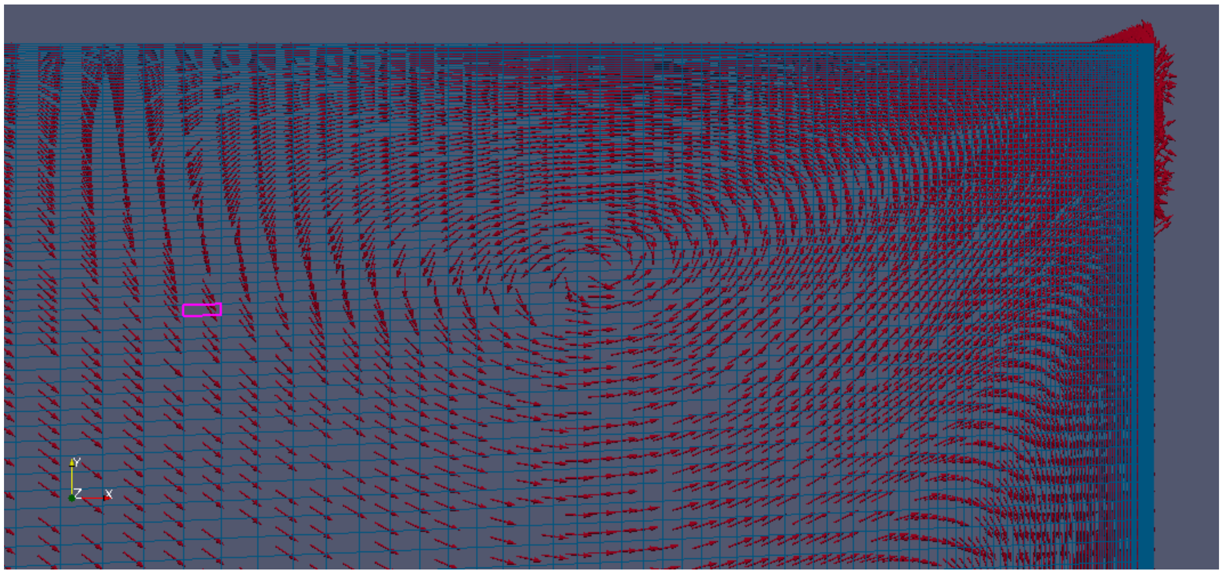

Figure 7.30: Velocity vectors at Re=50 and angle=0 degree

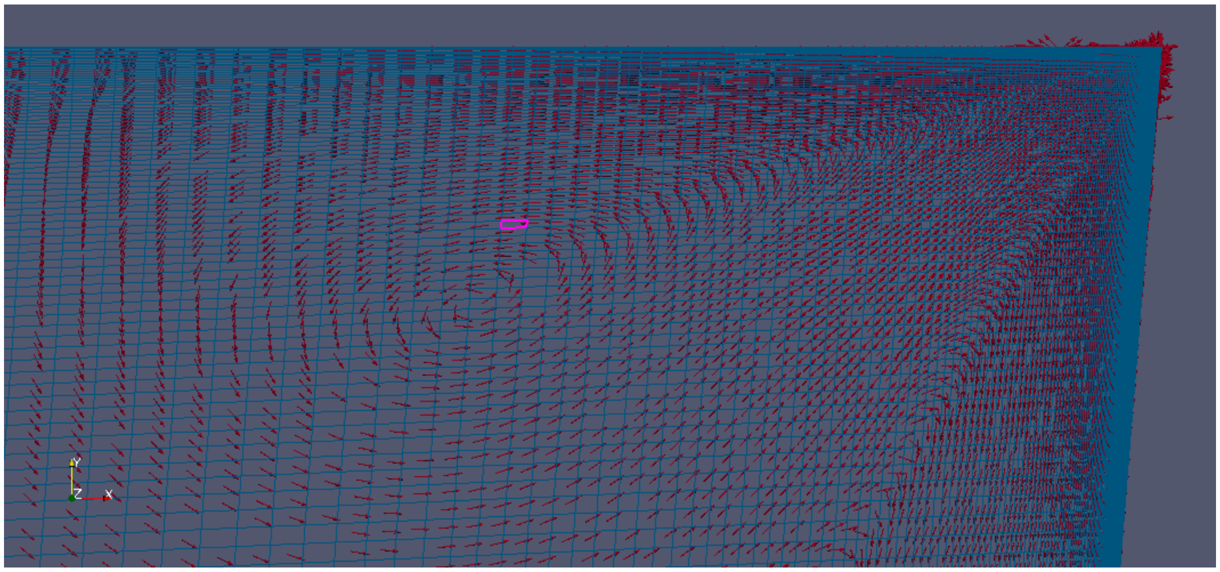

Figure 7.31: Velocity vectors at Re=50 and angle=5 degrees

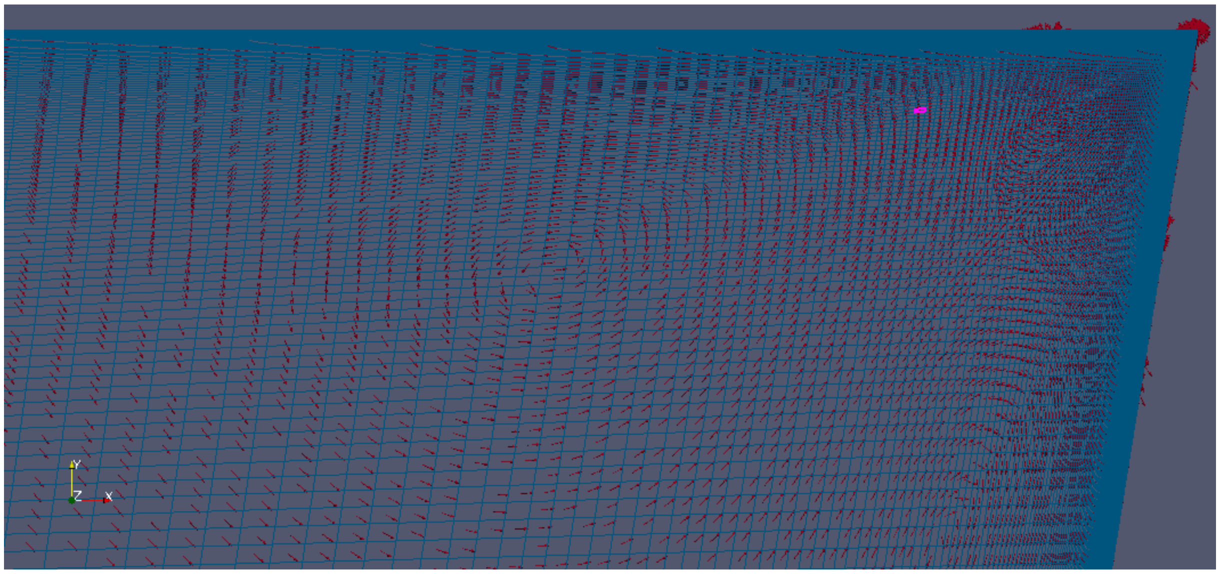

Figure 7.32: Velocity vectors at Re=50 and angle=10 degrees

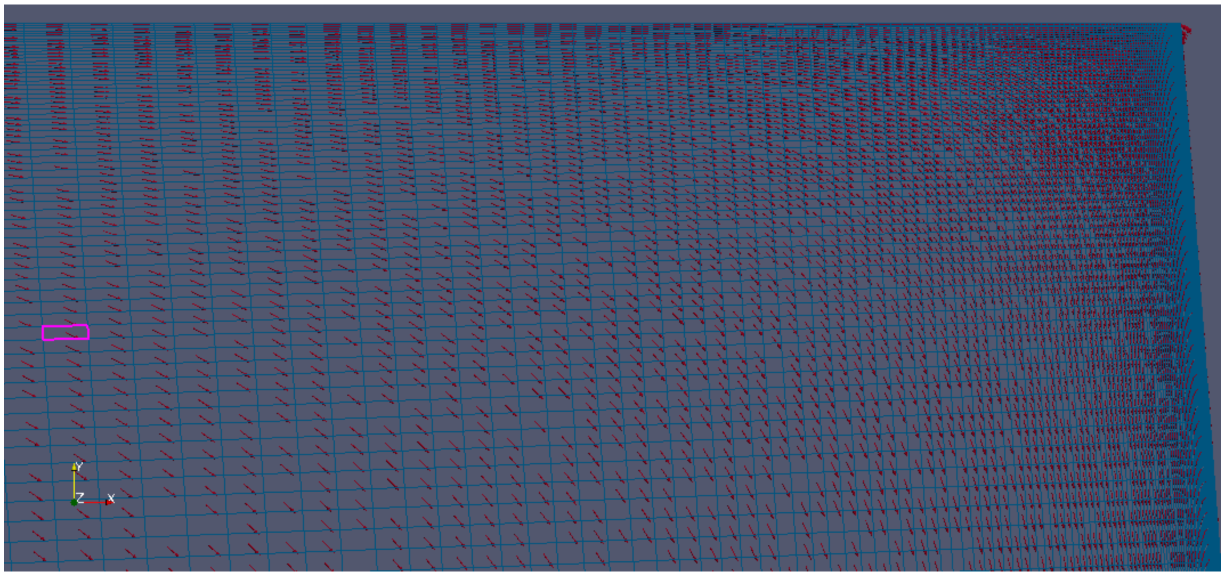

Figure 7.33: Velocity vectors at Re=50 and angle=-5 degrees

It is seen by the figures that a positive angle increase the area of circulation. A larger angle gives a larger area of circulation. On the other hand, a negative angle does not preserve the circulation from 0 degrees.

CPU time is a critical factor in these calculations. A larger area of circulation seems to need more CPU time, as seen in the table below for running on 4 nodes.

On average 1 second of simulation takes:

- 0 degree: 1.9 hours
- 5 degrees: 6.7 hours
- 10 degrees: 17.2 hours

Streamlines

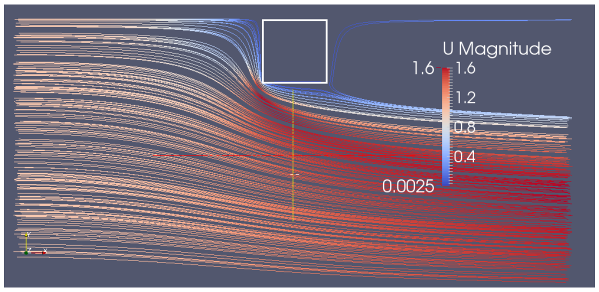

Figure 7.34: Streamlines, Re=50, angle=0 degree

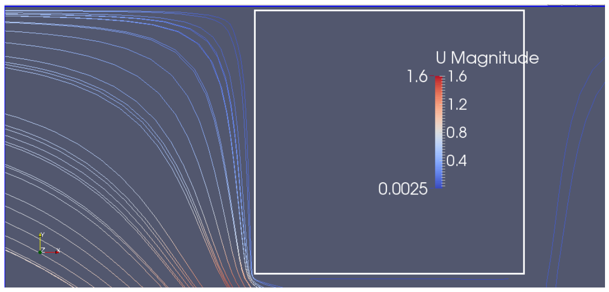

Figure 7.35: Streamlines, Re=50, angle=0 degree zoomed in

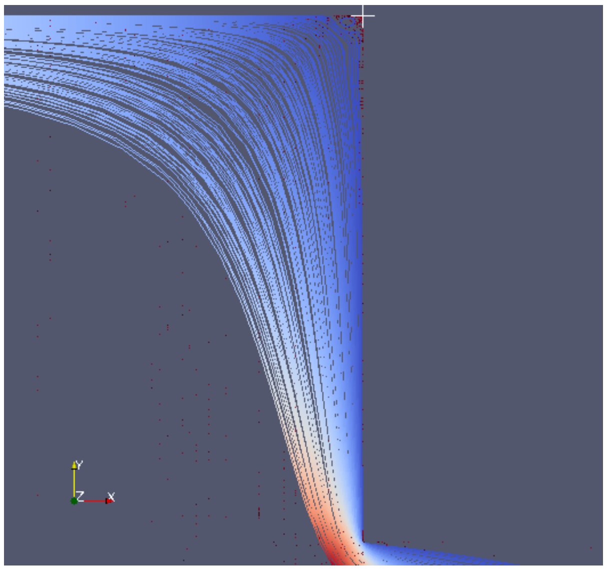
Figure 7.36: Streamlines showing area of circulation, Re=50, angle=0 degree

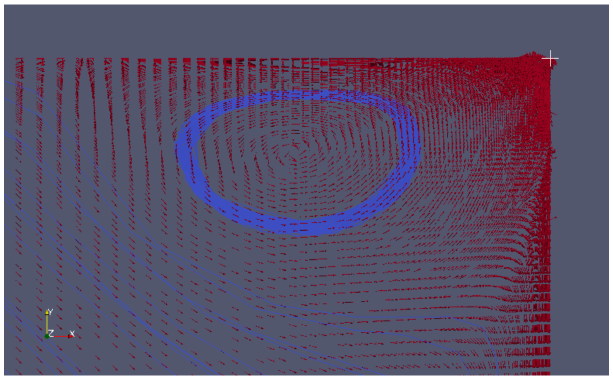
Figure 7.37: Streamlines showing area of circulation, Re=50, angle=0 degree zoomed in

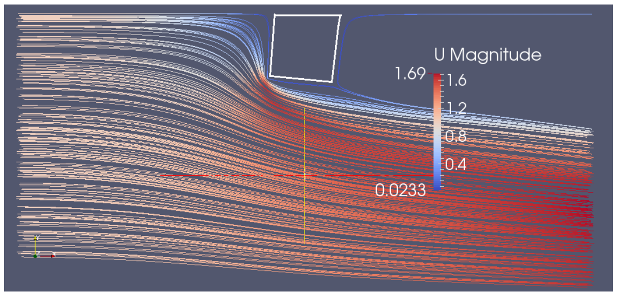

Figure 7.38: Streamlines, Re=50, angle=5 degrees

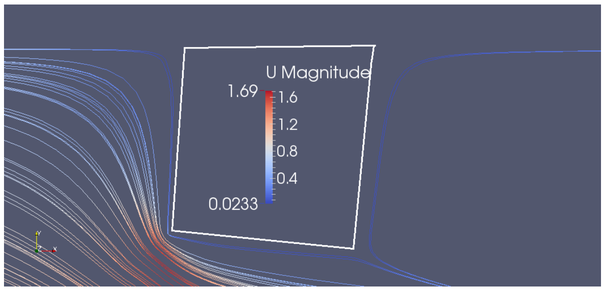

Figure 7.39: Streamlines, Re=50, angle=5 degrees zoomed in

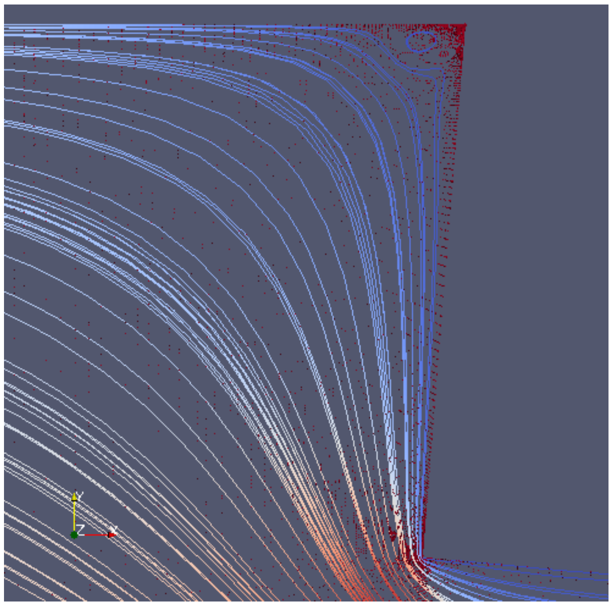

Figure 7.40: Streamlines showing area of circulation, Re=50, angle=5 degrees

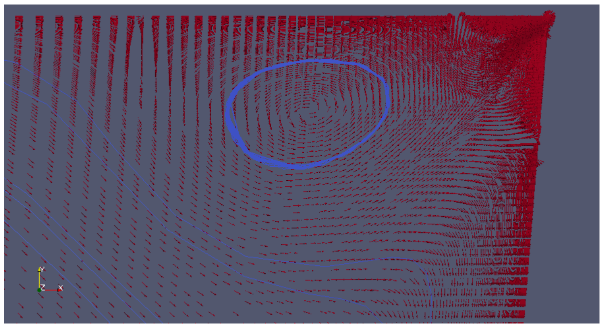

Figure 7.41: Streamlines showing area of circulation, Re=50, angle=5 degrees zoomed in

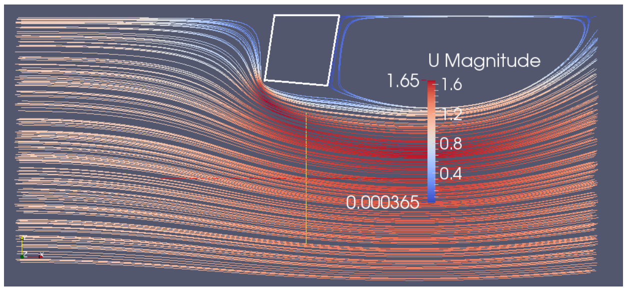

Figure 7.42: Streamlines, Re=50, angle=10 degrees

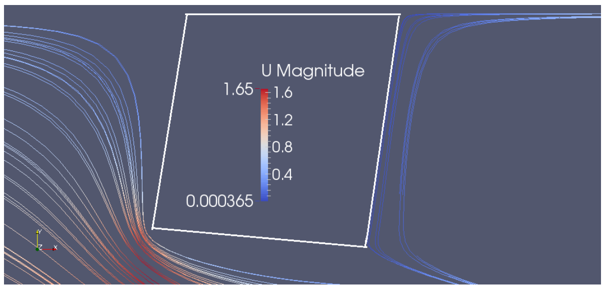

Figure 7.43: Streamlines, Re=50, angle=10 degrees zoomed in

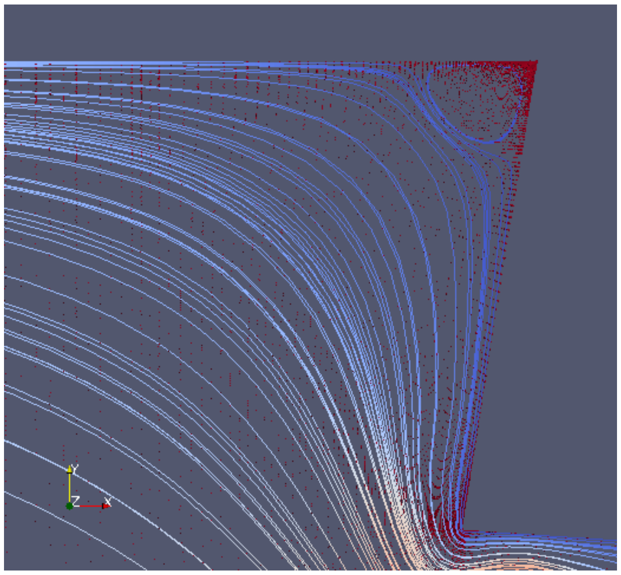

Figure 7.44: Streamlines showing area of circulation, Re=50, angle=10 degrees

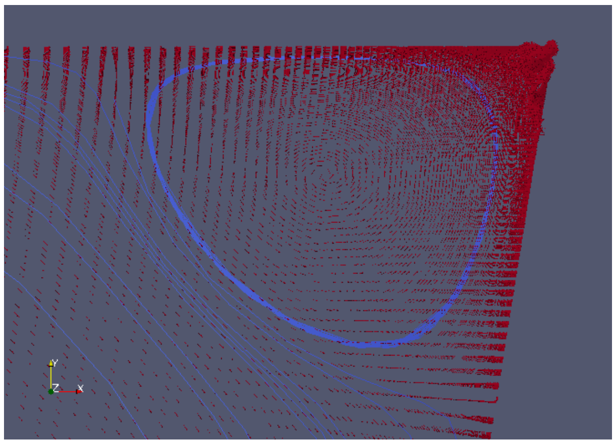

Figure 7.45: Streamlines showing area of circulation, Re=50, angle=10 degrees zoomed in

Chapter 8

Two-phase

When introducing an oil layer, a study of the interaction between oil and water is important. We first start of with a channel case, to see how different grid configurations, density etc. affects the interface region. Thereby we introduce an oil layer to the study done in chapter 7.

8.1 Channel flow

By running a case with no object, i.e. a channel flow, the effects of grid on the interaction between oil and water becomes more clear. The setup for this case is shown in figure 8.1.

In these cases, the two consecutive sections, we keep the properties of oil and water constant as shown below. These are scaled values, to adjust reynolds number as desired.

The real values corresponds to $\nu_{water} = 1e{-}6\frac{m^2}{s}$ and $\nu_{oil} = 3e{-}6\frac{m^2}{s}$. The oil value is an estimate based on Crude oil 48^0 API, retrieved from The Engineering Toolbox (2015).

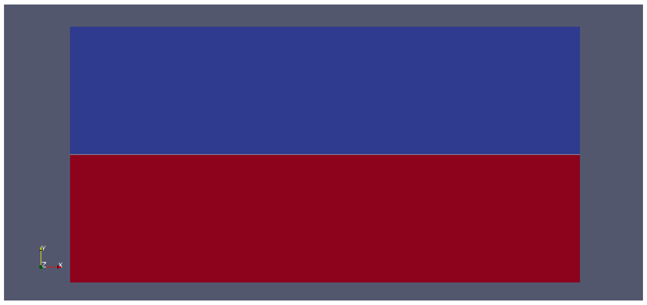
Figure 8.1: Setup for case without object, channel flow

- water: $\rho_{water} = 1000, \nu_{water} = 0.01$
- oil: $\rho_{oil} = 900, \nu_{oil} = 0.03$

- leftWall:
  U: fixedValue uniform (1 0 0)
  p_rgh: zeroGradient
- rightWall:
  U: zeroGradient
  p_rgh: fixedValue uniform 0
- atmosphere:
  U: slip
  p_rgh: zeroGradient
- lowerWall:
  U: slip
  p_rgh: zeroGradient
- frontAndBack:
  U: empty
  p_rgh: empty

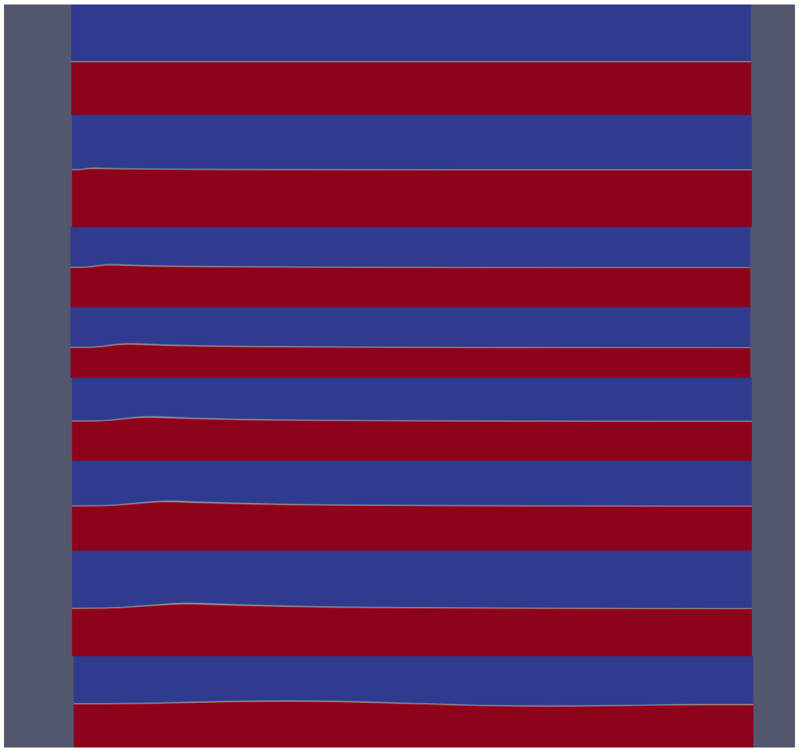
Figure 8.2: Channel flow, from t=0 to t=40

There seems to be made a wave at the inlet, which continues out in the flow, before the fluid stabilizes at the last picture of figure 8.2. The grid for this case is shown in figure 8.3 and figure 8.4.

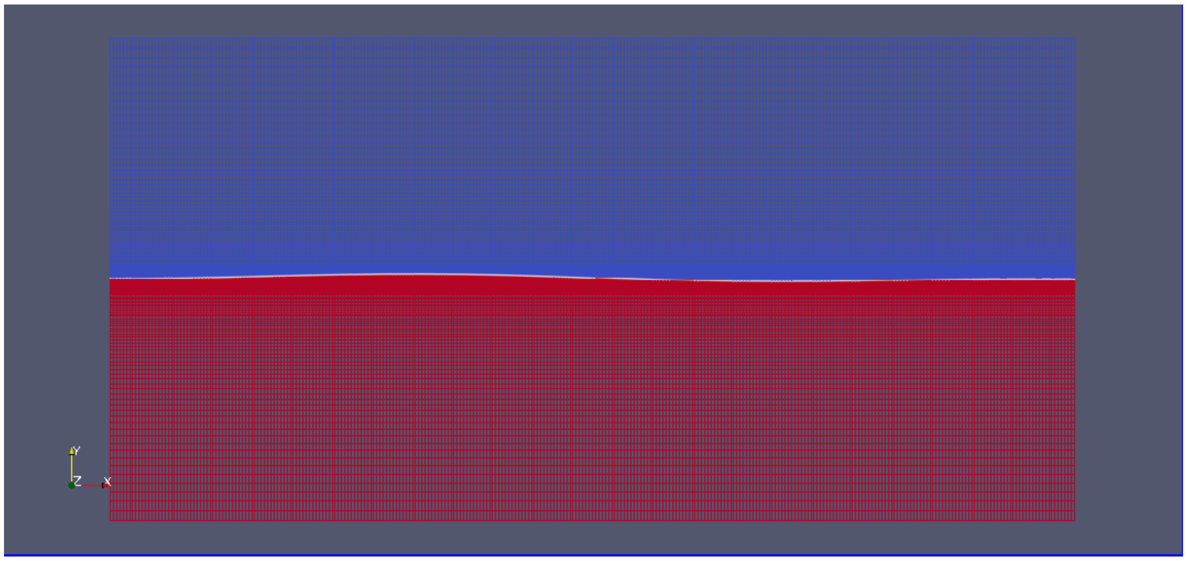

Figure 8.3: Grid for channel flow

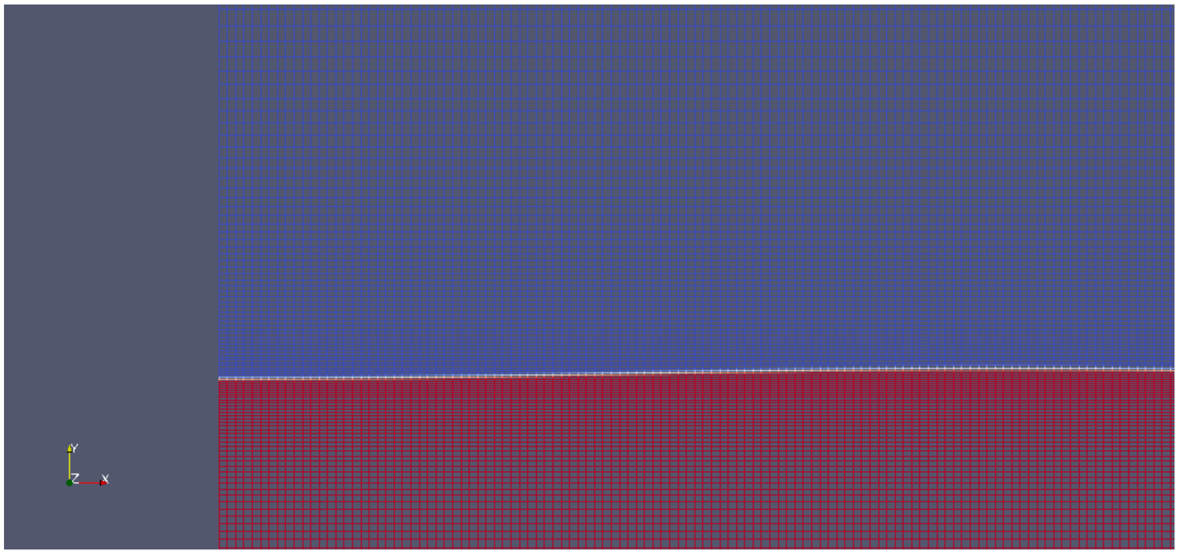

Figure 8.4: Grid at inlet for Channel flow

8.1.1 Structured grid

We want to test the effect of cell size near the layer between oil and water. There is no skewness in this case.

The different cases which are run can be seen below:

Case 1:
- 200 cells in horizontal direction
- 88 cells in vertical direction
- increment vertical dir. 1.02
- min. dl. 0.071940

Case 2:
- 200 cells in horizontal direction
- 134 cells in vertical direction
- increment vertical dir. 1.02
- min. dl. 0.037106

Case 3:
- 300 cells in horizontal direction
- 88 cells in vertical direction
- increment vertical dir. 1.02
- min. dl. 0.071940

Case 4:
- 300 cells in horizontal direction
- 134 cells in vertical direction
- increment vertical dir. 1.04
- min. dl. 0.016246

Case 5:
- As in case 4, only refined a lot near inlet

Case 1 to Case 5 all show the same flow pattern as in figure 8.2, i.e. they all seem well enough refined around the interface region. These cases can be seen in Appendix (A.4). The two cases below are on the contrary not well enough refined. We see, in figure 8.5 and figure 8.6, that the numerical diffusion will evolve out in the simulation. So it is extremely important to avoid numerical diffusion like in these cases. Case 6 is the first time we observe numerical diffusion, which does not occur with a finer mesh. For Case 7 the numerical diffusion is clearly observed. In Case 6 it is a bit hard to see, but the interface seem to diffuse close to the wave.

Case 6:
- 200 cells in horizontal direction
- 58 cells in vertical direction
- increment vertical dir. 1.02
- min. dl. 0.128892

Case 7:
- 200 cells in horizontal direction
- 48 cells in vertical direction
- increment vertical dir. 1.02
- min. dl. 0.164356

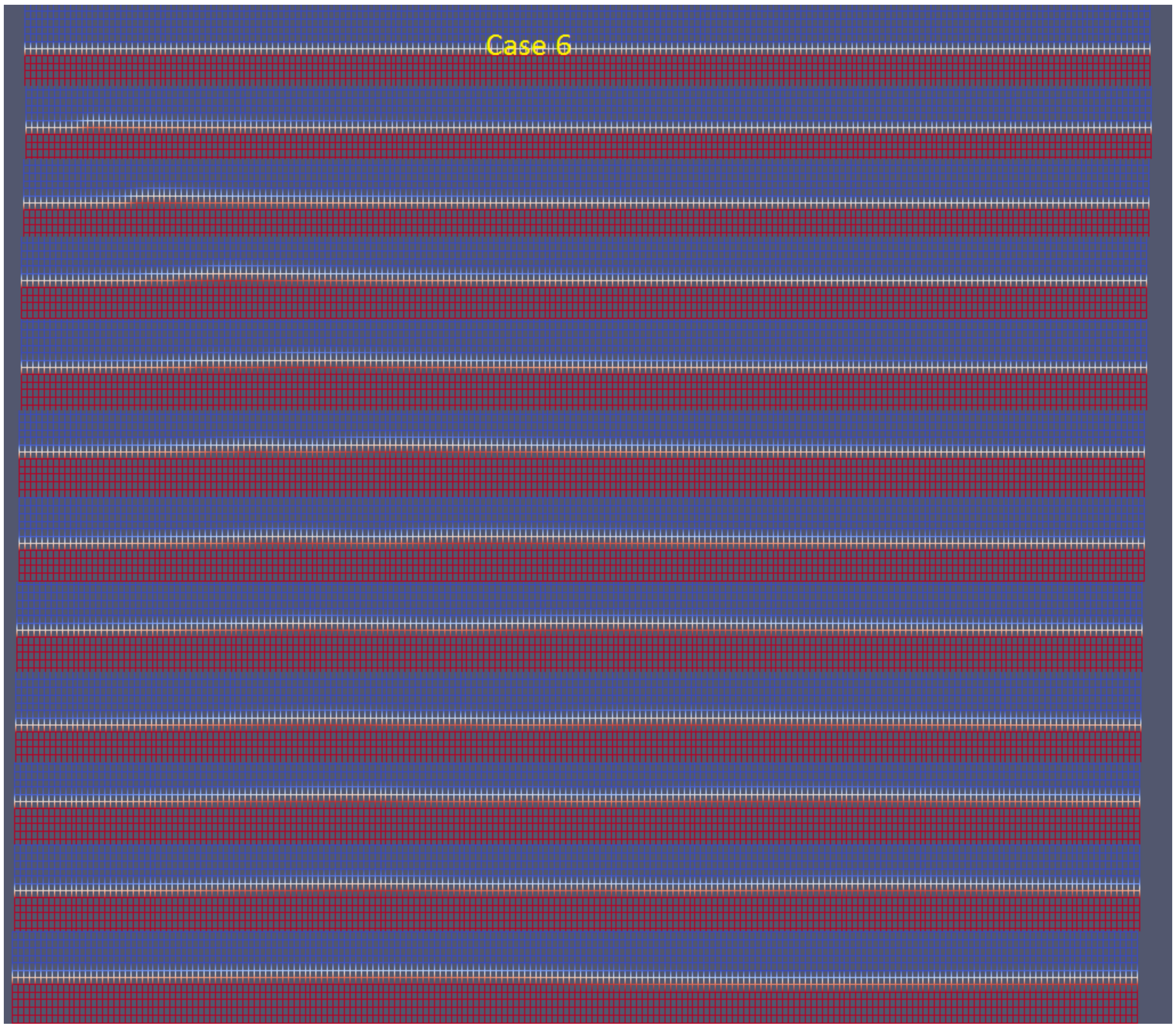

Figure 8.5: Numerical diffusion seen for Case 6

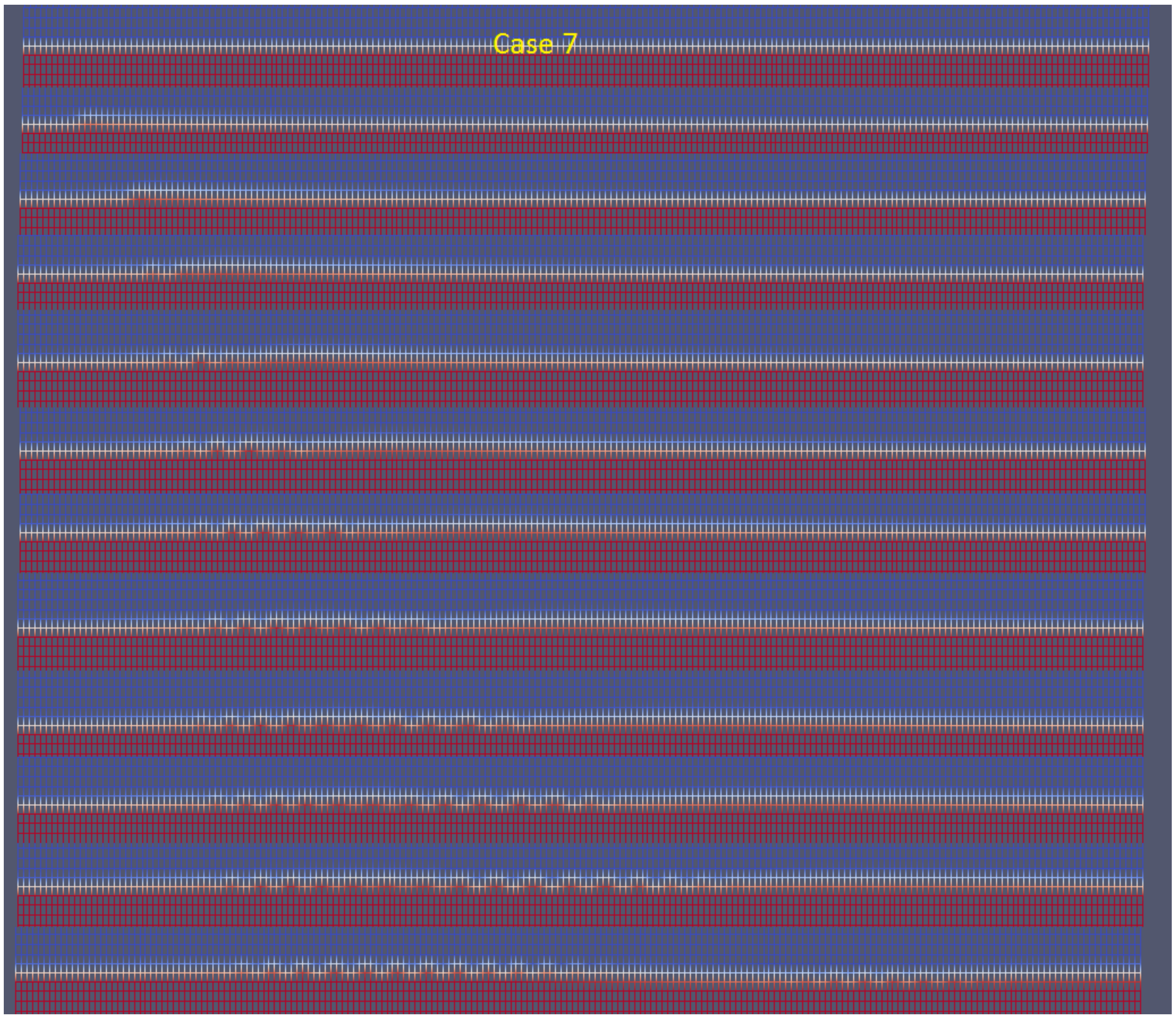

Figure 8.6: Numerical diffusion seen for Case 7

8.1.2 Unstructured grid

The interface is highly sensitive to unstructured grids. In this section we test the effect of a skewed mesh and a mesh with high or low aspect ratio cells. Several cases are tested as shown below.

- **Case 1**: Introducing a skewed mesh from the center of the domain
- **Case 2**: Introducing a skewed mesh from the inlet
- **Case 3**: Introducing a skewed mesh from the inlet, with a coarse mesh compared to Case 2
- **Case 4**: Introducing cells not aligned with flow direction, this is a concern discussed in section 5.1
- **Case 5**: Introducing cells with aspect ratio far from 1

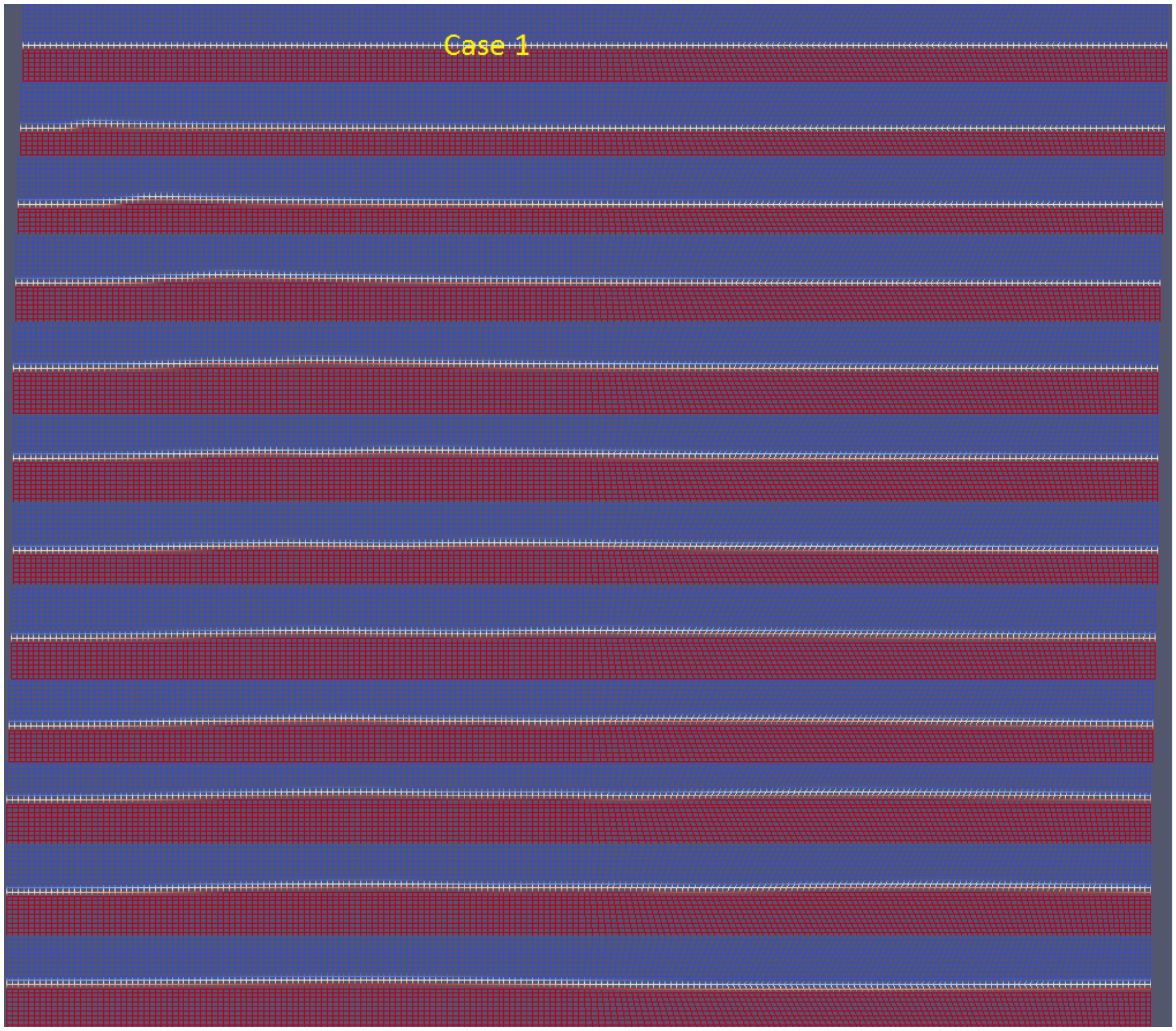

Figure 8.7: Case 1, skewed from the center of domain

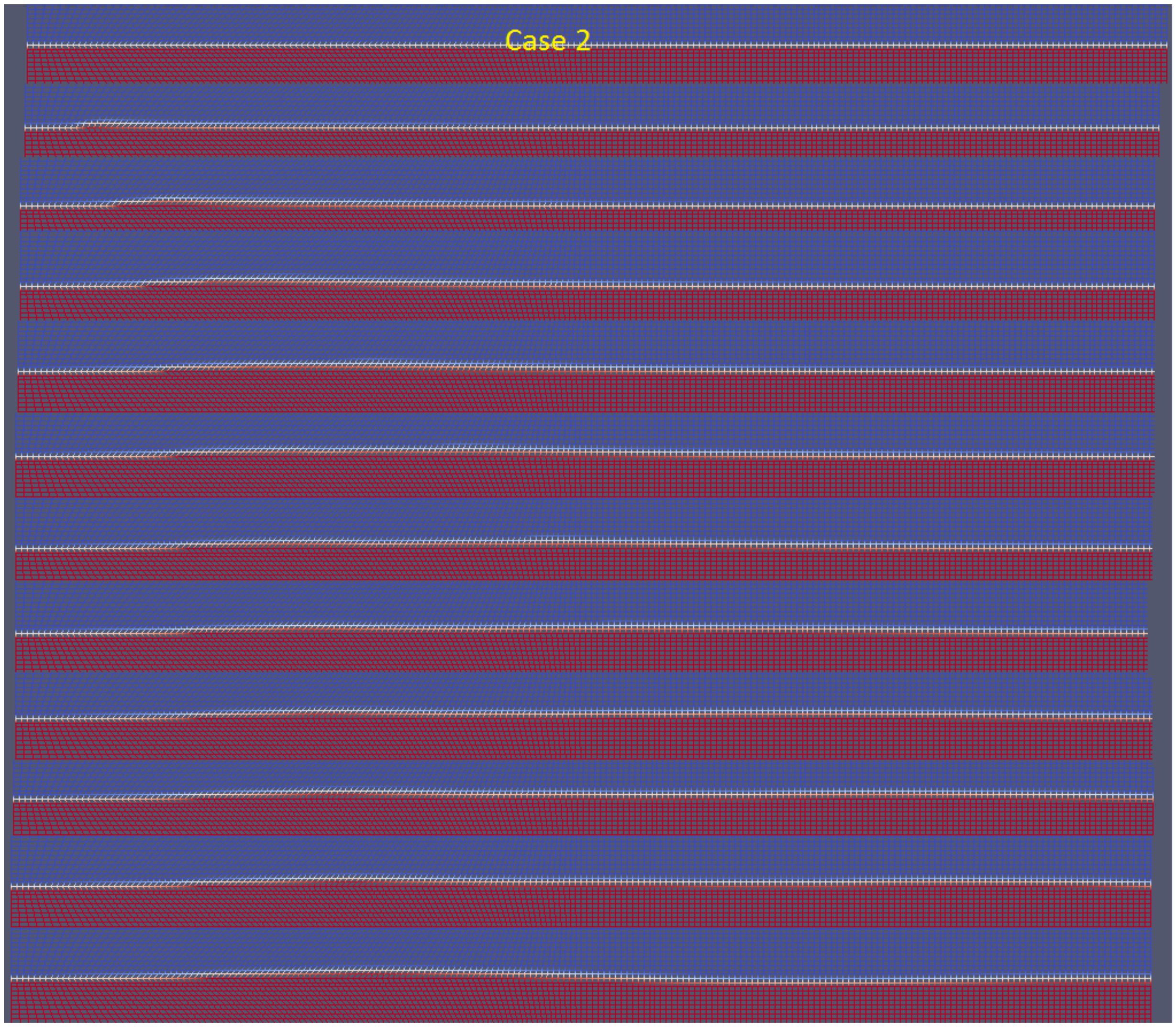

Figure 8.8: Case 2, skewed from inlet

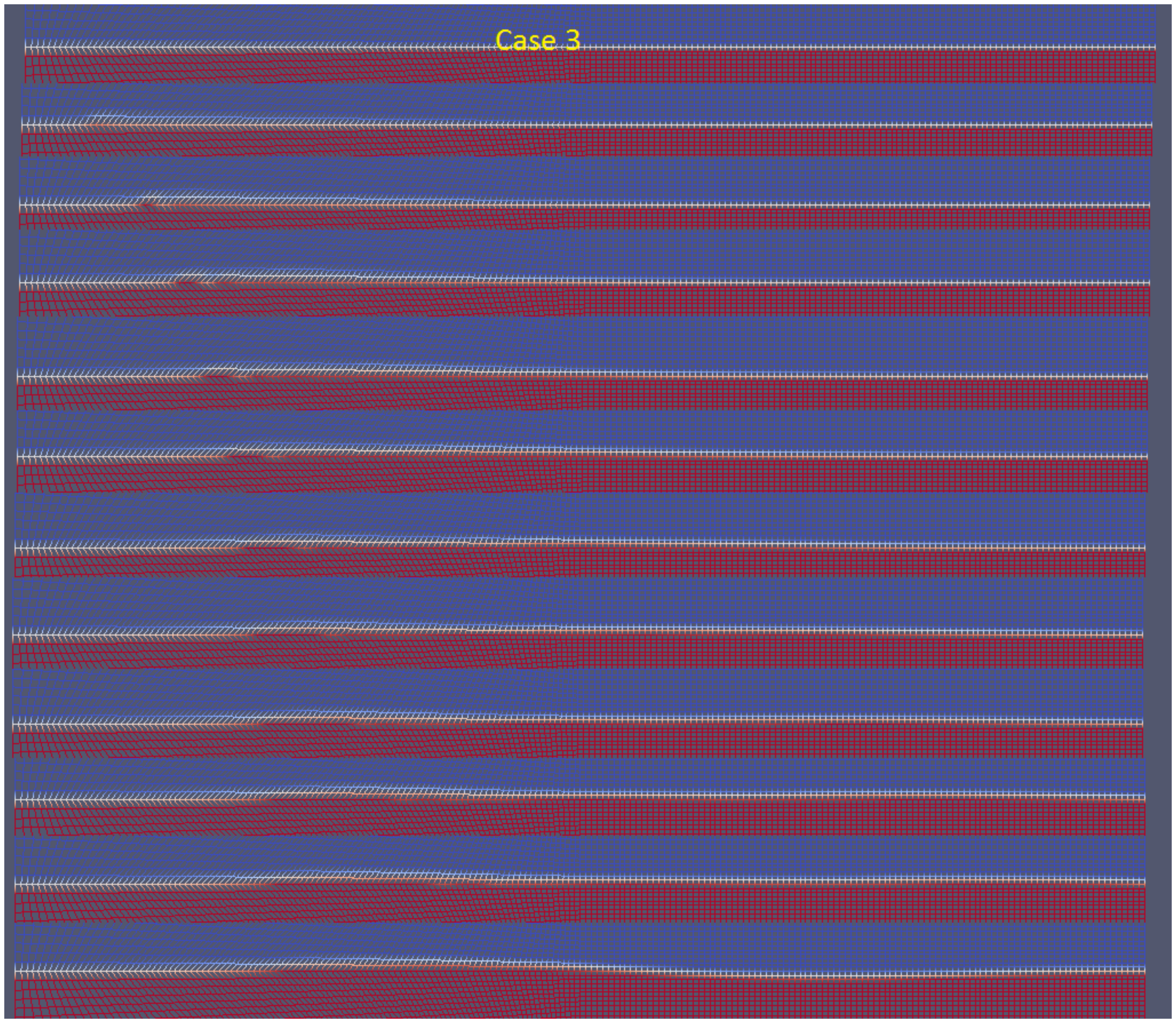

Figure 8.9: Case 3, skewed with coarser mesh than Case 2

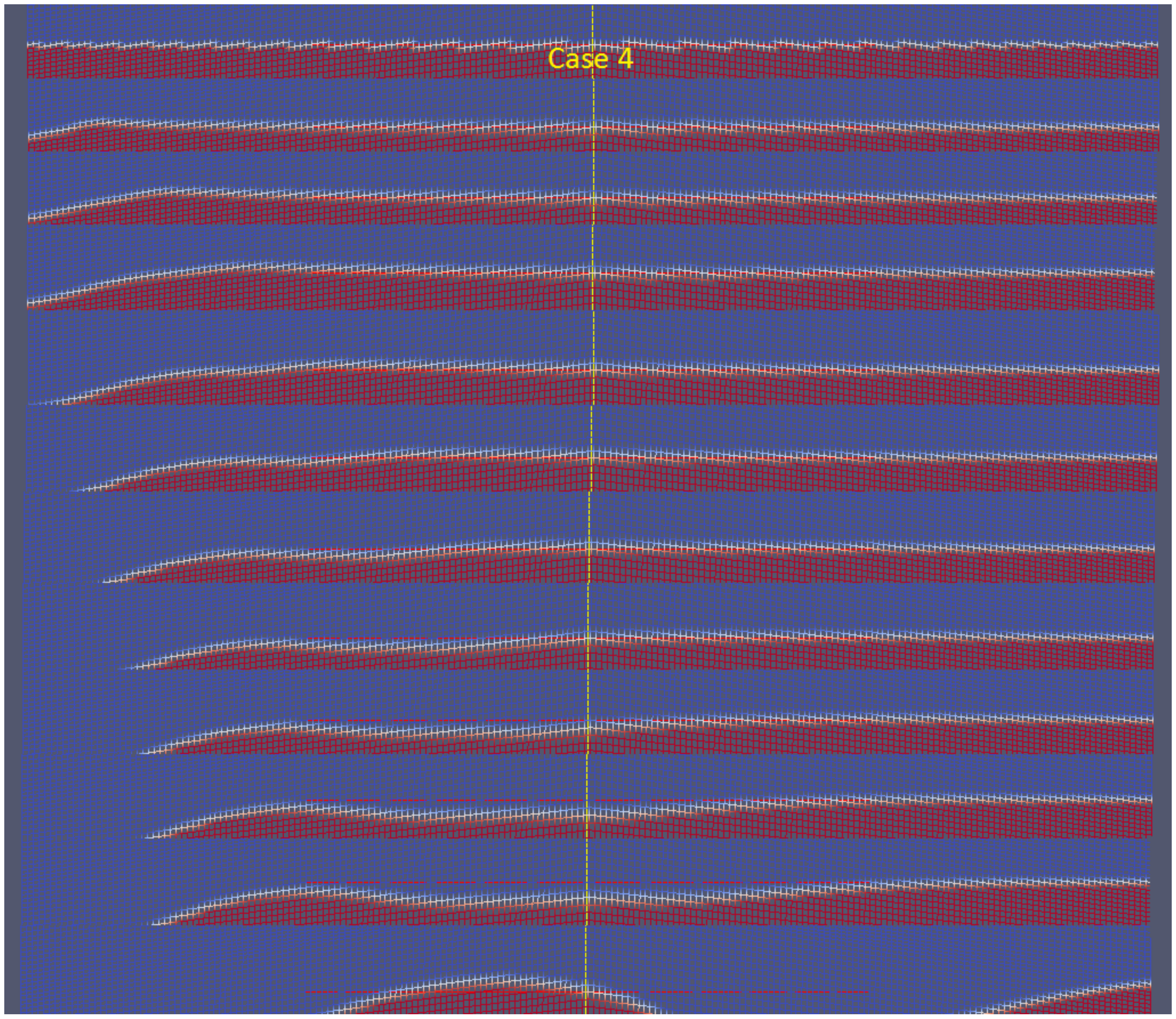

Figure 8.10: Case 4, non-horizontal mesh configuration at inlet

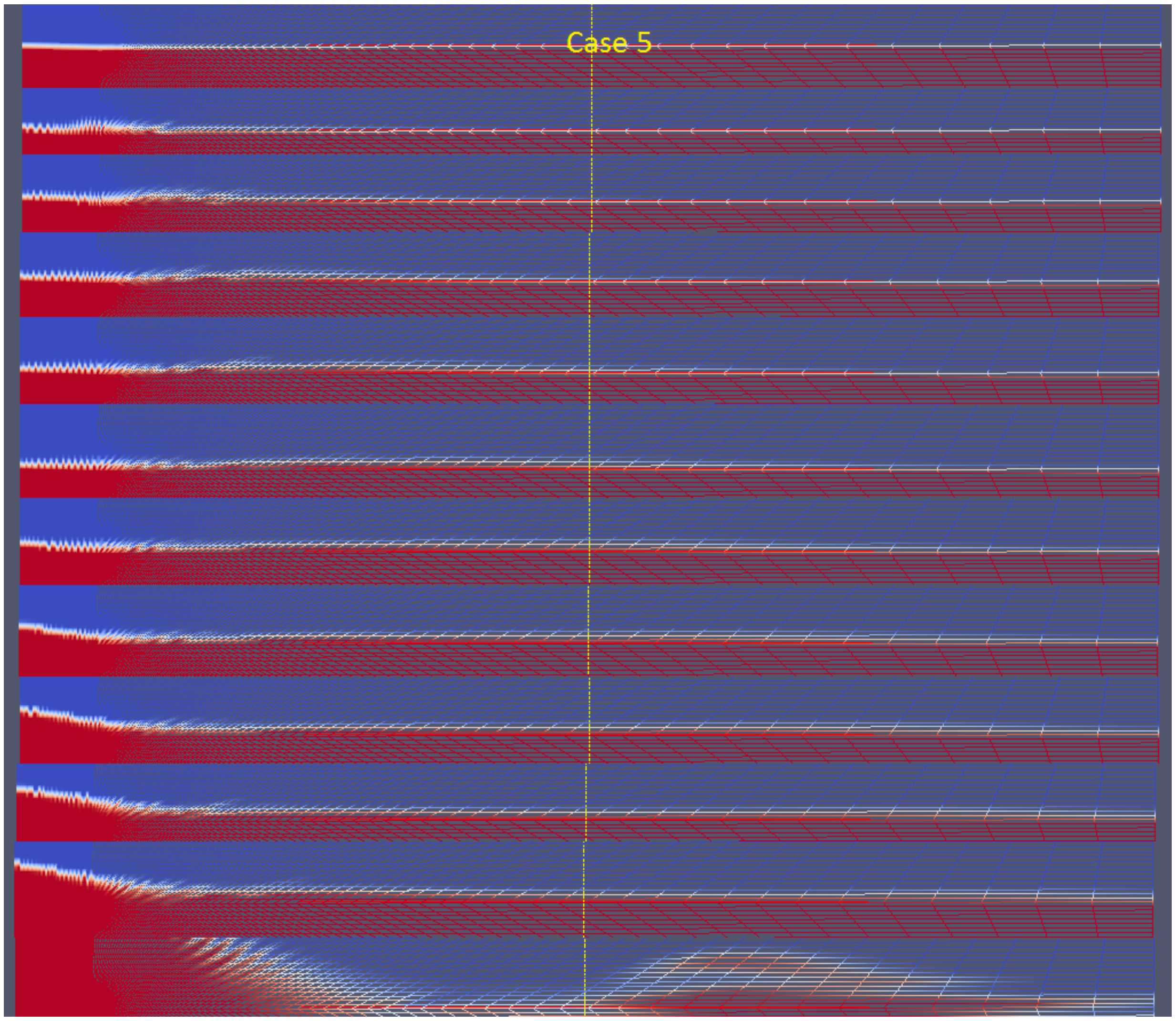

Figure 8.11: Case 5, aspect ratio far from 1

For case 5 the cells near inlet looks like this:

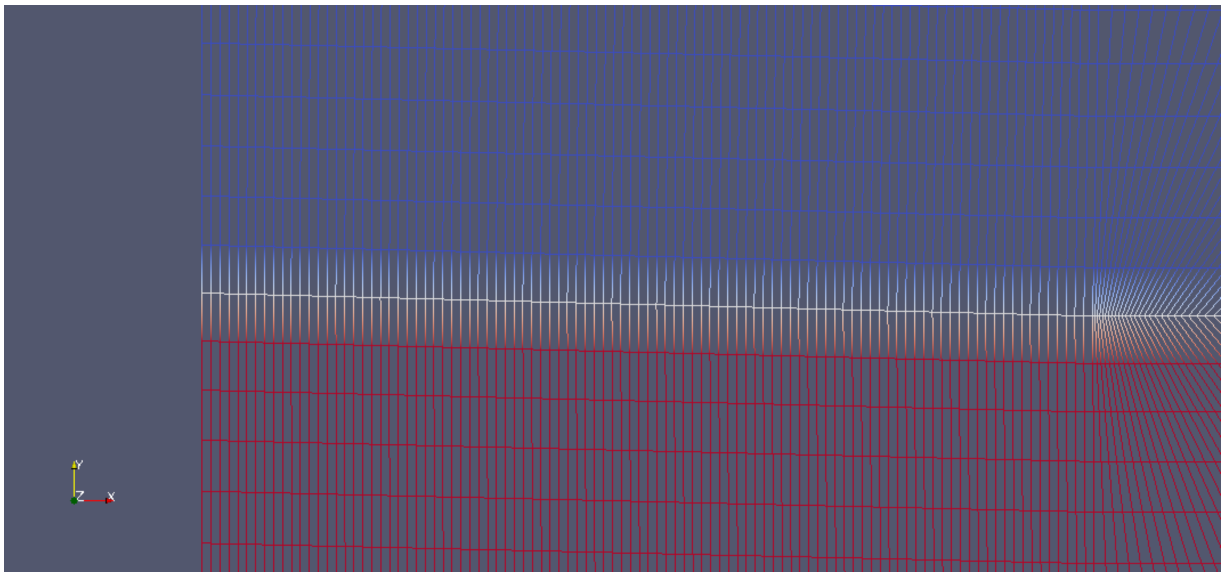

Figure 8.12: Case 5, aspect ratio far from 1. Zoomed in at inlet

The experience is that skewness, and aspect ratio far from 1 near inlet has a big impact on the flow. In contrary, skewness further out in the flow has less effect, this can be seen in Case 1 above.

A disturbance near the inlet will be magnified as the disturbance will be 'pushed' by the inlet flow, this can be clearly seen in case 4 above. Also, if the layer between oil and water is not symmetrical perpendicular to the inflow near the inlet, this will be magnified and carried out in the flow, as in Case 4. This, and in general issues with skewness, is connected to the discussed issues in section 5.1.

The results in this section are in accordance with the issues discussed in section 5.1.

8.1.3 Density

In this section, we choose an appropriate grid, say Case 4 in section 8.1.1, and we alternate the density of oil.

First, validating that no internal wave (i.e. the wave in the layer between the two phases) occurs when simulating flow with two fluids with same properties is important. In this simulation the properties of both fluids are set as water, i.e. $\nu = 0.01$ $\rho_{=}1000$. As seen in figure 8.13, no internal wave occurs in this simulation.

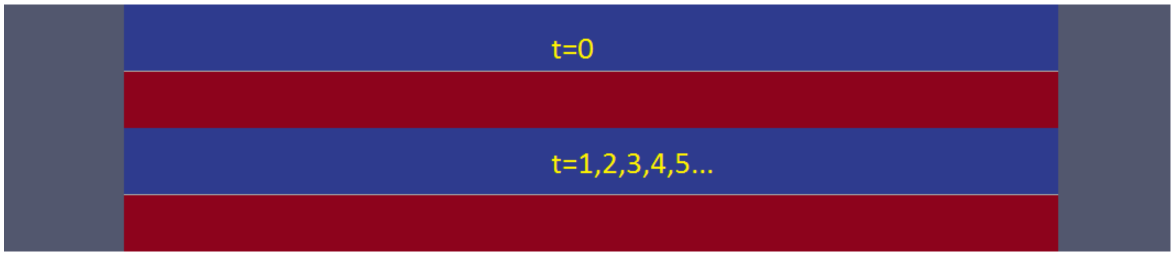

Figure 8.13: Two fluids with same properties does not create an internal wave

To test the behaviour of different fluid properties for the different phases, we test the following cases:

- **Case 1:**
- water: $\nu_{water} = 0.01$ $\rho_{water} = 1000$
- oil: $\nu_{oil} = 0.03$ $\rho_{oil} = 900$
- **Case 2:**
- water: $\nu_{water} = 0.01$ $\rho_{water} = 1000$
- oil: $\nu_{oil} = 0.03$ $\rho_{oil} = 800$
- **Case 3:**
- water: $\nu_{water} = 0.01$ $\rho_{water} = 1000$
- oil: $\nu_{oil} = 0.03$ $\rho_{oil} = 700$
- **Case 4:**
- water: $\nu_{water} = 0.01$ $\rho_{water} = 1000$
- oil: $\nu_{oil} = 0.03$ $\rho_{oil} = 970$
- **Case 5:**
- water: $\nu_{water} = 0.01$ $\rho_{water} = 1000$
- oil: $\nu_{oil} = 0.03$ $\rho_{oil} = 985$

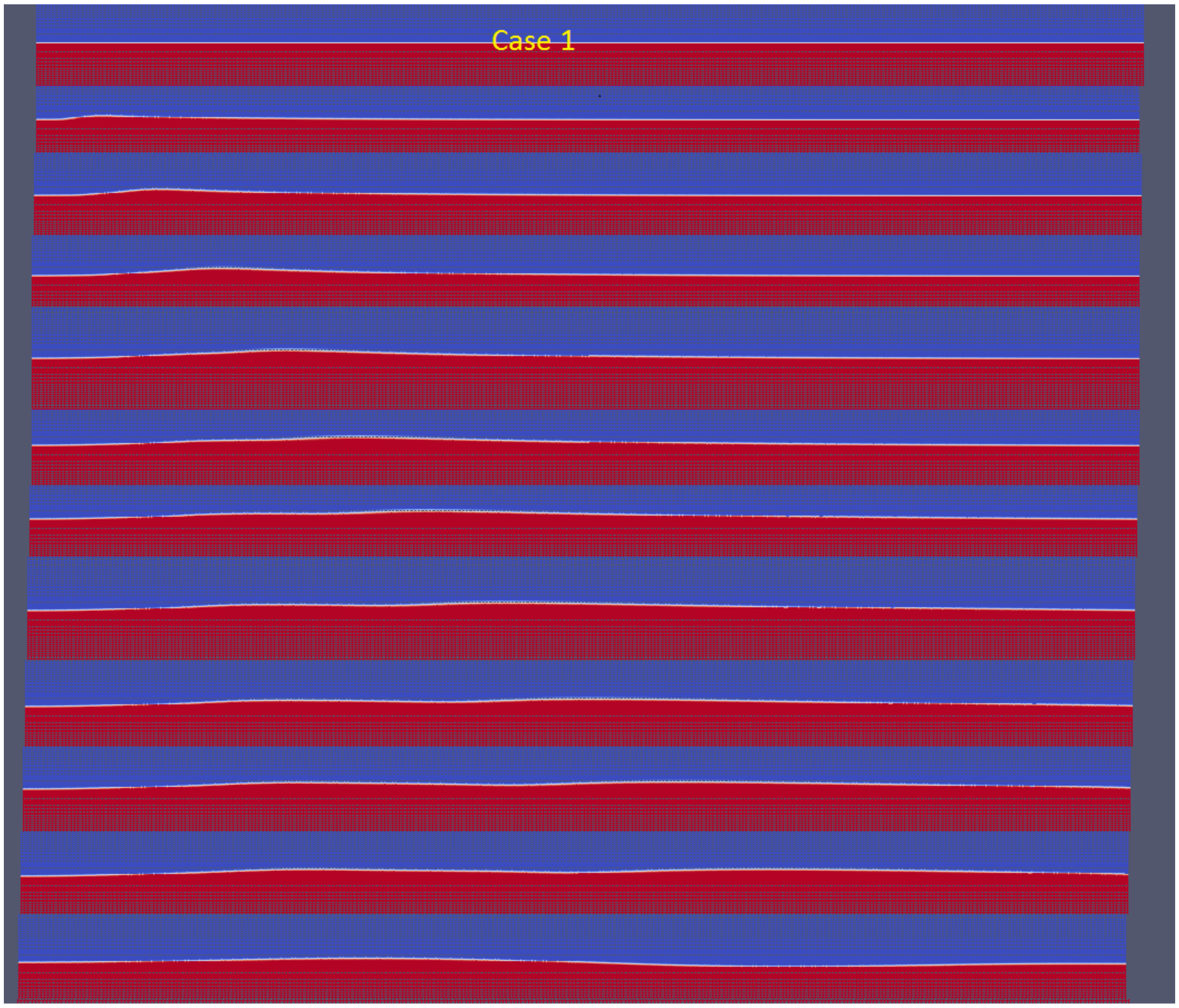

Figure 8.14: Flow pattern for Case 1

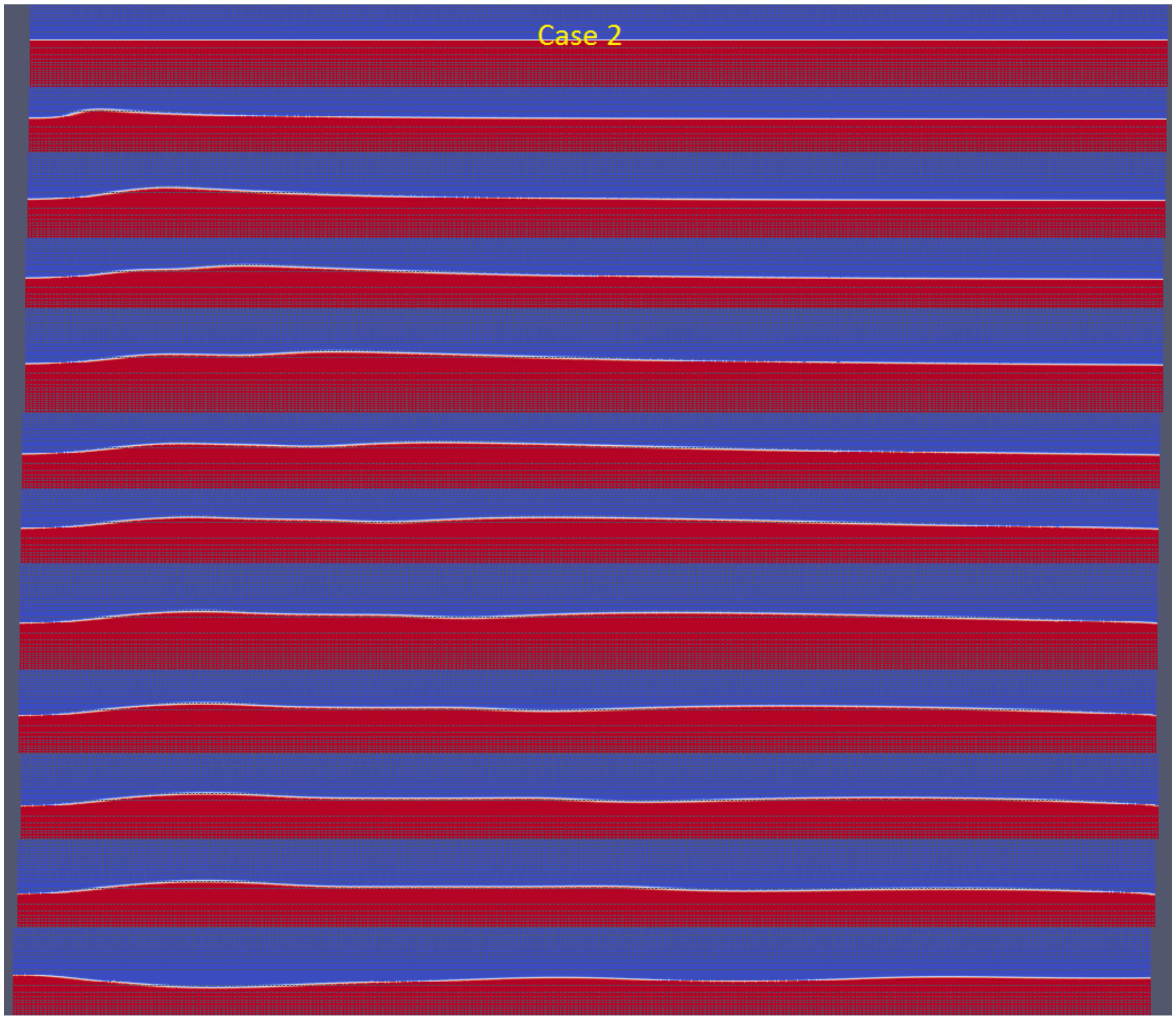

Figure 8.15: Flow pattern for Case 2

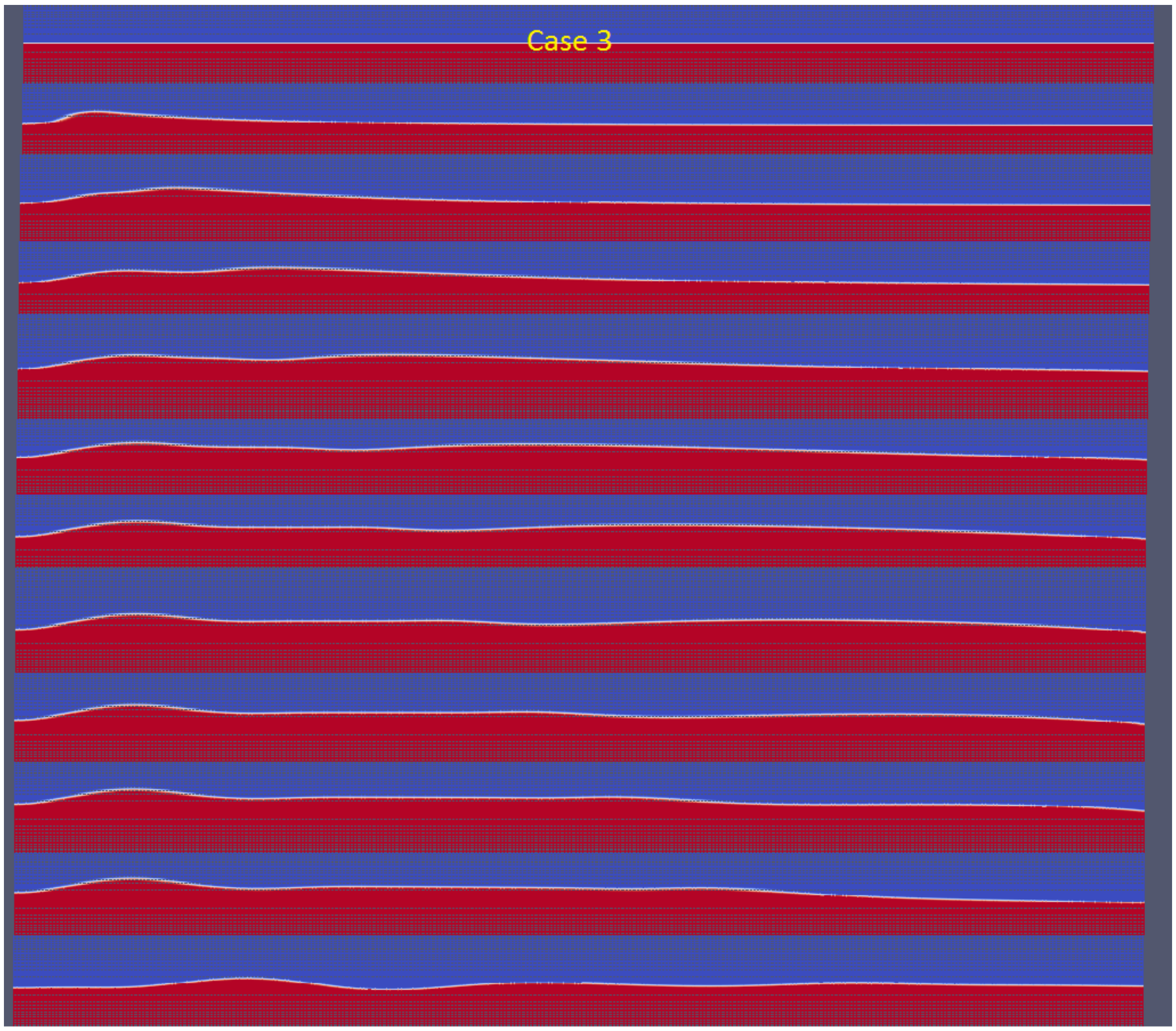

Figure 8.16: Flow pattern for Case 3

From these cases, it can be seen that the internal wave is enhanced when a larger difference in density between the phases is introduced.

Case 4 and Case 5 can be seen in Appendix, (A.7). These show almost no internal wave, as expected.

8.1.4 Kinematic viscosity

Introducing a higher viscosity difference seems to make the simulation more sensitive. Using Case 1 from section 8.1.3 and increasing ν_{oil} introduce some problems. Increasing ν_{oil} seems to make the wave 'sharper', and this eventually makes the flow separate at a certain value of v_{oil}. The cases below shows $v_{oil} = 0.35$ and $v_{oil} = 0.45$. We see that the flow starts separating at $v_{oil} = 0.45$. Increasing v_{oil} makes this effect enhance, which can be seen in section 8.2.4 where $v_{oil} = 1$.

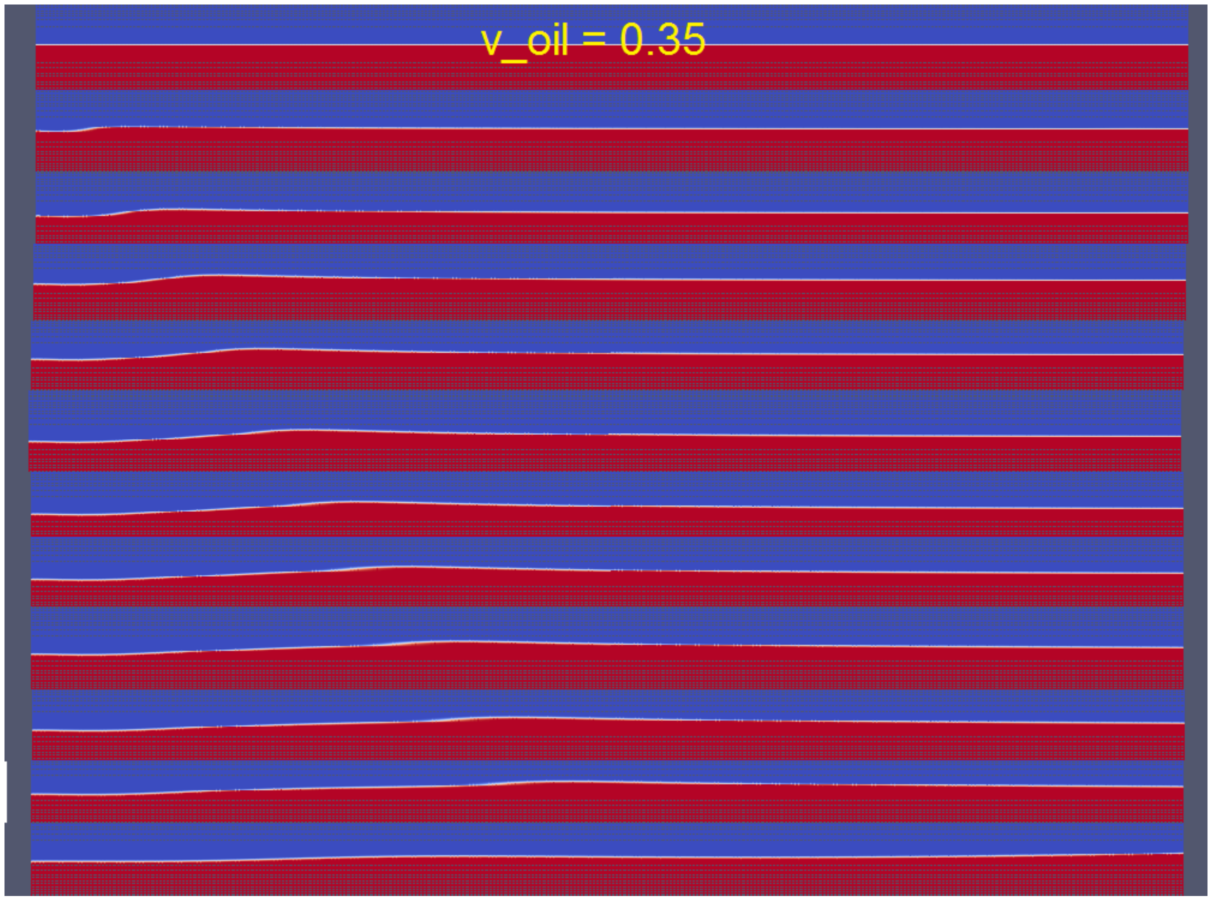

Figure 8.17: Flow pattern for case with $v_{oil} = 0.35$

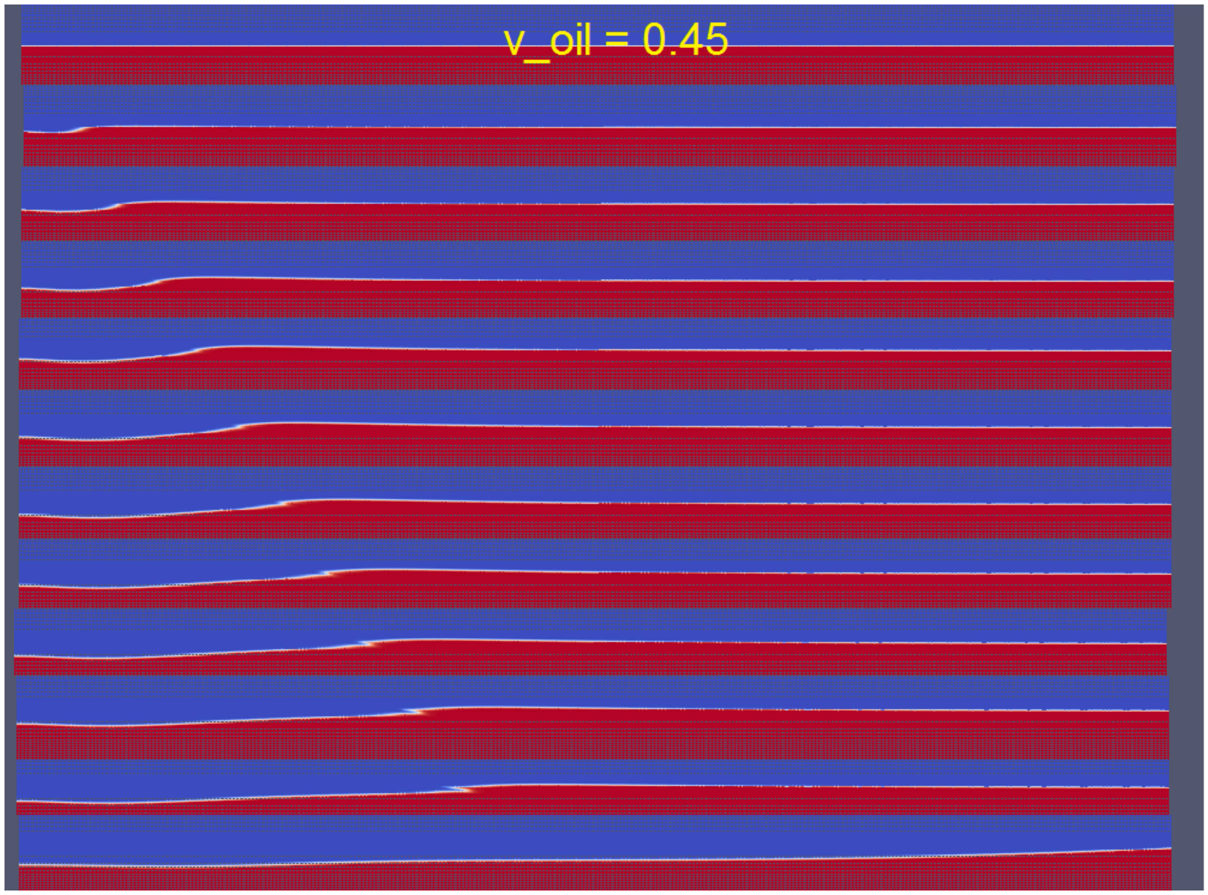

Figure 8.18: Flow pattern for case with $v_{oil} = 0.45$

8.2 Circulation of fluid

8.2.1 Setup

In the following we assume these fluid properties:

Fluid properties: $\rho_{oil} = 900$ $\rho_{water} = 1000$ $\nu_{oil} = 0.03$ $\nu_{water} = 0.01$

In this case a small oil layer, as shown in figure 8.19, is applied. This case is much harder to simulate than the channel case as the fluid will move over a larger area vertically and the flow will also not be aligned with the cells. A typical flow pattern is shown in figure 8.20.

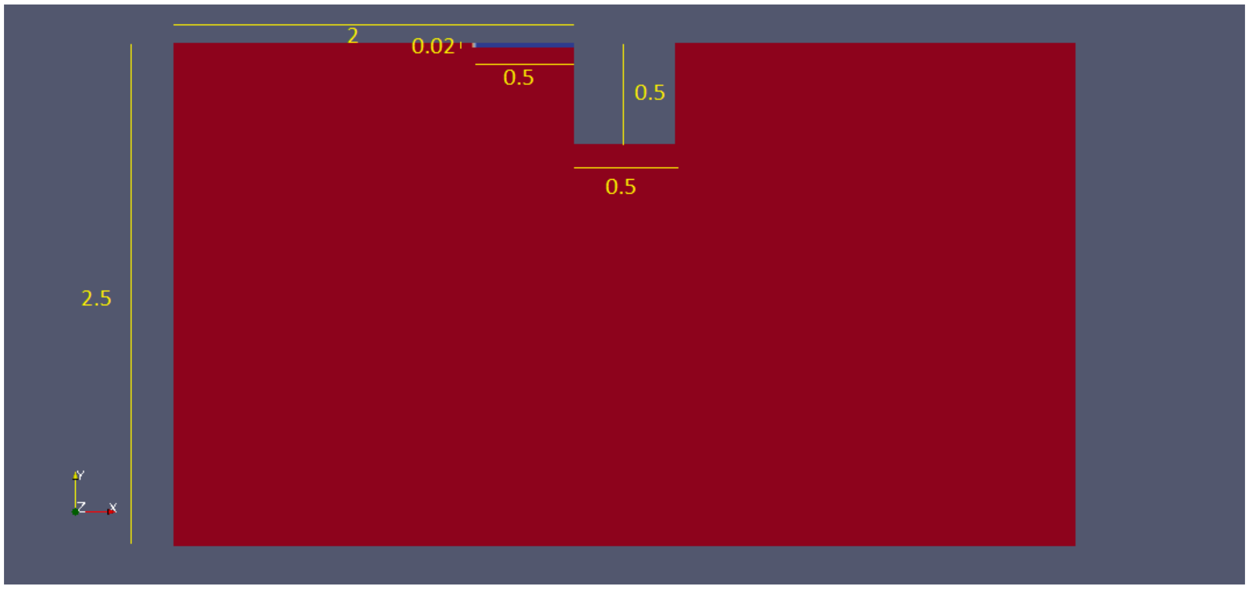

Figure 8.19: Dimensions of applied oil layer

In figure 8.21 and figure 8.22, the flow pattern for a simulation with the same grid, but with different temporal discretisation, i.e. Crank-Nicolson and Euler is shown.

From the figures it seem that the Crank-Nicolson method preserves the circulation slightly better than the Euler method in this case. The Crank-Nicolson method seems to give a wider area of circulation. From here we will therefore use Crank-Nicolson for temporal discretisation.

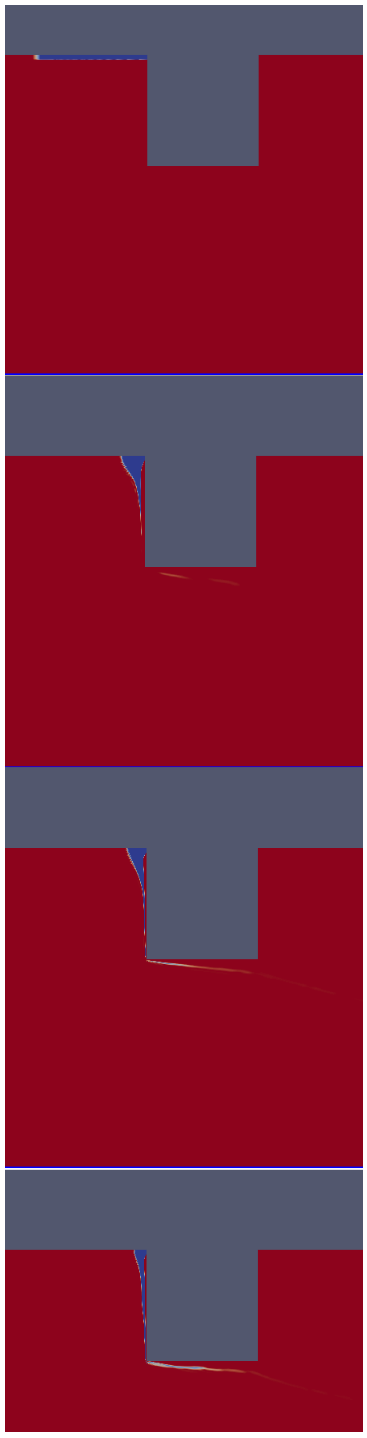

Figure 8.20: Flow pattern with applied oil layer

Re is in this case set to be 50, where Re is based on water.

The boundary conditions are set to be, as before:

- leftWall:
  U: fixedValue uniform (1 0 0)
  p_rgh: zeroGradient

- rightWall:
  U: zeroGradient
  p_rgh: fixedValue uniform 0

- atmosphere:
  U: slip
  p_rgh: zeroGradient

- walls:
  U: fixedValue uniform (0 0 0)
  p_rgh: zeroGradient

- lowerWall:
  U: slip
  p_rgh: zeroGradient

- frontAndBack:
  U: empty
  p_rgh: empty

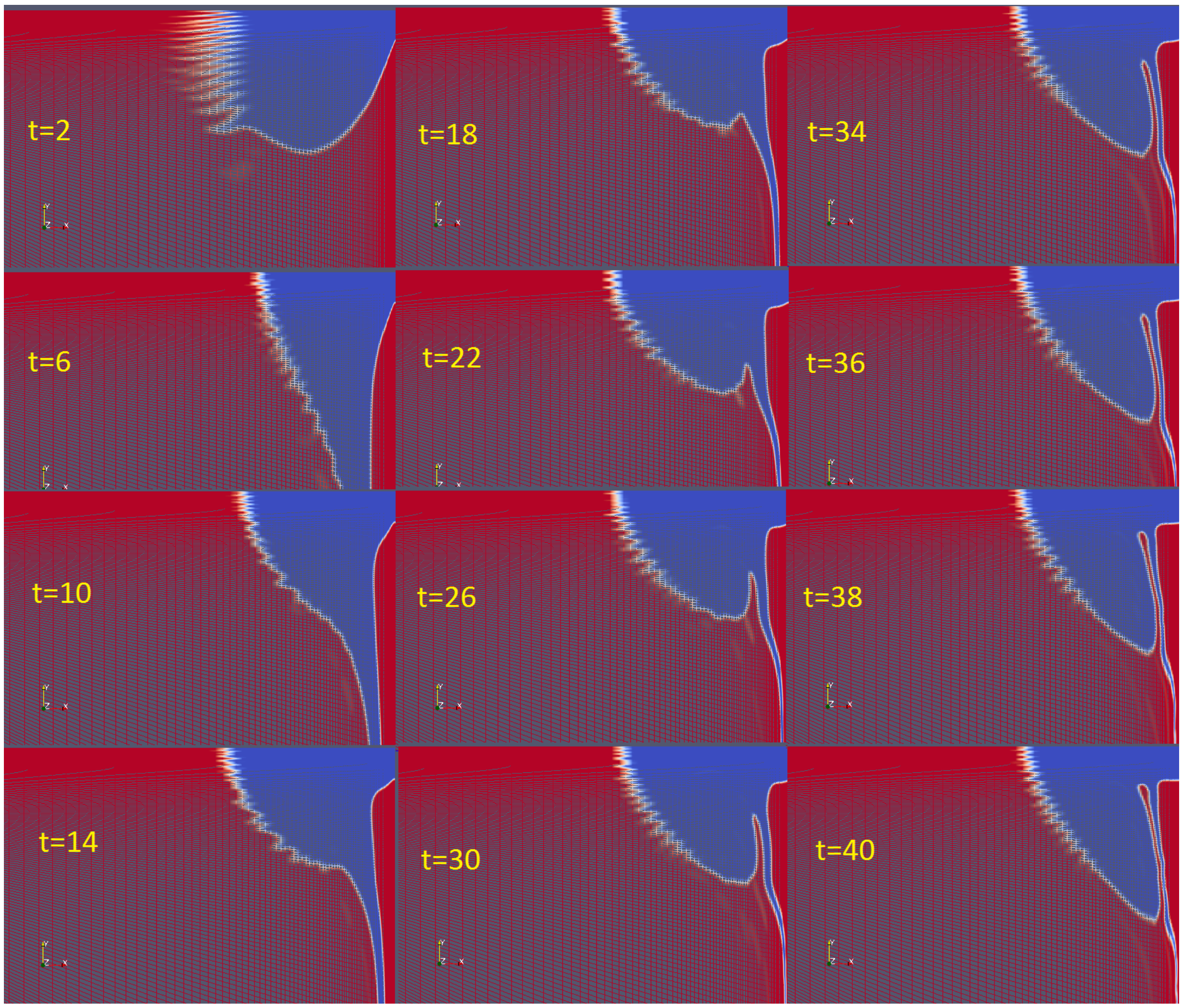

Figure 8.21: Flow pattern with Crank-Nicolson temporal discretisation

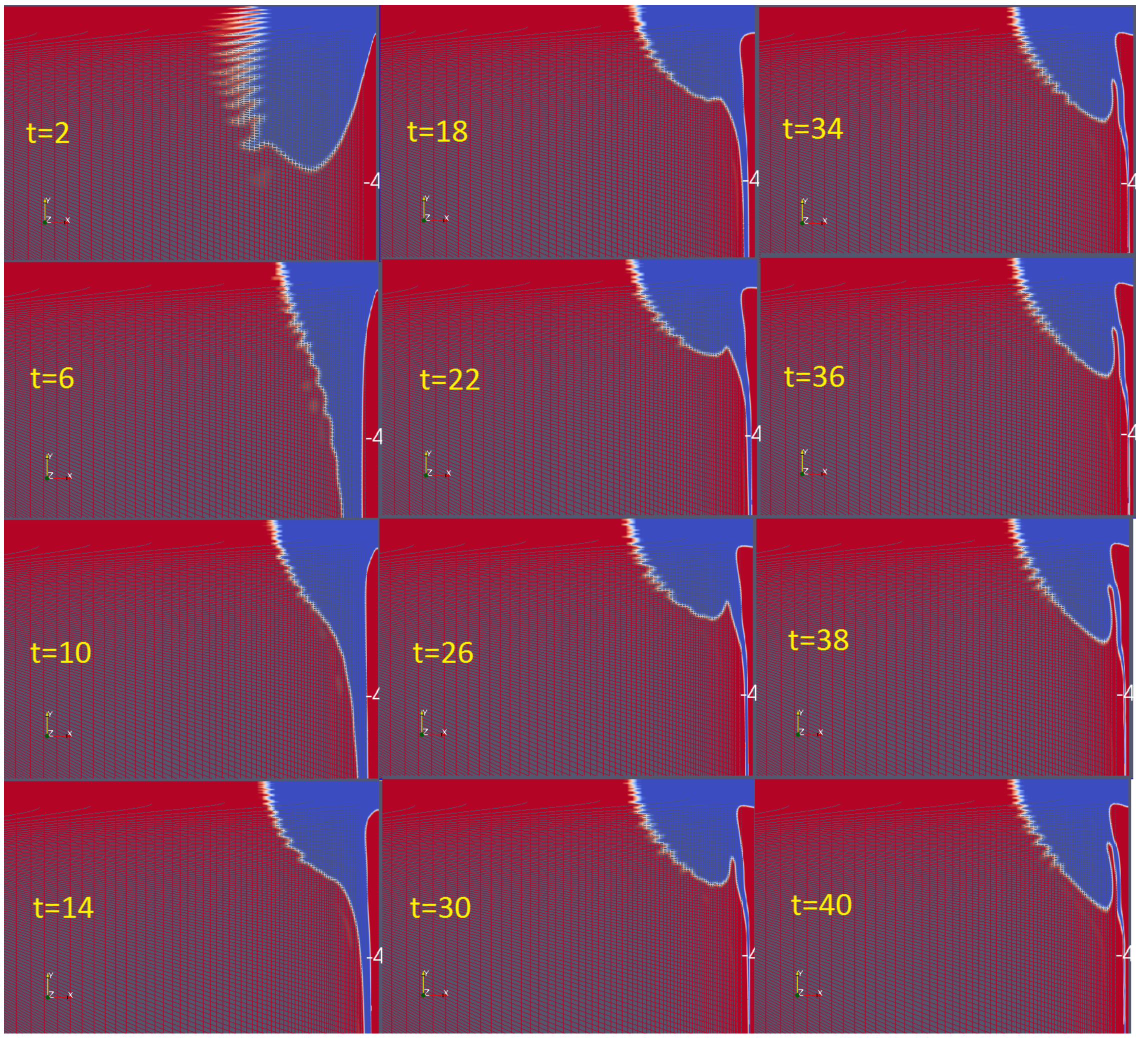

Figure 8.22: Flow pattern with Euler temporal discretisation

For the oil to be contained by the object, it is essential to simulate the circulation ahead of the object in order to prevent the oil from draining directly under the object, as mentioned in section 7.3. The key to conserve this circulation is a refined mesh close to the body. This is highly essential, as it is at the no-slip surface the vortices are created. As seen in figure 8.23 the circulation of fluid can be seen clearly inside the oil layer by the velocity vectors.

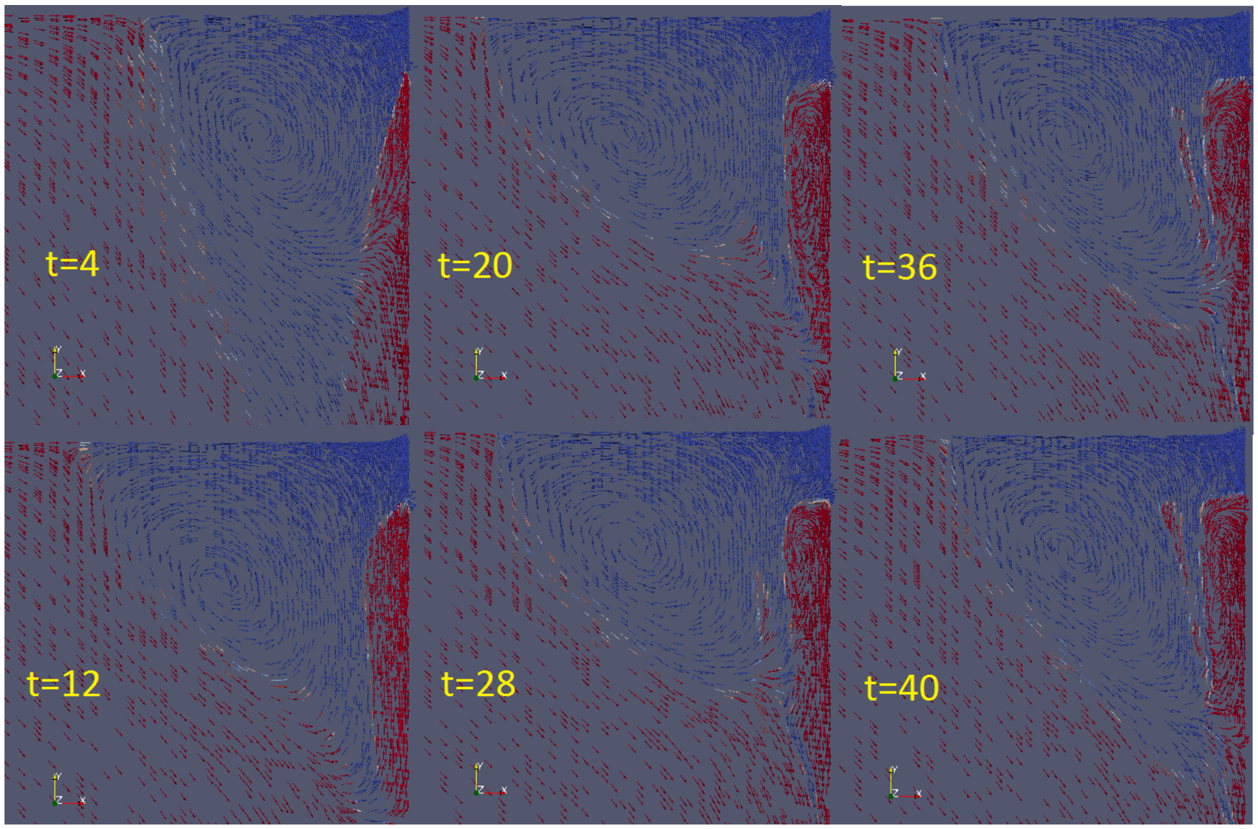

Figure 8.23: Velocity vector field with phase fractions

The trouble with this mesh is that the layer between the oil and water is not fully refined, as seen by the zigzag pattern.

8.2.2 Sensitivity to mesh

The shown zigzag pattern comes mainly from the vertical layer between oil and water, as this is not refined enough in horizontal direction, and also the aspect ratio is not close enough to 1. This topic is discussed in section 8.1. Since the interface is initially not refined well enough, this error will therefore be carried throughout the simulation, no matter how refined the grid is elsewhere. A new pattern, with aspect ratio close to 1 near the vertical interface, is shown in figure 8.24.

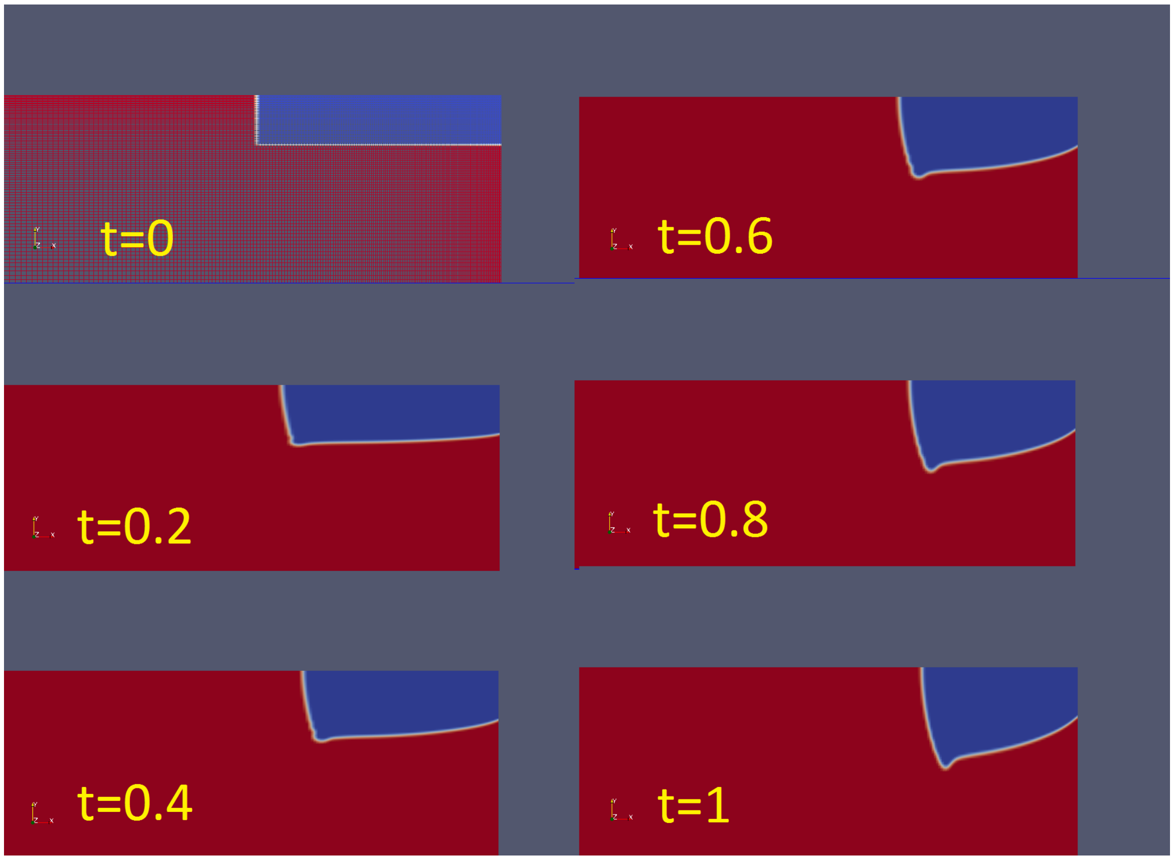

Figure 8.24: Refined horizontal dir., reducing zig zag pattern

As seen from $t = 0$ in figure 8.24 the vertical layer between oil and water are much more refined than the previous simulations. The previous refinement is shown in figure 8.26 and figure 8.25. Even though we refine the grid significantly we still observe this numerical diffusion, which will arise further out in the simulation. This will be further discussed in this section.

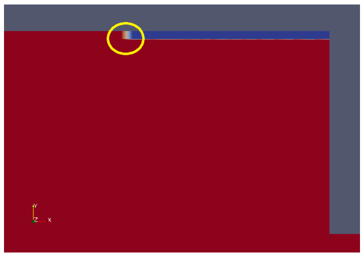

Figure 8.25: Previous set-up, marked area shows vertical interface

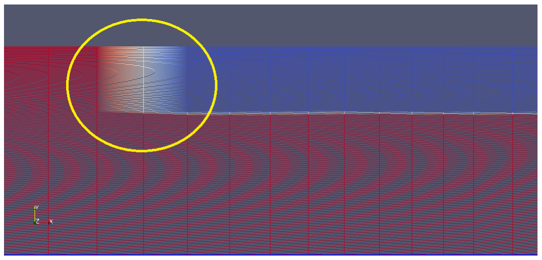

Figure 8.26: grid refinement previous set-up

Another problem arise from compressing the interface in orthogonal direction. A small interruption in the interface seems to distort the interface more significantly when compressed in orthogonal direction. No previous literature are found discussing this issue.

The figure below shows a case where a mesh error, in this case skewness, is carried out in the simulation is shown.

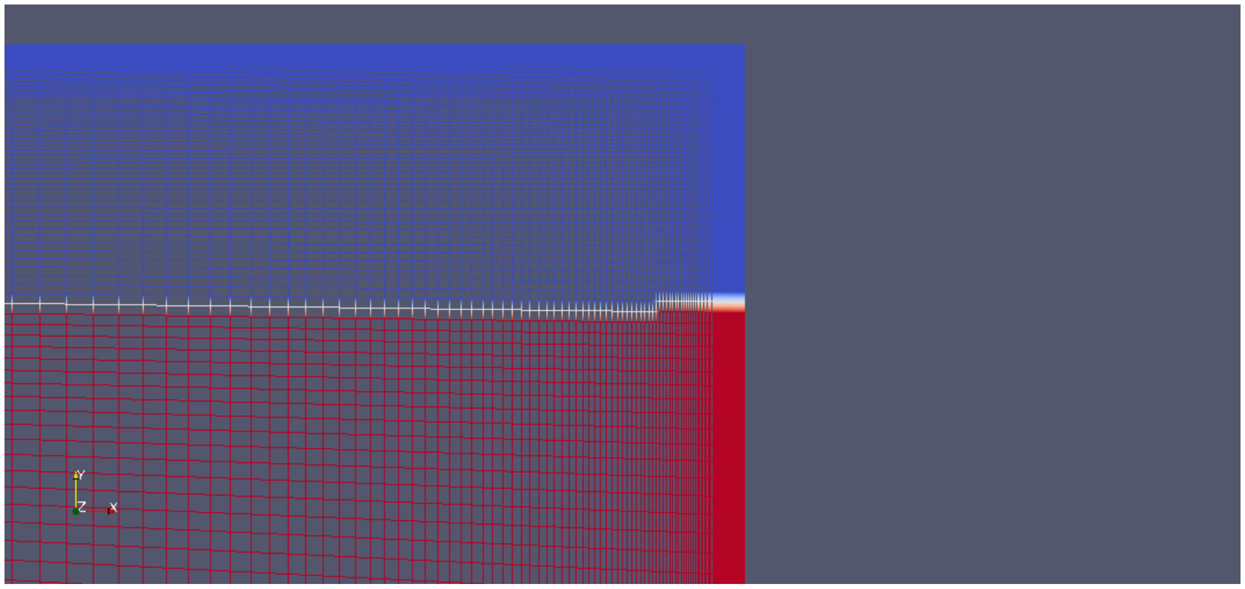

Figure 8.27: Mesh configuration in area of interest

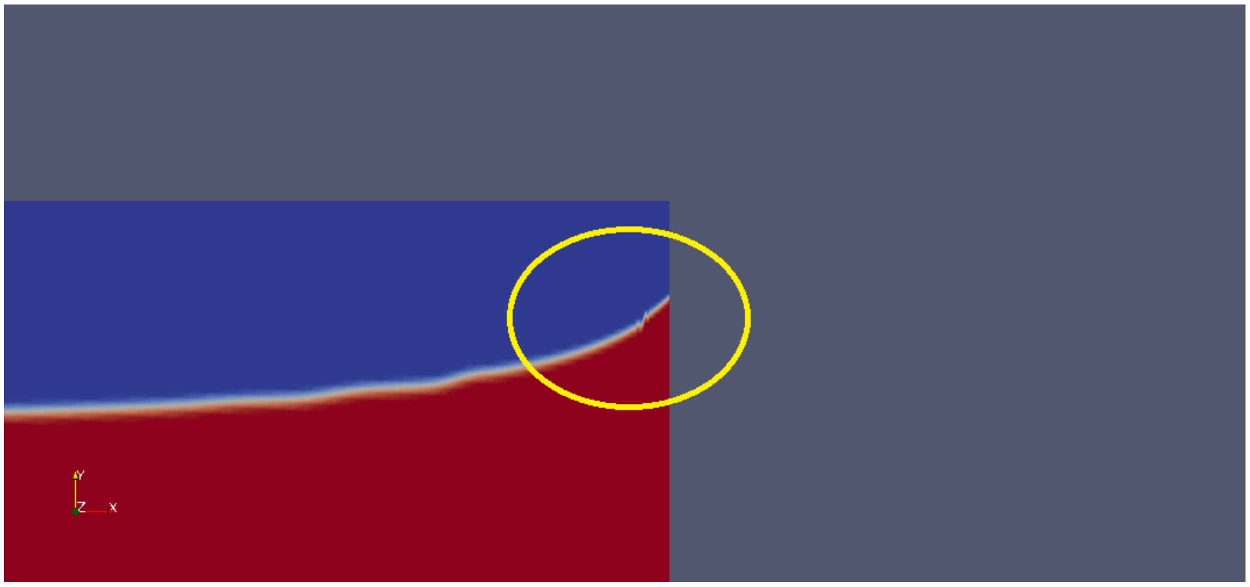

Figure 8.28: The error evolves out in the simulation

As we discussed in section 5.1.2, one of the challenges with the VOF method is that it tries to align the interface with the mesh lines. In a case where the interface between the oil and water is smooth, this error does not occur. We therefore try to make the mesh as aligned to the flow as possible. An example of how an interrupted interface will evolve can also be seen in Appendix (A.8).

This is very troublesome. Ways of improving these errors are analyzed in the sections below.

fvSolution

It is hard to keep the mesh as refined as in figure 8.24 for a longer layer of oil. There are numerous other factors which might be adjusted to obtain a stable solution, e.g. CFL number, discretisation schemes etc. In this section we will adjust solver algorithm criteria and discretisation schemes to see how it affects the flow. In section 8.2.4 we will adjust the CFL number.

InterFoam uses MULES to solve the phase transport equation, as mentioned in section 5.1. This method ensures boundedness of the phase fraction, i.e. it limits the phase fraction to drop below zero or go above 1 in a specific cell (this can happen due to numerical errors). This is done by correcting, by iteration, the volume fraction calculated by the velocity field. This can be seen in the source code of interFoam. Results by Emad (2014) shows that interface profiles predicted by MULES are much sharper and has the lowest diffusive results compared to other algorithms in OpenFOAM.

We can adjust the criteria for MULES in the file 'fvSolution', where we adjust the following under "alpha.water.*":

- **nAlphaCorr**: number of α-corrections, via fixed point iteration
- **nAlphaSubCycles**: sub-cycles within the α-equation, within a time-step
- **cAlpha**: compression factor for the interface (0=no compression, 1=conservative compression, larger than 1=enhanced compression)
- **nLimiterIter**: limit for number of iterations
- **tolerance**: iteration tolerance, i.e. iteration criteria

From the simulations it seems that when the interface is being compressed, i.e. pressed towards the object the numerical diffusion enhances. In the following we look at how we can improve this diffusion by alternating the MULES criteria.

In these tests we use 300x300 cells in the marked box in figure 7.16. This is a significant refinement from the results in section 7.3, but is done to keep the vertical interface refined. The mesh is more refined near the vertical interface than near the object. So moving towards the object while being compressed eventually causes diffusion.

To find the best suited criteria we run the cases shown in table below, keeping the mesh and other conditions the same, using the same set-up as in figure 8.19.

In this study we track the time until numerical diffusion appears. We note the time for each case to see which case performs best. Figure 8.29 shows the first time-step where this numerical diffusion is observed for Case 1. For the other cases see Appendix (A.6). A zoomed in picture of the diffusion area and how this diffusion is evolving can be seen in Appendix (A.5).

- **Case 1:**
- nAlphaCorr: 2
- nAlphaSubCycles: 1
- cAlpha: 1
- nLimiterIter: 3
- tolerance: 1e-8
- **Case 2:**
- nAlphaCorr: 3
- nAlphaSubCycles: 1
- cAlpha: 1
- nLimiterIter: 10
- tolerance: 1e-10
- **Case 3:**
- nAlphaCorr: 10
- nAlphaSubCycles: 1
- cAlpha: 1
- nLimiterIter: 20
- tolerance: 1e-20
- **Case 4:**
- nAlphaCorr: 20
- nAlphaSubCycles: 1
- cAlpha: 1
- nLimiterIter: 30
- tolerance: 1e-30
- **Case 5:**
- nAlphaCorr: 10
- nAlphaSubCycles: 10
- cAlpha: 1
- nLimiterIter: 20
- tolerance:1e-20
- **Case 6:**
- nAlphaCorr: 10
- nAlphaSubCycles: 1
- cAlpha: 1.5
- nLimiterIter: 20
- tolerance: 1e-20
- **Case 7:**
- nAlphaCorr: 2
- nAlphaSubCycles: 2
- cAlpha: 1
- nLimiterIter: 3
- tolerance:1e-8
- **Case 8:**
- nAlphaCorr: 2
- nAlphaSubCycles: 4
- cAlpha: 1
- nLimiterIter: 3
- tolerance:1e-8

The experience from running these cases is that alternating nAlphaCorr, cAlpha, nLimiterIter and tolerance seem to have small effect on the computation time. In contrary, increasing nAlphaSubCycles increases computation time significantly. Running Case 5 takes approximately 10 times longer time than Case 3. The experience in this thesis is that increasing nAlphaSubCycles makes the computation time approximately nAlphaSubCycles times longer.

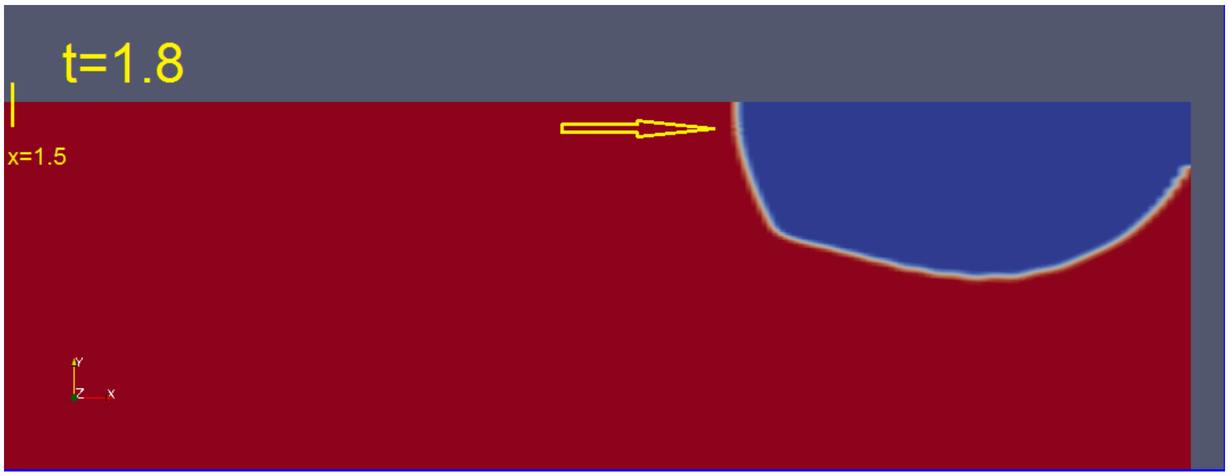

Figure 8.29: First time-step where numerical diffusion appears for Case 1

We find the following results for this study of MULES:

Time for numerical diffusion to appear

- **Case 1:** 1.8 s
- **Case 2:** 2.4 s
- **Case 3:** 2.8 s
- **Case 4:** 2.8 s
- **Case 5:** 3.2 s
- **Case 6:** 2.2 s
- **Case 7:** 2.0 s
- **Case 8:** 2.2 s

From Case 3 and Case 4 we see that increasing nAlphaCorr, nLimiterIter and tolerance from Case 3 does not improve results. Also enhancing the compression, i.e. increasing cAlpha, from the conservative level at cAlpha=1, rather makes results worse than improving as seen from Case 6.

Increasing nAlphaSubCycles seems to improve the results significantly, especially from Case 3 to Case 5. The downside with this is, as previously mentioned, the computation time. From Case 7 and Case 8 it seems that nAlphaSubCycles should be increased to more than 4 for improving the results significantly. Increasing nAlphaSubCycles from 1 to 2 have almost no effect as seen by comparing Case 1 with Case 7.

One last check we do is to check the PIMPLE algorithm, i.e. a hybrid algorithm of PISO and

SIMPLE. However, with certain criteria (keeping nOuterCorrectors below 2) the PIMPLE will run as pure PISO. See section 5.1 for further information on PISO. We alternate the PIMPLE criteria for Case 1 to see if the results are improved. The PIMPLE criteria for Case 1 looks like:

```
PIMPLE
{
 momentumPredictor no;
 nOuterCorrectors 1;
 nCorrectors 3;
 nNonOrthogonalCorrectors 0;
 pRefCell 0;
 pRefValue 0;
}
```

To test this algorithm we set nOuterCorrectors=2 and nCorrectors=6. Setting nCorrectors=6 means that we are calculating the pressure 6 times.

Setting nOuterCorrectors=2 means we are not longer solving using PISO, but instead PIMPLE. The result is that the error evolves from the beginning (t=0.6), see Appendix (A.6). This shows that using PIMPLE is not a good idea for this problem.

Also, testing PISO with a higher nCorrectors is done. In this test nOuterCorrectors was set to 1, while nCorrectors was set to 10. This results in a significantly worse result. The numerical error appears as fast as at t=0.4, see Appendix (A.6).

Lowering nCorrectors to 1, while using PISO, makes the flow stay stable for a longer time (t=5.4). However, the flow of the oil layer seems to be moving slower and takes significantly longer time for the layer to be compressed as much as in the other cases. These results should be disregarded as it is stated in Greenshields (2015a) that PISO requires more than one correction. Also stated in Greenshields (2015a) is that PISO normally does not require more than 4 corrections. A comparison of PISO with nCorrectors=1 and nCorrectors=3 can be seen in Appendix (A.6). Considering the slip boundary condition at the top, the results for nCorrectors=1 seem non-physical as it smears out the edge on the interface too fast.

fvSchemes

The different schemes used also influence the solution. In this section we use Case 1 from the fvSolution section. Case 1 has the following set-up for divergence operator, i.e. divSchemes, in fvSchemes:

```
divSchemes
{
 div(rhoPhi,U)  Gauss linearUpwind grad(U);
 div(phi,alpha)  Gauss vanLeer;
 div(phirb,alpha)  Gauss linear;
 div((muEff*dev(T(grad(U))))) Gauss linear;
}
```

Several papers, e.g. Herreras and Labeaga (2013), suggest using Gauss interfaceCompression for div(phirb,alpha) under 'divSchemes'. The results in this thesis shows that, in combination with the previously used schemes, Gauss interfaceCompression is not desirable compared to Gauss Linear. The first interruption with Gauss interfaceCompression occurs at t=1.6 (see Apendix (A.6)), which is t=0.2 before Gauss linear.

Changing div(phi,alpha) from Gauss vanLeer to Gauss limitedLinear 1 also worsen the result, more significantly. As seen in Appendix (A.6) we can see the numerical errors evolve two places on the interface with this discretisation.

We also test the proposed schemes in Schulze and Thorenz (2014). This is setting div(phi,alpha) to Gauss Minmod and div(rhoPhi,U) to higher order (in this case we use SFCDV). The results show no improvement, and the first interruption occurs at t=1.8, see Appendix (A.6).

Structured grid

The results in this section improves the simulations significantly, by adjusting MULES, but the interface is still very sensitive to mesh configuration. Even though we have refined the mesh from 180x300 (section 7.3) to 300x300, we still observe problems with a skewed mesh. Figure

8.30 shows a development for t=0, t= and t=2 for the same mesh configuration as in the cases above. But in this case the mesh is a little bit less incremented (half the value from the cases tested in fvSolution) horizontally, which in order introduce a bit more skewed mesh. In Appendix (A.8), t=0 and t=2 for even less incremented ($\frac{1}{4}$ of fvSolution results) can bee seen.

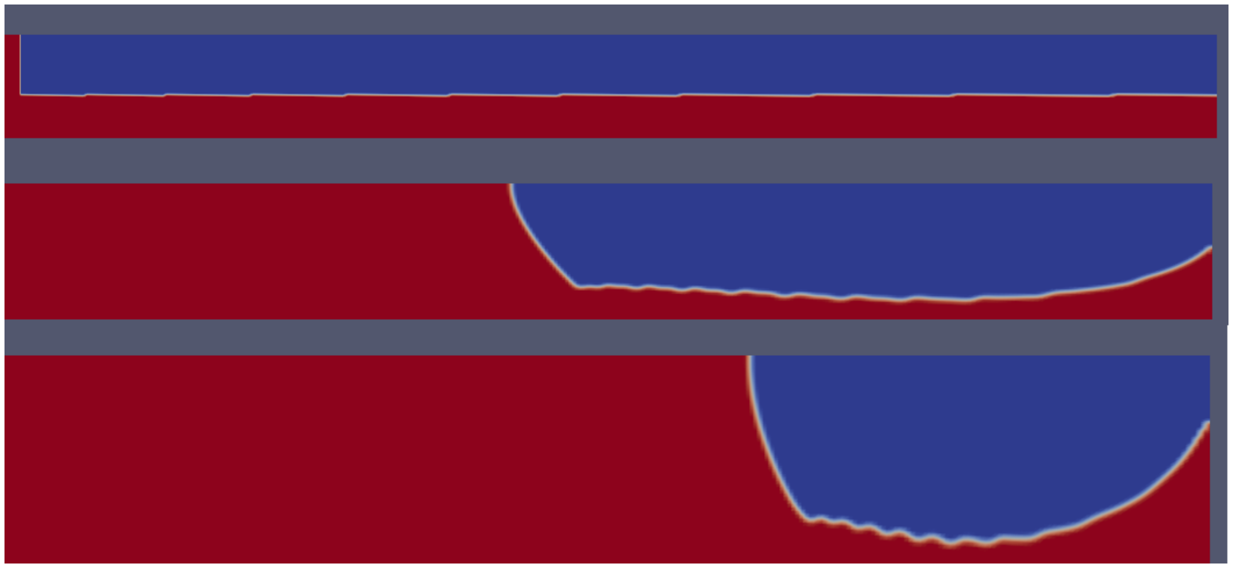

Figure 8.30: Skewness imposed by increments evolving during the simulation

In order to ease this issue of distorted interface, the experience in this thesis is that it is necessary (for this problem) to introduce a structured grid in the box marked in figure 7.16.

Using a 260x300 structured mesh and setting the oil layer $\frac{1}{5}$ times the length of the above results seem to give results without a diffusive interface. The reason for shortening the oil layer is the fact that with a longer layer we will need an extensive amount of cells to keep both the vertical interface and the boundary layer (in accordance with with section 7.3) refined enough.

Using the results from the section fvSolution (Case 3, as Case 5 is too time-consuming) and section fvSchemes (the initial set-up) we obtain the flow pattern shown in figure 8.31. We also use maxAlphaCo=0.15 as will be discussed in section 8.2.4.

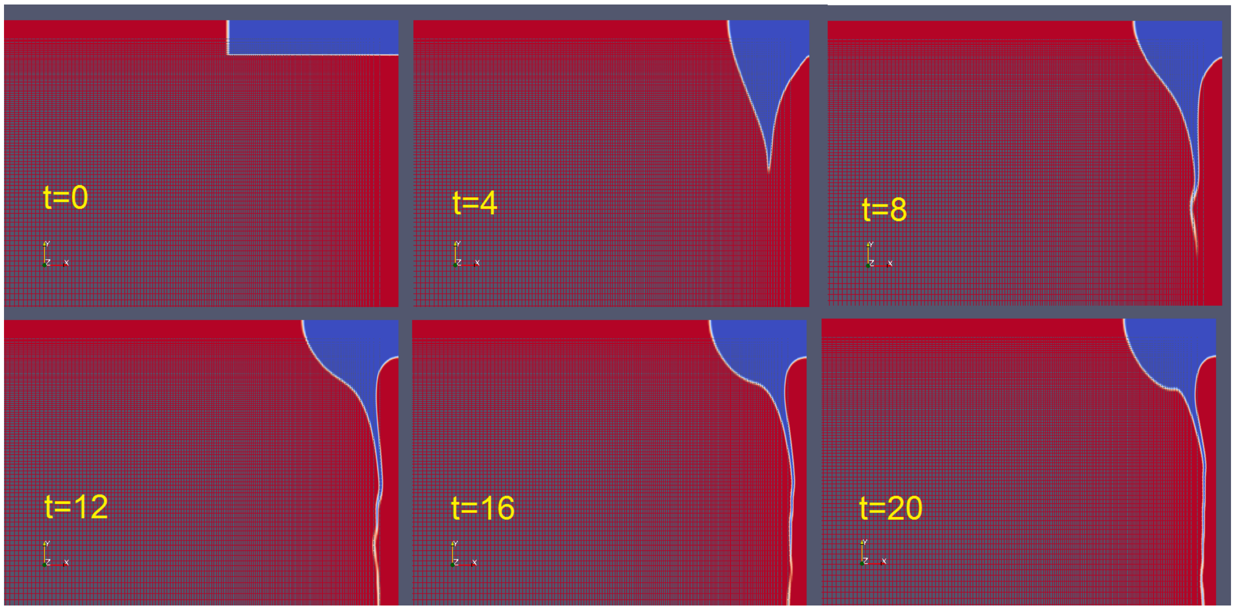

Figure 8.31: Flow pattern with a structured grid

The result for t=20 zoomed in on the interface can be found in Appendix (A.9).

It is interesting to compare the circulation with the results in section 7.3.2 to see how the oil layer affects the area of circulation. Figure 8.32 shows the streamlines for the case with applied oil layer, and figure 8.33 shows the velocity vectors. Compared to figure 7.36 and figure 7.37 we can see that the area of circulation is greatly enhanced by imposing an oil layer.

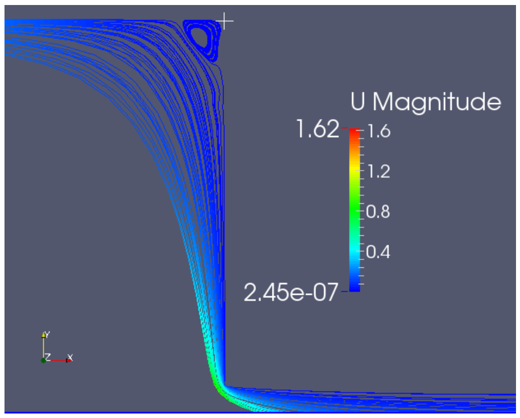

Figure 8.32: Streamlines showing area of circulation, Re=50, angle=0 degree with applied oil layer

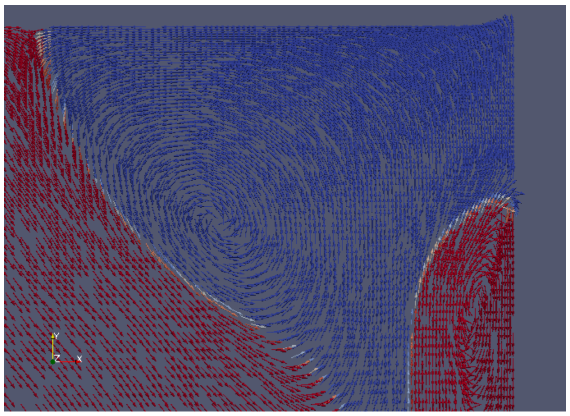

Figure 8.33: Velocity vectors showing area of circulation, Re=50, angle=0 with applied oil layer

8.2.3 Simulations at different Re

Simulations are run for Re=25, Re=50, Re=75 and Re=100.

The flow pattern for Re=25 and for Re=75 can be seen in figure 8.37 and figure 8.38 respectively. The flow pattern for Re=100 can be seen in Appendix (A.10).

Comparing these flow patterns to the flow pattern for Re=50 (figure 8.31) we can see that the circulation is larger and last longer for smaller Re. As seen by the case with Re=75, the circulation is only preserved up to a certain time (for Re=75 up to approximately t=17). Figure 8.36 shows the velocity vectors at t=40. As seen, there are no circulation in the oil layer at this time. For Re=100 this effect occurs even earlier (approximately t=9), while for Re=25 the circulation is still preserved at t=71 (figure 8.35). For Re=50 the circulation stops approximately at t=50.

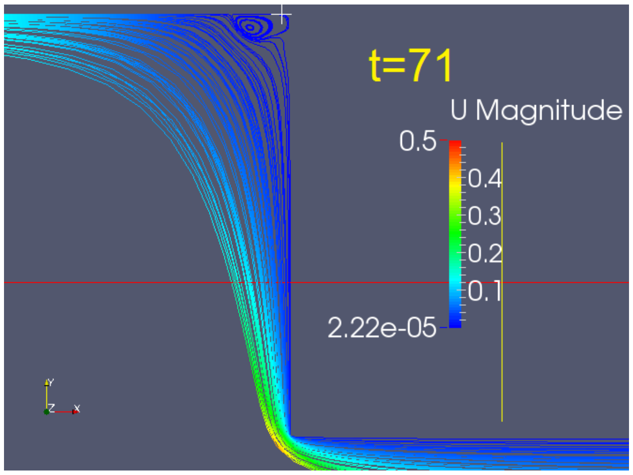

Figure 8.34: Streamlines showing area of circulation, Re=25, angle=0 degree with applied oil layer

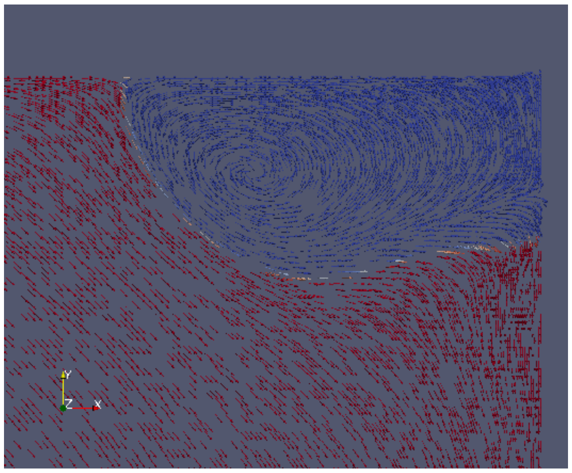

Figure 8.35: Velocity vectors showing area of circulation, Re=25, angle=0 with applied oil layer

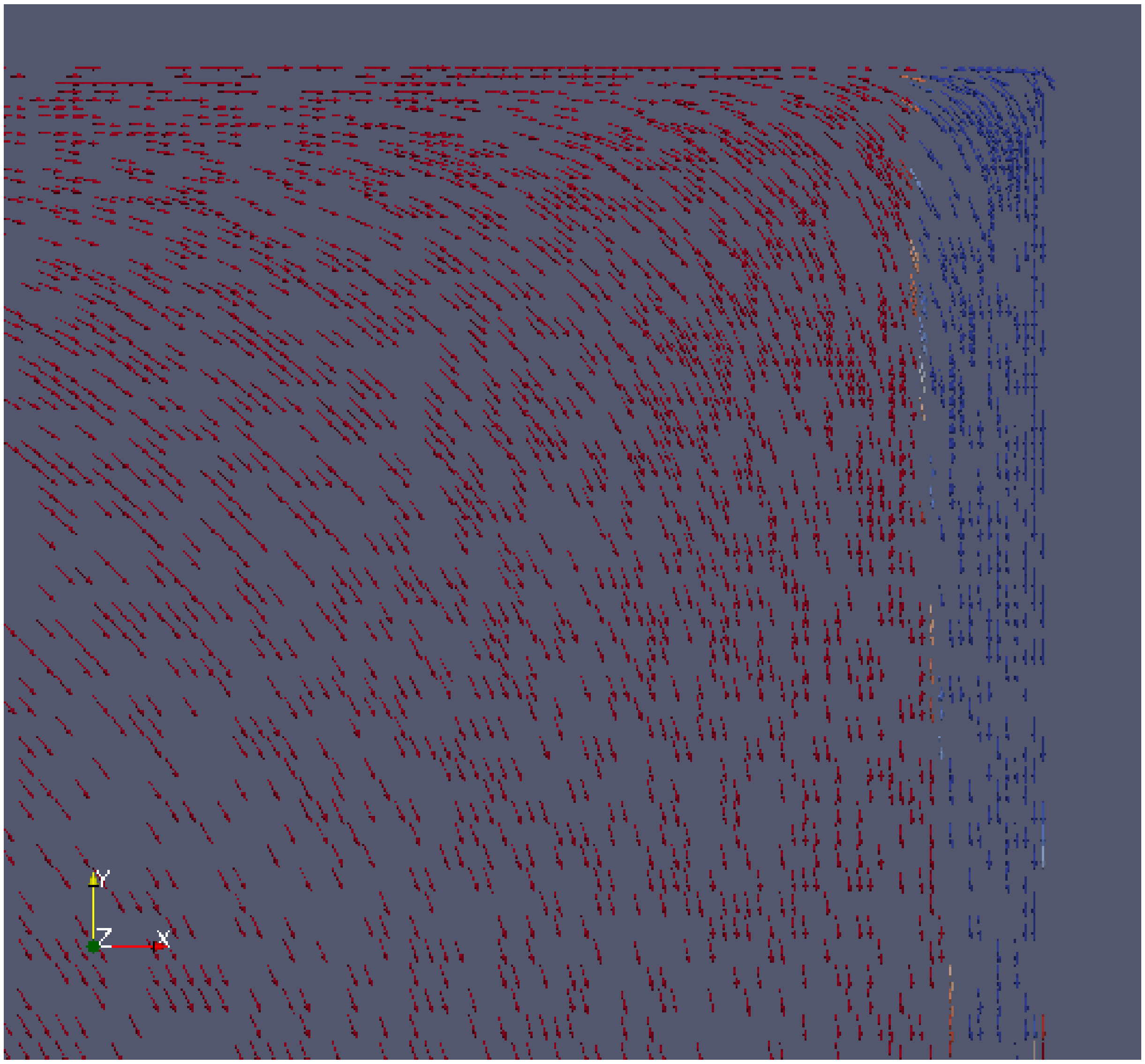

Figure 8.36: Velocity vectors showing area of circulation, Re=75, angle=0 with applied oil layer

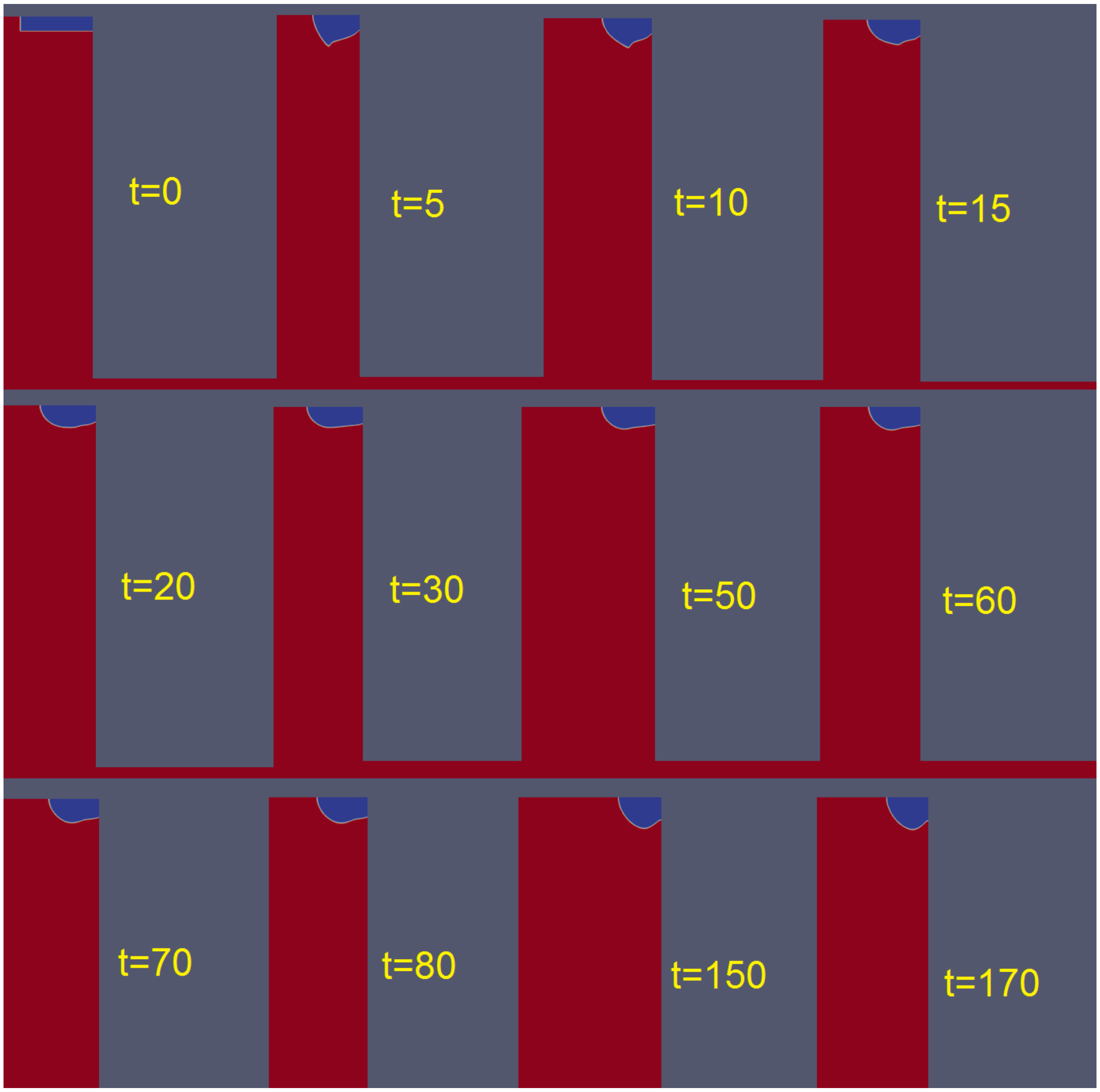

Figure 8.37: Flow pattern with Re=25

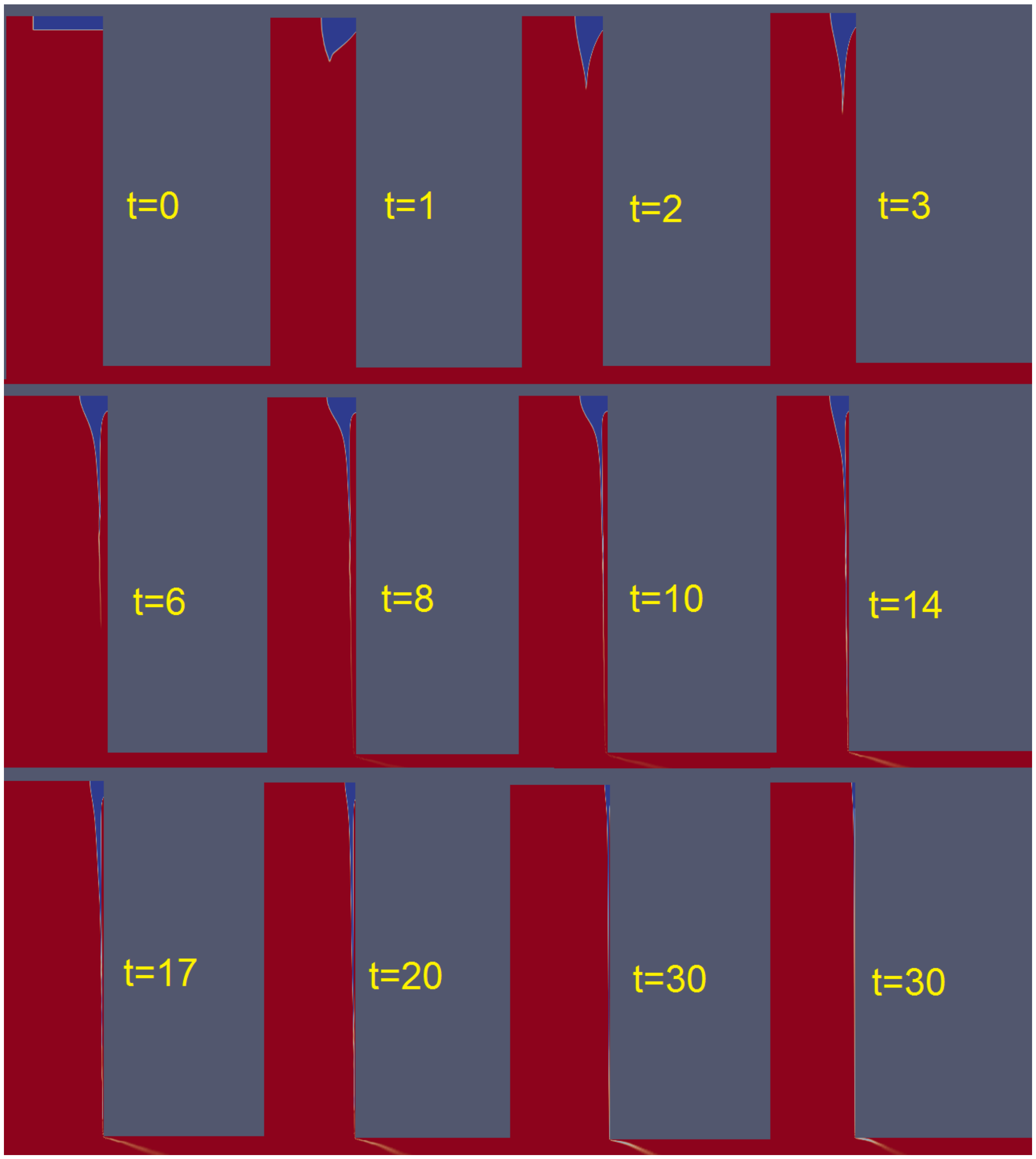

Figure 8.38: Flow pattern with Re=75

8.2.4 oil/water properties

The previous cases are simulated to get a better understanding of how the flow behave. With a higher difference in kinematic viscosity, the oil will move slower. This will be introduced in this section, with more realistic properties of crude oil.

Surface tension is set to 0.013 N/m, according to Mackay and Hossain (1982). This value depends on the oil which can be seen in Isehunwa and Olubukola (2012).

We use the following properties for oil and water:

Initially:

- $\rho_{water} = 1025 \frac{kg}{m^3}$
- $\nu_{water} = 10^{-6} \frac{m^2}{s}$
- $\rho_{oil} = 900 \frac{kg}{m^3}$
- $\nu_{oil} = 10^{-4} \frac{m^2}{s}$

Adjusted:

- $\rho_{water} = 1025 \frac{kg}{m^3}$
- $\nu_{water} = 0.01 \frac{m^2}{s}$
- $\rho_{oil} = 900 \frac{kg}{m^3}$
- $\nu_{oil} = 1 \frac{m^2}{s}$

adjusting CFL number

As mentioned in section 4.5.5, the CFL number should not exceed 0.5 in the interface region.

By running the initial 'controlDict' file from the 'Breaking of a dam' tutorial, the settings are:

- maxCo 1;
- maxAlphaCo 1;
- maxDeltaT 1;

This can cause numerical diffusion as the CFL number might be too high (close to 1) near the interface, i.e. maxAlphaCo is too high. In this case, we find that running the initial condition, i.e. maxAlphaCo = 1 is not sufficient. By running several cases, and refining the mesh, we find that maxAlphaCo has a big impact on the result. To see this we test 3 cases, with same boundary conditions as in section 8.1:

Using the marked area in figure 8.39 as a reference frame, we refine the mesh in the following way.

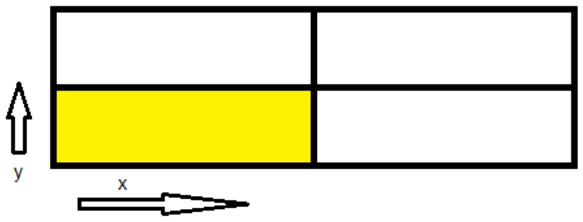

Figure 8.39: Reference for refining mesh

Case 1
x-dir.: 150 cells
y-dir.: 88 cells, $min.dl. = 0.021221$

Case 2
x-dir.: 200 cells
y-dir.: 120 cells, $min.dl. = 0.021735$

Case 3
x-dir.: 240 cells
y-dir.: 160 cells, $min.dl. = 0.012775$

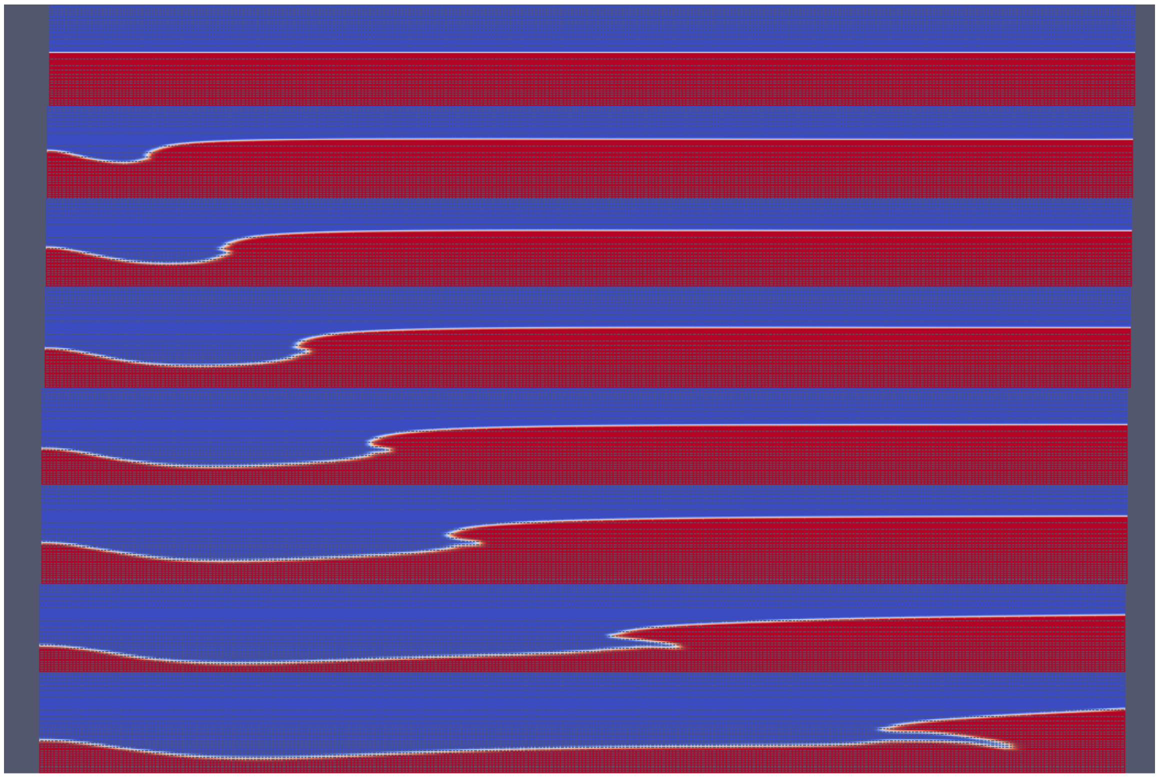

Figure 8.40: Case 1

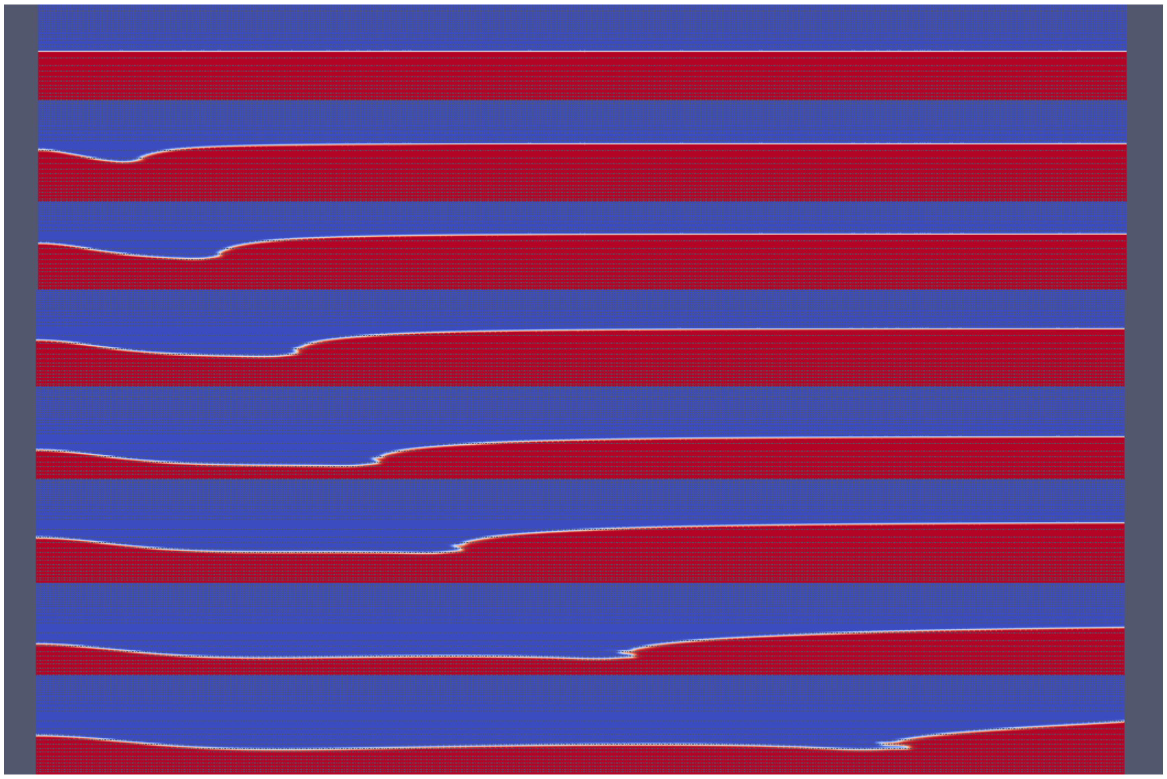

Figure 8.41: Case 2

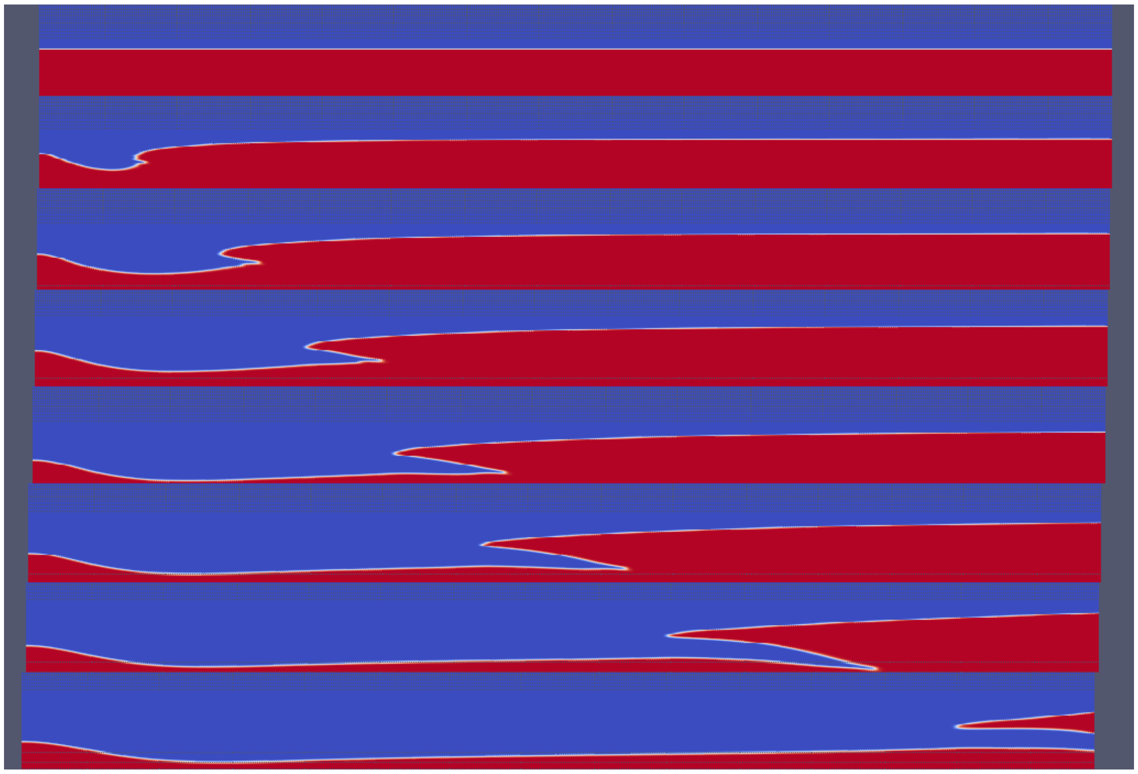

Figure 8.42: Case 3

We can clearly see numerical diffusion in these cases at the internal wave. Solving this requires us to lower the time steps. This is done by setting max CFL number to a lower limit near the interface, i.e. lowering maxAlphaCo. The two cases below shows the same mesh as in Case 2 (figure 8.41), but with maxAlphaCo set to 0.5 and 0.25. As seen from the results, a lower CFL number improves the results significantly. From here we will run the simulations with a CFL number below 0.25 around the interface, and in most cases as low as 0.15. Also Jasak and Weller (1995) states that the CFL number should in most cases be kept below 0.2.

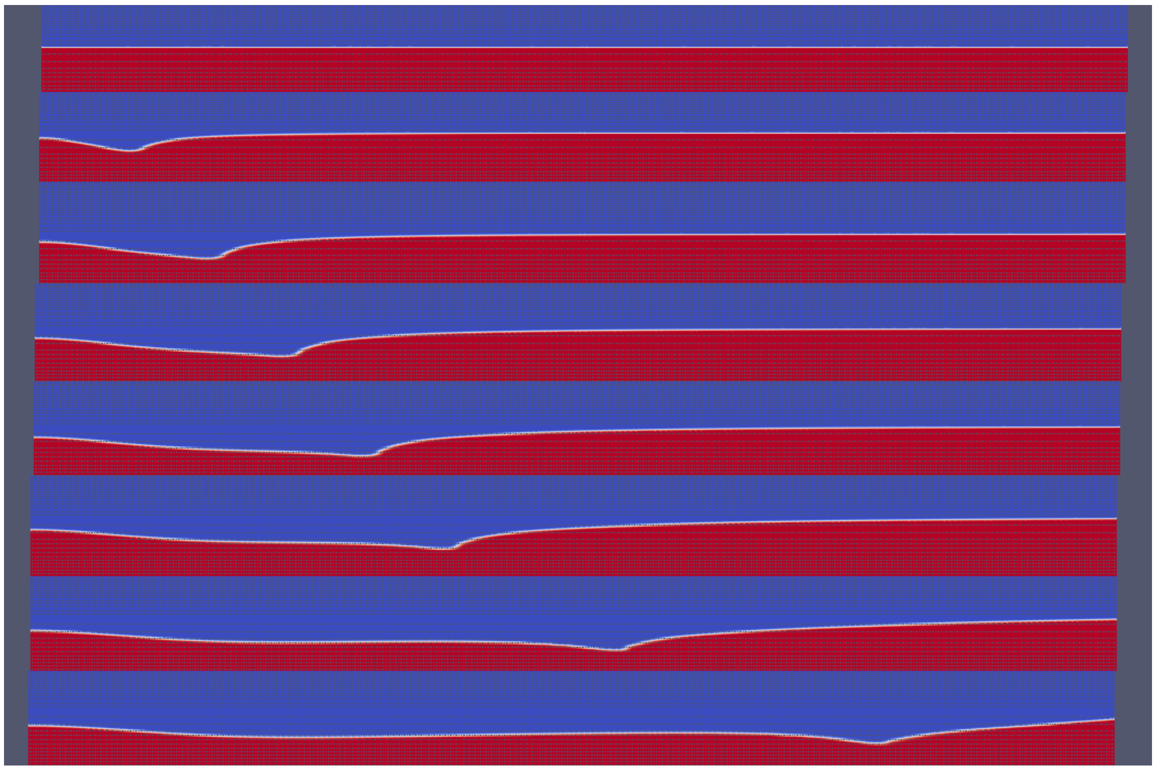
Figure 8.43: Case 2, with maxAlphaCo=0.5

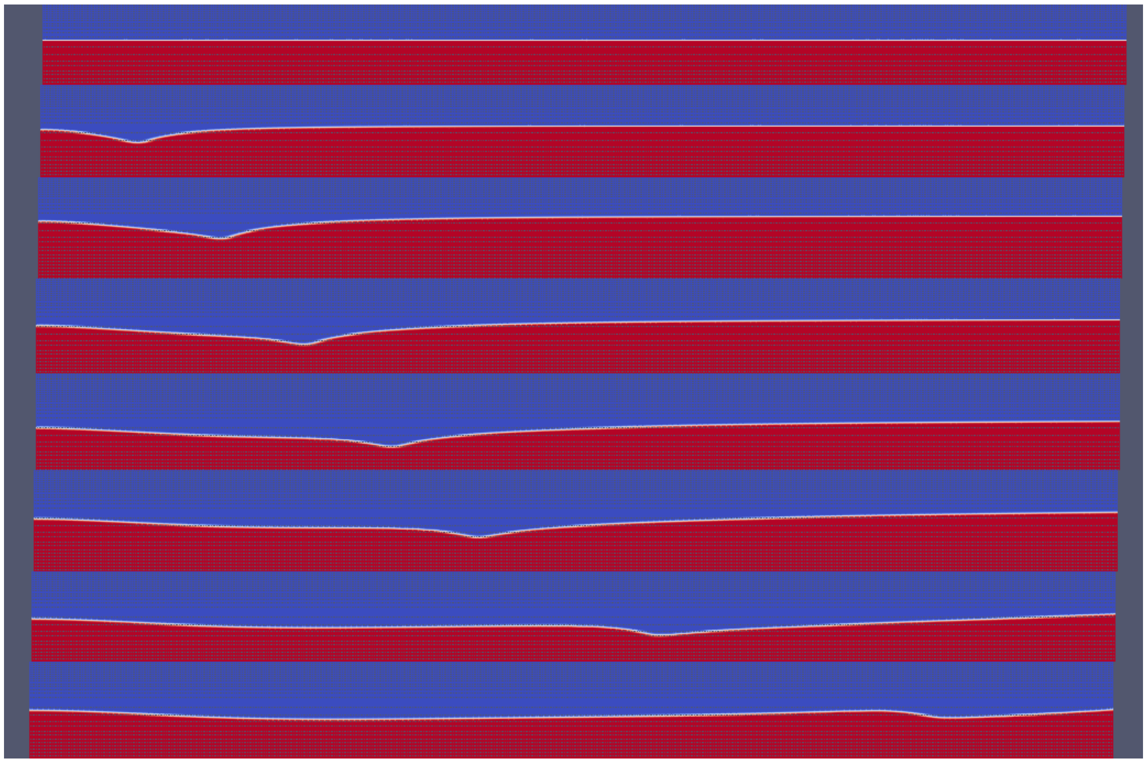
Figure 8.44: Case 2, with maxAlphaCo=0.25

adjusting Froude number

In this section we adjust the Froude number to see the effect on the internal wave. Froude number is calculated as in section 4.5.2. We use the depth of the oil layer as characteristic length in this case, i.e. L=5. The cases in section 8.2.4 are all run with Fr=0.14.

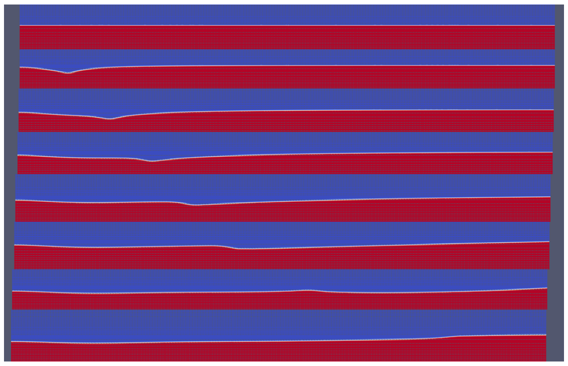
Figure 8.45: Case 2, with maxAlphaCo=0.25, Fr=0.095

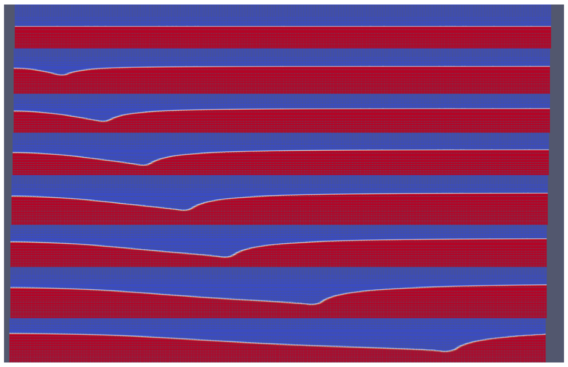
Figure 8.46: Case 2, with maxAlphaCo=0.25, Fr=0.56

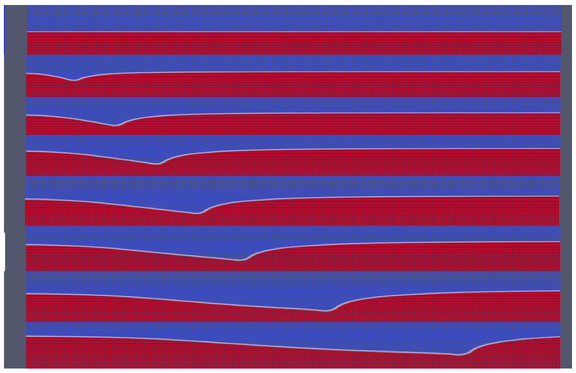

Figure 8.47: Case 2, with maxAlphaCo=0.25, Fr=1

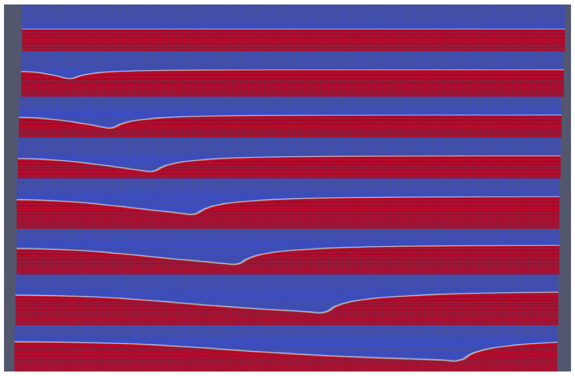

Figure 8.48: Case 2, with maxAlphaCo=0.25, Fr=1.4

Adjusting Fr will at some point, between Fr=0.14 and Fr=0.56, shift how the wave propagates. This is seen by comparing figure 8.44 and figure 8.46. Making Fr lower or higher results in the same behaviour as those two cases.

8.3 Drainage simulations with oil properties

Using the same domain dimensions as in figure 7.1, we apply a small layer of oil as in the section Structured grid under section 8.2. We use the same fluid properties as in section 8.2.4.

We set the following properties, by the experience from previous sections:

- Crank-Nicolson as temporal discretisation
- maxAlphaCo = 0.15
- nAlphaCorr: 10
- nAlphaSubCycles: 1
- cAlpha: 1
- nLimiterIter: 20
- tolerance: 1e-20
- the fvSchemes shown in section 8.2.2

The experience done in this work shows that simulating a flow with larger difference in viscosity is a harder task. A theory to this might be that it undermines the idea of VOF. With a higher viscosity, the oil layer moves more rigid and does not mix as well with the water. This can clearly be seen by comparing figure 8.49 with figure 8.31.

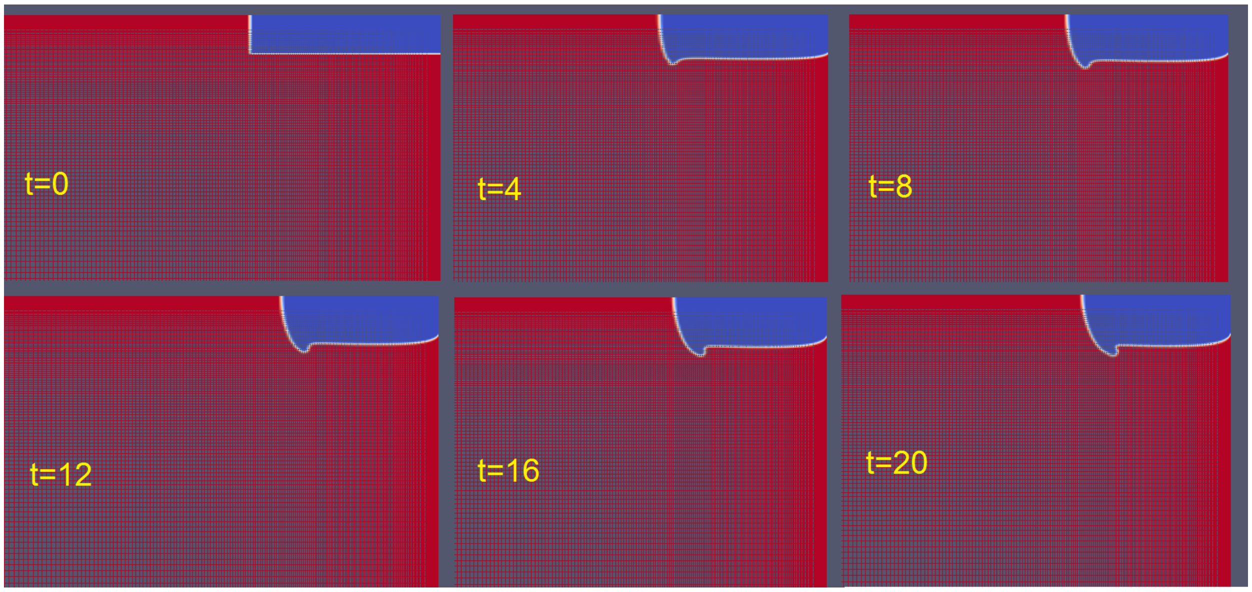

Figure 8.49: Flow pattern with $\nu_{oil} = 1$

The problem with this case occurs further out in the simulation. In contrary to the case with $\nu_{oil} = 0.03$, the horizontal surface in the case with $\nu_{oil} = 1$ will retain its shape much longer. In this case it is easy to see how compressing the horizontal surface easily leads to numerical diffusion, as seen at t=60 in figure 8.50.

Also, edges are more difficult to handle when high viscosity difference is imposed. This is also clearly seen in figure 8.50 where the edge seem to break of from the rest of the oil layer.

These problems are also seen for higher Re, which can be seen in Appendix (A.11).

The circulation area seems to be significantly smaller than for $\nu_{oil} = 0.3$, this can be seen by comparing figure 8.51 with figure 8.33.

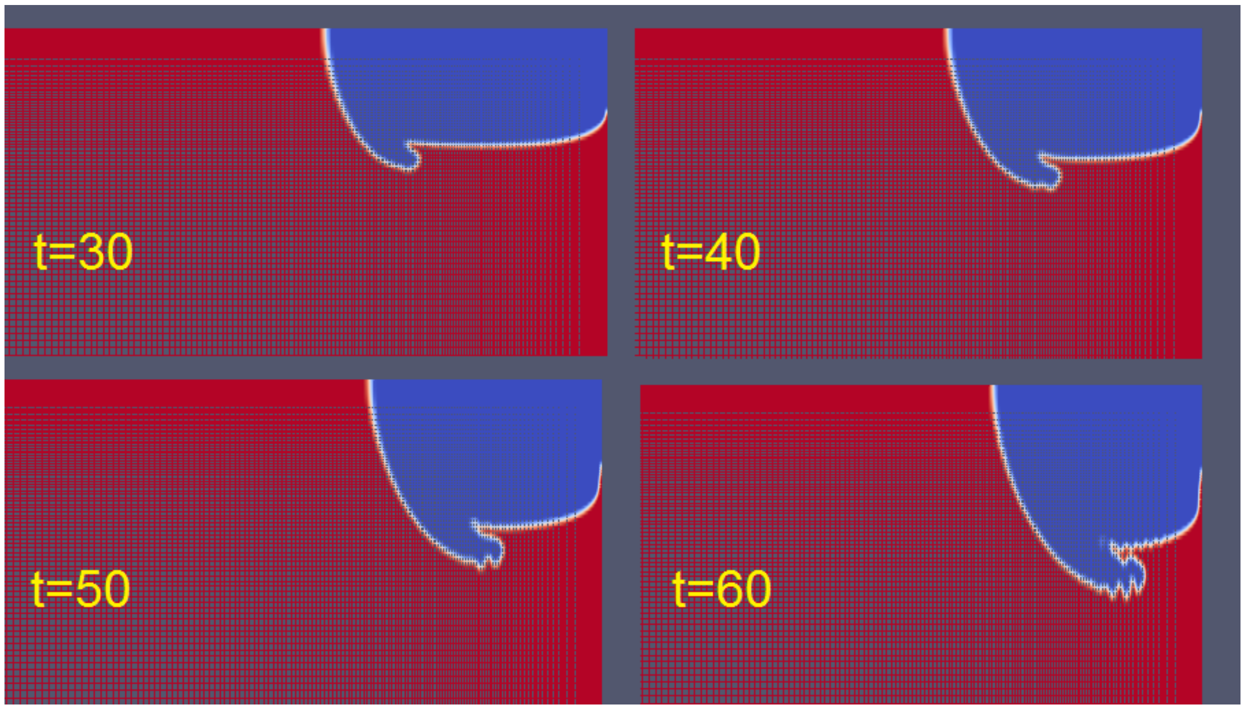

Figure 8.50: Flow pattern with $\nu_{oil} = 1$, continued

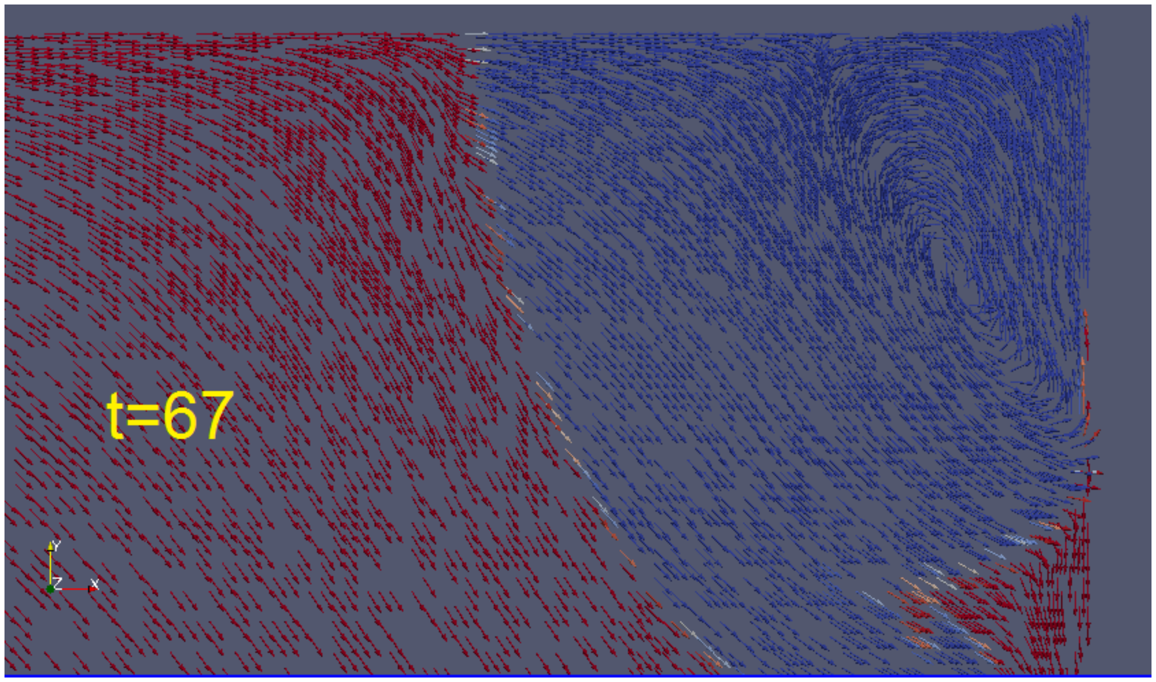

Figure 8.51: Velocity vectors showing area of circulation, Re=50, angle=0 with applied oil layer, with $\nu_{oil} = 1$

8.4 Typical problems encountered

8.4.1 Grid increments

A problem encountered in this thesis is that the mesh generator MEGA does not support more than 300 cells over a line, i.e. 300x300 cells is maximum in a block. Adding more than 300 cells over a line causes a segmentation fault. This can be fixed in the source-code of MEGA.

By making smaller increments and adding more cells, we can keep the min.dl. for the inner cells, i.e. the cells close to the body. Making increments towards both the vertical interface and boundary layer near the object, makes the cells in between less refined. As seen by the pictures below, problems can occur when the vertical interface moves into these less refined cells. A close-up on the marked region can be seen in Appendix, (A.3).

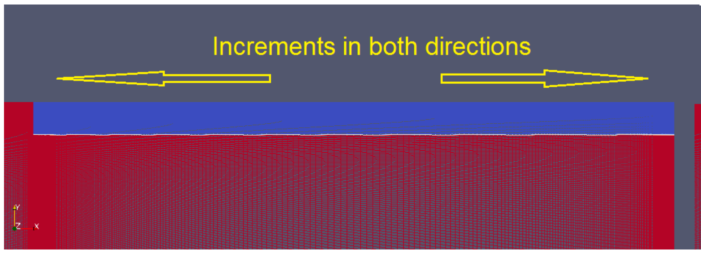

Figure 8.52: Dimensions of oil layer

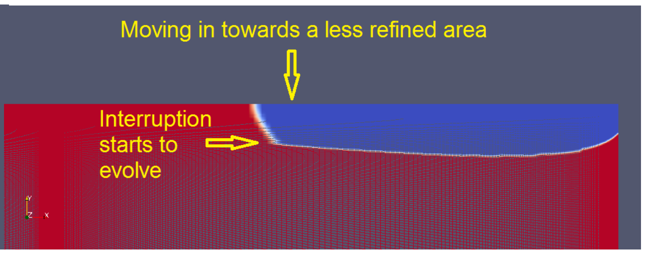
Moving in towards a less refined area
Interruption starts to evolve

Figure 8.53: grid refinement at interfaces

8.4.2 Continuous flow of oil

A more realistic approach to an oil spill is a continuous flow of oil. This can be simulated by setting the oil layer in the file "setFieldsDict" all the way to the inflow. However, as the interface layer needs to be well refined it is harder to keep an aspect ratio close to 1 near the inlet. This can make problems, as seen in figure 8.54, that the oil layer gets thicker at the inflow from the start.

The velocity vectors seem to point slightly downward, also in the water phase in figure 8.55. In figure 8.55 the time steps are 0, 0.1, and 0.2 respectively.

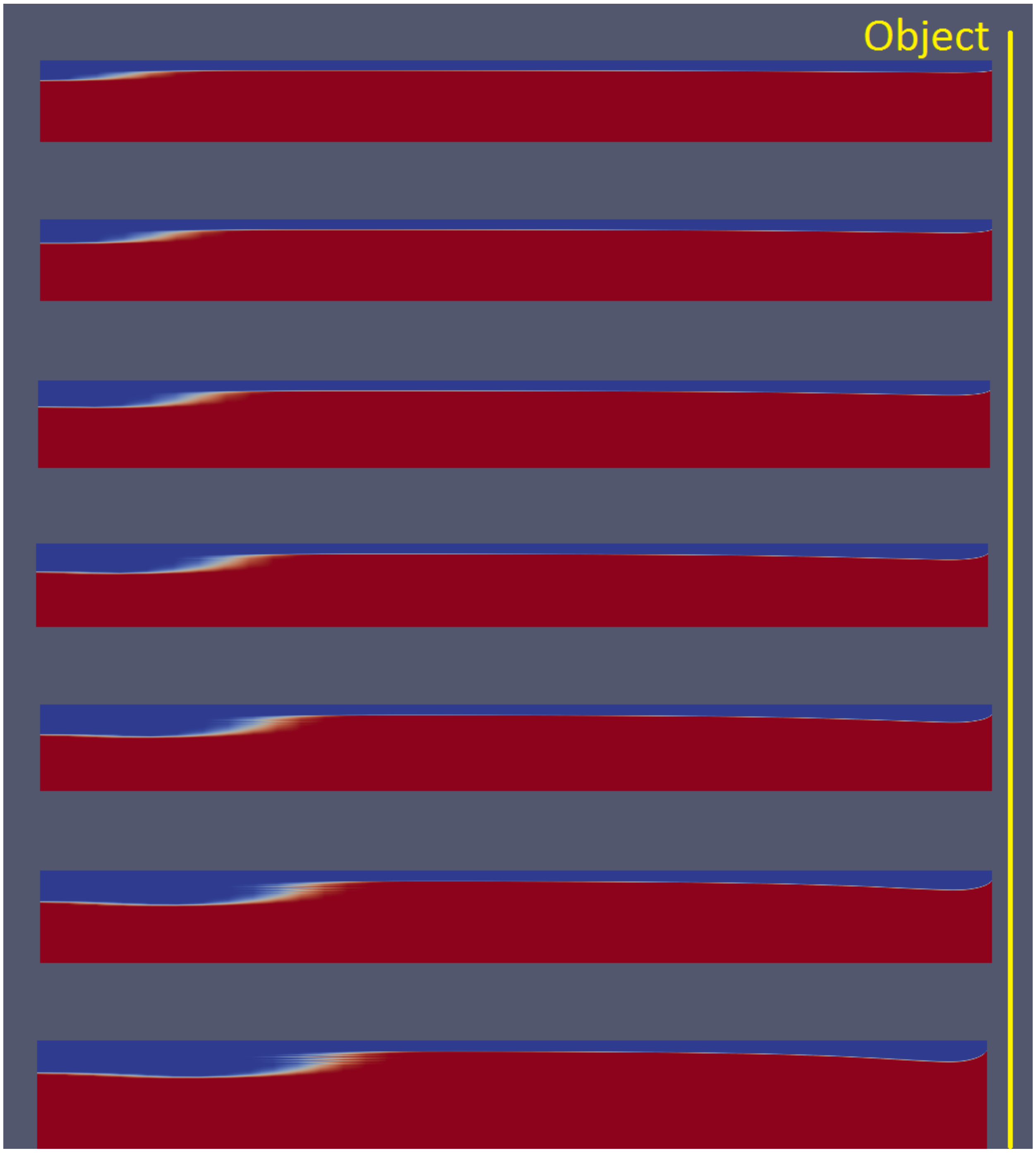

Figure 8.54: Numerical diffusion at inflow

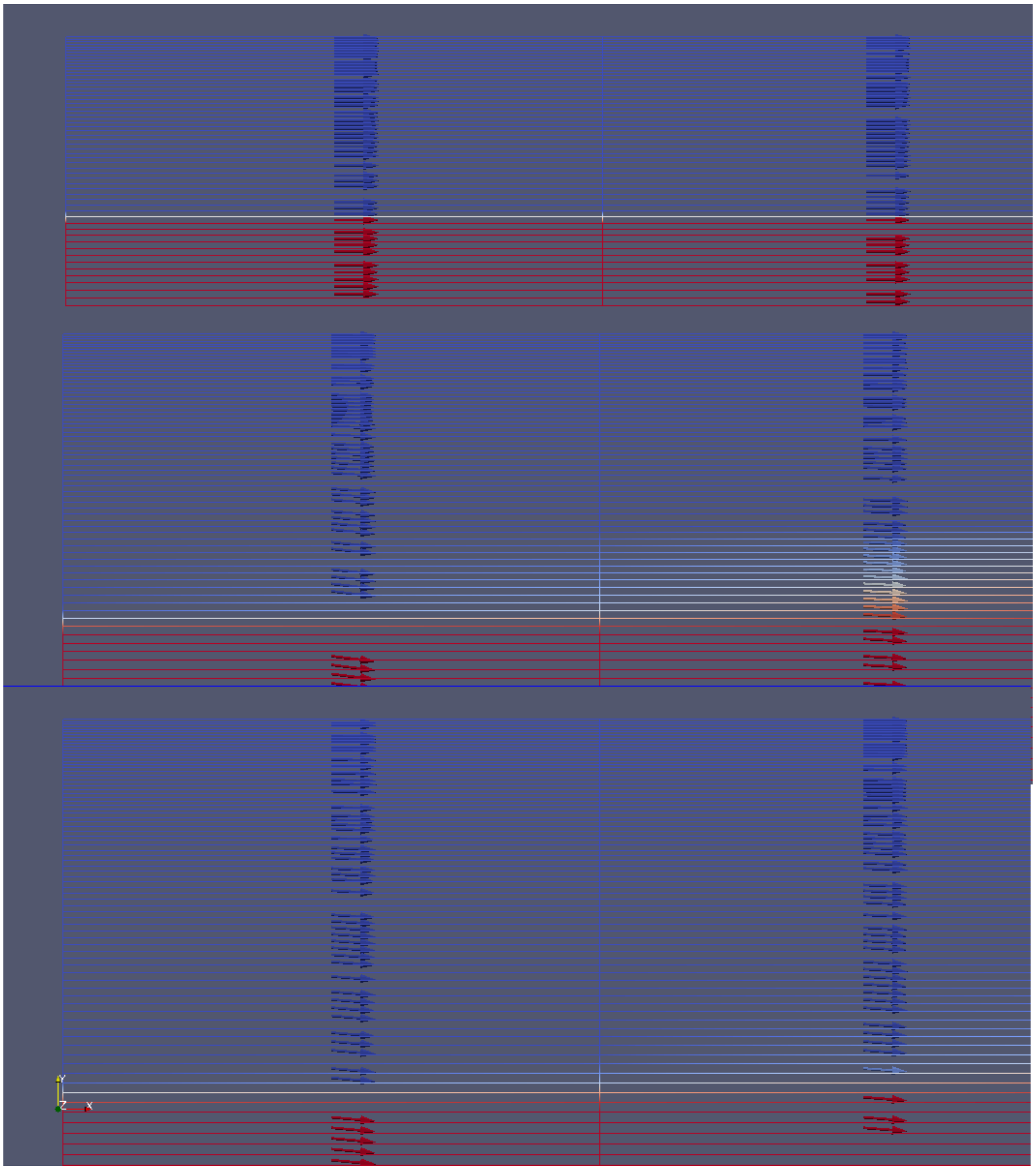

Figure 8.55: Velocity vectors at inflow

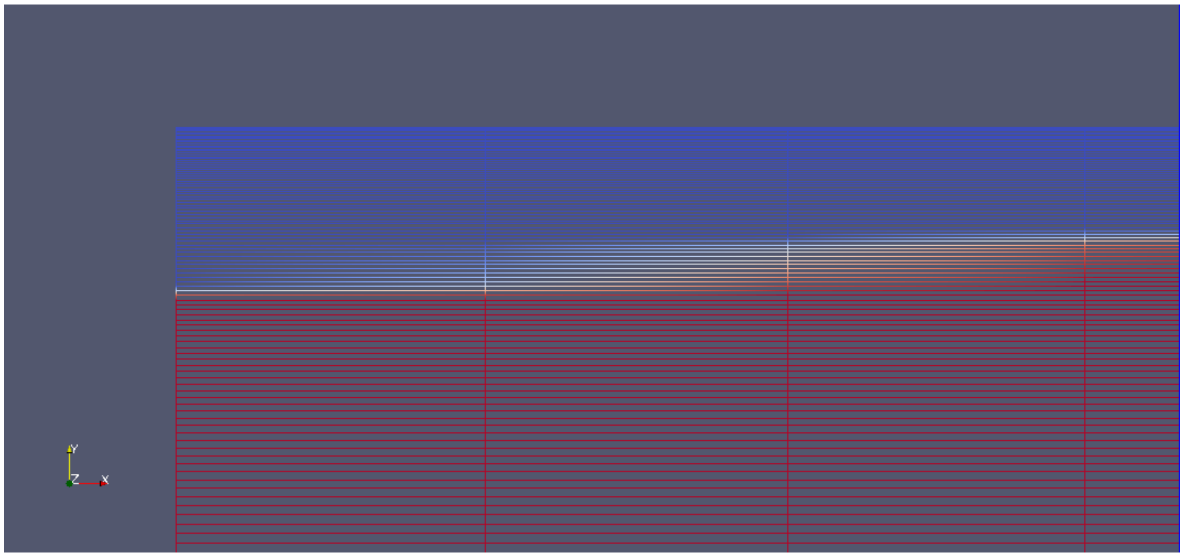

Figure 8.56: Numerical diffusion at inflow, zoomed in

What we have learned from the previous cases is that the interface between oil and water are highly sensitive to the mesh configuration. Therefore, a solution to this problem can be found by studying the channel case, as in section 8.1. We present the solution in the section below.

Solution to the continuous flow problem

The solution to the continuous flow problem, i.e. the velocity vectors pointing slightly downwards at the inlet, is found in a structured grid.

In this case the aspect ratio, width/height, is very high. This causes a numerical diffusion at the inlet. This is clearly seen in the case below, figure 8.57 and figure 8.58, where the velocity vectors at the inlet cell is pointing slightly upwards.

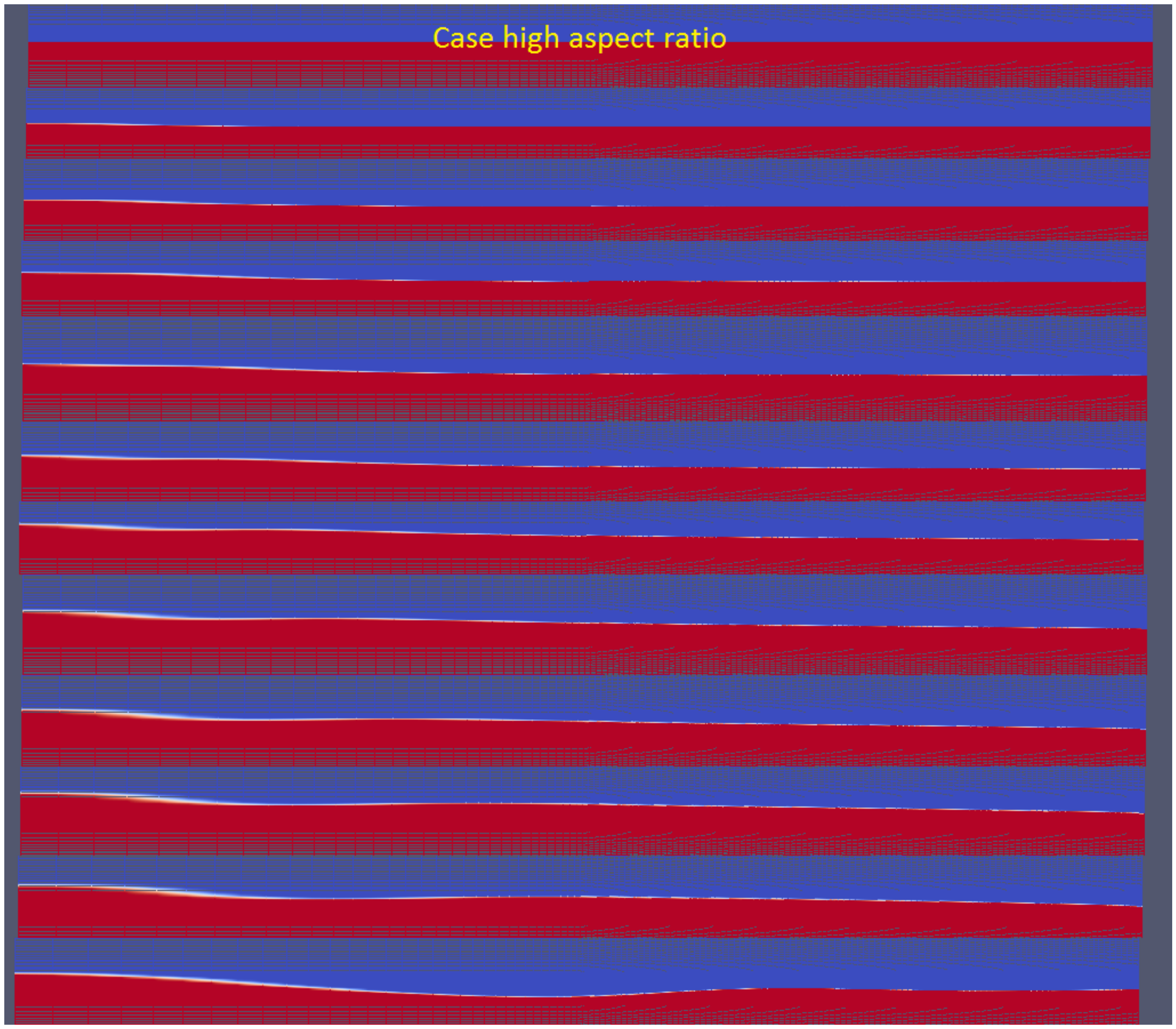

Figure 8.57: Case high aspect ratio

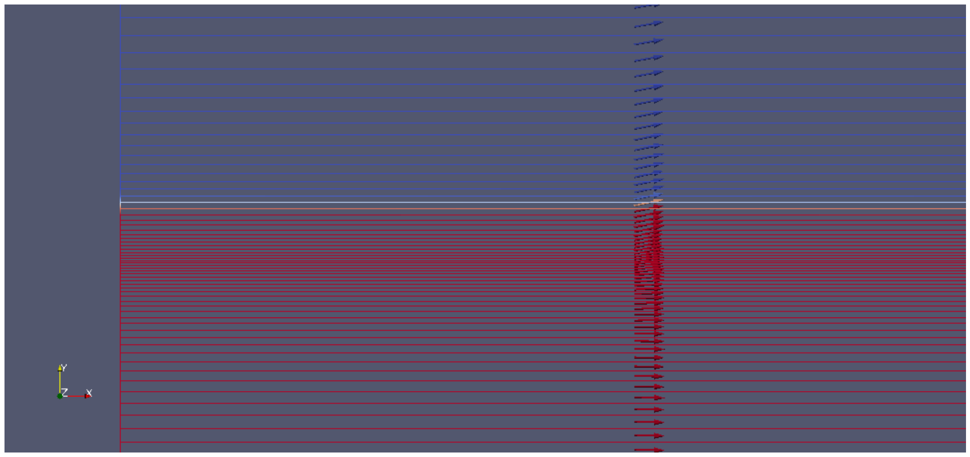

Figure 8.58: Case high aspect ratio, zoomed in

For cells with an aspect ratio close to 1, as in figure 8.59, we see that the inlet value of uniform (1 0 0) is preserved.

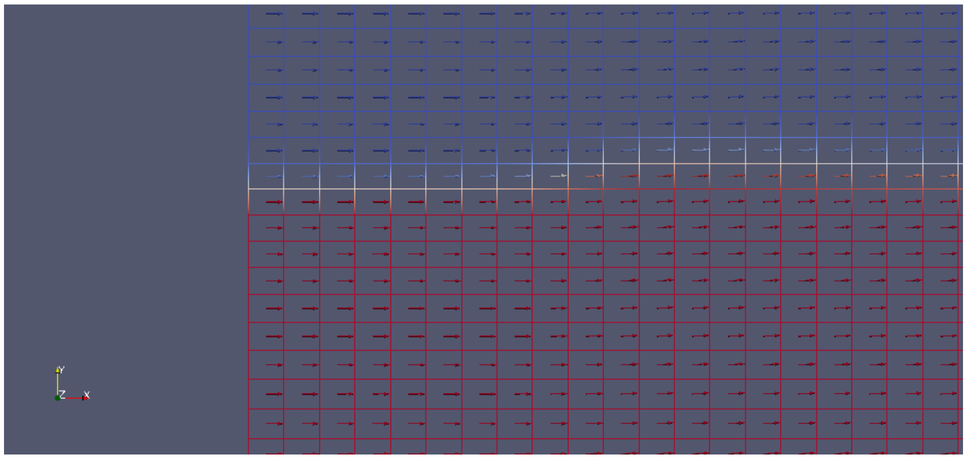

Figure 8.59: Case aspect ratio close to 1

As seen before, in figure 8.11, a low aspect ratio is more severe than a high aspect ratio. This is clearly seen in the figure 8.60, where low aspect ratio cells are imposed in the centre of the domain.

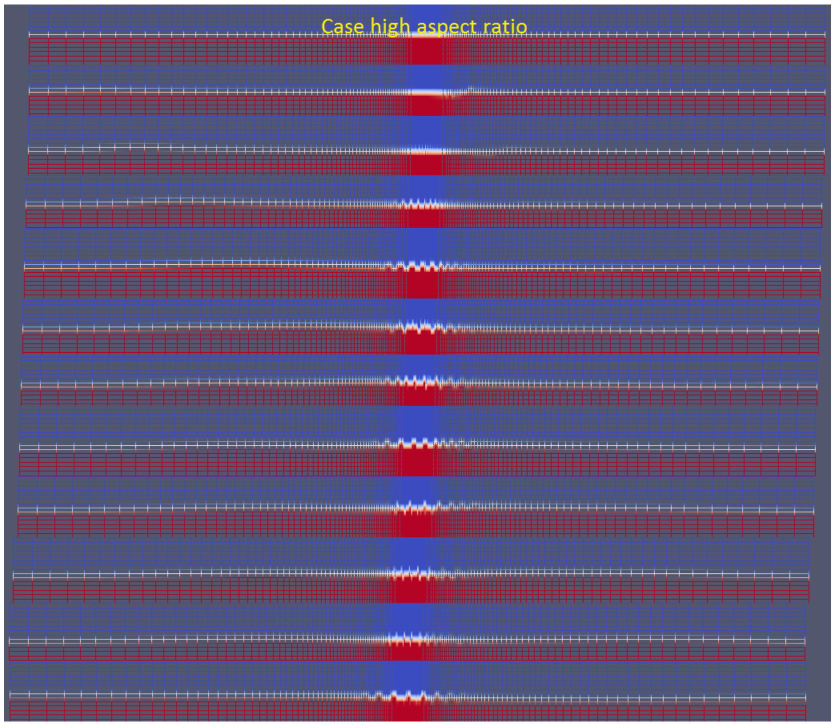

Figure 8.60: Case high aspect ratio

Chapter 9

Conclusion and further work

To test the interFoam solver we performed a simulation with equal conditions as for the pisoFoam solver, i.e. we applied a small layer of another fluid far away from the body and set the properties of this other fluid equal to water. Doing this we have tricked the two-phase interFoam solver into a one-phase solver. The simulations done give equal results for drag and lift coefficients, velocity distribution, and pressure and viscous forces. Also, a comparison with published work (using FLUENT) is done, which shows good compliance for velocity distribution and streamlines.

In order to prevent drainage under the object, we need to preserve the circulation at the face of the object. A convergence study with one-phase simulations are performed to find the necessary grid resolution to preserve the circulation. This study shows that 180x300 cells are necessary in a small area in front of the object. This area measures 0.4x0.5 (the object measures 0.5x05). We also checked the how different Re and different angle affects the circulation area. These simulations show that higher Re decrease the circulation area, and the circulation totally vanishes at Re=300. Increasing the angle significantly increases the area of circulation.

Previous studies using interFoam have shown that modelling the interface is a hard task. A channel case is therefore studied before moving on to the circulation problem. In the channel case we have tested how the interface of the fluids behave when alternating grid, density, viscosity, CFL number and Fr. The interface is found to be highly sensitive to unstructured mesh. This is in accordance with the theory behind the VOF method, i.e. the method tries to align the interface with the grid lines. Especially if the interface is distorted initially, this error will be carried throughout the simulation. It is therefore very important to model the interface as smooth as possible. This study also shows that the aspect ratio of the cells close to the interface should be as close to 1 as possible. Another important issue is the CFL number. The work done in this thesis shows that the CFL number close to the interface, i.e. maxAlphaCo, should be kept below 0.5 and preferably below 0.25.

The circulation of fluid simulations are shown to be a much harder problem to simulate than the channel case. The interface will move over a larger area, both horizontally and vertical. There will also be both a horizontal and vertical interface to be traced, which means the mesh needs to be refined significantly in both directions.

In this thesis we have tested different schemes for different operators, and also different conditions for the algorithm (MULES) used to solve the volume fractions, i.e. trace the interface. We find that the optimal criteria for MULES in this case is to raise nAlphaCorr from 2 to 10 corrections, raise nLimiterIter to 20 from 3, and lower the iteration tolerance from 1e-8 to 1e-20. This does not significantly increase computation time. Increasing nAlphaSubCycles does however increase computation time significantly. Increasing this criteria is proved to improve the results, but the experience in this thesis is that the simulation time is multiplied by nAlphaSubCycles.

Dealing with an unstructured grid in this problem was shown to be a challenging task, even when improving the algorithm criteria. Simulations performed with a structured grid shows realistic results with no distorted interface. Adding an oil layer to the one-phase study increase the circulation area significantly. Simulations with Re=25, Re=50, Re=75 and Re=100 is performed with a small oil layer ($\frac{1}{5}$ the length of the object). A longer oil layer demands significantly larger amount of cells horizontally to sufficiently refine the vertical interface.

Increasing the viscosity difference between the fluids is proved to be challenging both by the circulation simulations and the channel case simulations. As the oil layer will behave more like a fixed wall a theory might be that this undermines the theory of the VOF method as the fluid does not mix well together. Numerical errors occur much easier than with smaller viscosity difference.

This work can be taken further in many ways. Some suggestions for further work are:

- Introduce a continuous oil layer. This will be a harder task as the interface will be stretched over a much larger area both in horizontal and vertical (as it moves down, due to the body) direction.
- Include a third phase, i.e. air. For example can the solver multiphaseInterFoam be used to add an additional phase.
- Include waves. The package waves2Foam can be used to generate waves (valid up to OF 2.3.0). Also, the user should consider using interDyMFoam which does mesh modifications at specific areas (e.g. sharpening interface).
- Introduce a dynamic mesh which can handle the movements of the object

Bibliography

Amini, A. and Schleiss, A. J. (2009). Numerical Modelling of Oil-Water Multiphase Flow Contained by an Oil Spill Barrier. *Engineering Applications of Computational Fluid Mechanics **3**(2), 207-219.*

Bakker, A. (2006). *Lecture 6 - Boundary Conditions, Applied Computational Fluid Dynamics*, distributed for class ENGS 150 at Dartmouth College. `http://www.bakker.org/dartmouth06/engs150/06-bound.pdf`.

Brackbill, J. U., Kothe, D. B., and Zemach, C. (1992). A Continuum Method for Modeling Surface Tension. *Journal of Computational Physics **100**(2), 335-339.*

Butler, J. S. (2011). BP Macondo Well Incident U.S Gulf of Mexico Pollution Containment and Remediation Efforts. *2011 Lillehammer Energy Claims Conference.*

Cengel, Y. A. and Cimbala, J. M. (2009). *Fluid Mechanics, Fundamentals and Applications (pp. 420-450).* McGraw-Hill Professional, 2nd edition.

Coil, D., Lester, E., and Higman, B. (2010). Exxon Valdez Oil Spill. `http://www.groundtruthtrekking.org/Issues/AlaskaOilandGas/ExxonValdezSpill.html`.

Damian, S. M. (2013). *An Extended Mixture Model for the Simultaneous Treatment of Short and Long Scale Interfaces* (pp.95). Phd thesis, Universidad Nacional Del Litoral.

Deshpade, S. S., Anumolu, L., and Trujillo, M. F. (2012). Evaluating the performance of the two-phase flow solver interFoam. *Computational Science & Discovery **5**, 2-14.*

Emad, V. (2014). *Evaluating the Performance of Various Convection Schemes on Free Surface Flows by Using Interfoam Solver* (pp. 2-17)(pp.49-76). Master's Thesis, Eastern Mediterranean University, Department of Mechanical Engineering.

Exxon Valdez Oil Spill Trustee Council (2015). Oil Spill Facts. `http://www.evostc.state.ak.us/index.cfm?FA=facts.QA`.

Gopala, V. R. and Wachem, B. G. M. (2008). Volume of fluid methods for immiscible-fluid and free-surface flows. *Chemical Engineering Journal **141**(1-3), 204-221.*

Greco, M. (2012). Tmr4215: Sea loads, lecture Notes, distributed at NTNU.

Greenshields, C. J. (2015a). OpenFOAM User Guide: 4.5 Solution and algorithm control. `http://cfd.direct/openfoam/user-guide/fvsolution/`.

Greenshields, C. J. (2015b). OpenFOAM User Guide (pp.u-62 - u-70). `http://foam.sourceforge.net/docs/Guides-a4/UserGuide.pdf`.

Gulf Coast Ecosystem Restoration Council (2010). The ongoing administration-wide response to the deepwater bp oil spill september 14 2010. `https://www.restorethegulf.gov/release/2015/07/01/ongoing-administration-wide-response-deepwater-bp-oil-spill`.

Harlow, F. H. and Welch, J. E. (1965). Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *The Physics of Fluids **8**(12), 2182-2189.*

Herreras, N. and Labeaga, J. I. and, O. (2013). *Two-Phase pipeflow simulations with OpenFoam* (pp.48). Master's Thesis, Norwegian University of Science and Technology, Department of Energy and Process Engineering.

Heyns, J. A. and Oxtoby, O. F. (2014). MODELLING SURFACE TENSION DOMINATED MULTIPHASE FLOWS USING THE VOF APPROACH. *11th World congress on Computational Mechanics.*

Holleman, M. (2014). After 25 years, Exxon Valdez oil spill hasn't ended. `http://edition.cnn.com/2014/03/23/opinion/holleman-exxon-valdez-anniversary/`.

Isehunwa, S. O. and Olubukola, O. (2012). INTERFACIAL TENSION OF CRUDE OIL-BRINE SYSTEMS IN THE NIGER DELTA. *International Journal of Research and Reviews in Applied Sciences **10**(3), 460-464.*

Issa, R. J. (1984). Solution of the Implicity Discretised Fluid Flow Equations by Operator-Splitting. *Journal of Computational Physics **62**, 40-65.*

Ivshina, I. B., Kuyukina, M. S., Krivoruchko, A. V., Elkin, A. A., Makarov, S. O., Cunningham, C. J., Peshkur, T. A., Atlas, R. M., and Philp, J. C. (2015). Oil spill problems and sustainable response strategies through new technologies. *Environmental Science: Processes & Impacts.*

Jasak, H. and Weller, H. G. (1995). *Interface* T*racking* C*apabilities of the* I*nter-*G*amma* D*ifferencing* S*cheme.* Technical report, Imperial College, University of London.

Jasak, H. and Weller, H. G. (1999). Application of the finite volume method and unstructured meshes to linear elasticity. *International Journal for Numerical Methods in Engineering **48**, 270-273.*

Keim, B. (2009). The Exxon Valdez Spill is All Around Us. `http://www.wired.com/2009/03/valdezlegacy/`.

Kwak, J. (2012). *Design of a new oil boom* (pp.5). Master's Thesis, Korea Advanced Institute of Science and Technology, Department of Ocean Systems Engineering.

Mackay, D. and Hossain, K. (1982). Interfacial Tensions of Oil, Water, Chemical Dispersant Systems. *Canadian Journal of Chemical Engineering **60**(4), 546-550.*

Met office (2010). Beaufort, National Meteorological Library and Archive Fact sheet 6 - The Beaufort Scale. `http://www.metoffice.gov.uk/media/pdf/b/7/Fact_sheet_No._6.pdf`.

No, W. F. and Woodward, P. (1976). SLIC (Simple Line Interface Calculation). *5th International Conference of Fluid Dynamics.*

OpenFOAM Foundation (2015a). Numerical Method. `http://www.openfoam.org/features/numerical-method.php`.

OpenFOAM Foundation (2015b). Openfoam release notes for version 1.7.0. `http://www.openfoam.org/archive/1.7.0/docs/release-notes.php`.

OpenFOAM Foundation (2015c). Openfoam Release History. `http://www.openfoam.org/download/history.php`.

Ornitz, B. E. and Champ, M. A. (2002). *Oil Spills First Principles: Prevention and Best Response (pp. IX-XXII)*, volume 1. Elsevier Science Ltd.

Popinet, S. (2010). Gerris Flow Solver. `http://gfs.sourceforge.net/wiki/index.php/Main_Page`.

Pringuey, T. (2012). *Large Eddy Simulation of Primary Liquid-Sheet Breakup* (pp.69-79). Phd thesis, University of Cambridge.

Ramshaw, J. D. (2011). *Elements of Computational Fluid Dynamics (pp. 71-73)*, volume 2. Imperial College Press.

Ransan, S. (n.d.). Lecture notes for the course: Numerical Methods in Marine Hydrodynamics, distributed at NTNU.

Rønningsen, H. P. (2012). Rheology of Petroleum Fluids. *Annual Transactions of the Nordic Rheology Society **20**, 11-18.*

Ruifeng, S. and Xun, M. (2014). Numerical Simulation and Structural Analysis of Oceanic oil Containment Booms in Towing Condition. *Proceedings of the Twenty-fourth (2014) International Ocean and Polar Engineering Conference.*

Rusche, H. (2002). *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions* (pp. 99-162). Phd thesis, Imperial College.

Schulze, L. and Thorenz, C. (2014). The Multiphase Capabilities of the CFD Toolbox OpenFOAM for Hydraulic Engineering Applications. *ICHE, 1007-1014.*

Skytruth (2010). BP / Gulf Oil Spill - 68,000 Square Miles of Direct Impact. `http://blog.skytruth.org/2010/07/bp-gulf-oil-spill-68000-square-miles-of.html`.

The Engineering Toolbox (2015). The Engineering ToolBox. `http://www.engineeringtoolbox.com/kinematic-viscosity-d_397.html`.

Tryggvason, G., Bunner, B., Ebrat, O., and Tauber, W. (1998). Computations of Multiphase Flows by a Finite Difference/Front Tracking Method. I. Multi-Fluid Flows. *Lecture notes for the 29th Computational Fluid Dynamics Lecture Series.*

United States Coast Guard (1993). *Federal On Scene Coordinator's Report T/V Exxon Valdez Oil Spill*, **1** 36. United States Coast Guard, Department of Transportation.

United States Coast Guard (2011). *On Scene Coordinator Report Deepwater Horizon Oil Spill*, (pp. vi). United States Coast Guard, Department of Transportation.

Violeau, D., Buvat, C., Abed-Meraim, K., and de Nanteuil, E. (2007). Numerical modelling of boom and oil spill with SPH. *Coastal Engineering* ***54****, 895-913*.

Warren, S. (2011). BP Oil Still Ashore one Year After End of Gulf Spill. `http://www.bloomberg.com/news/articles/2011-07-15/bp-oil-still-washing-ashore-one-year-after-end-of-gulf-spill`.

Appendix A

Additional results

A.1 One-phase circulation at Re=50

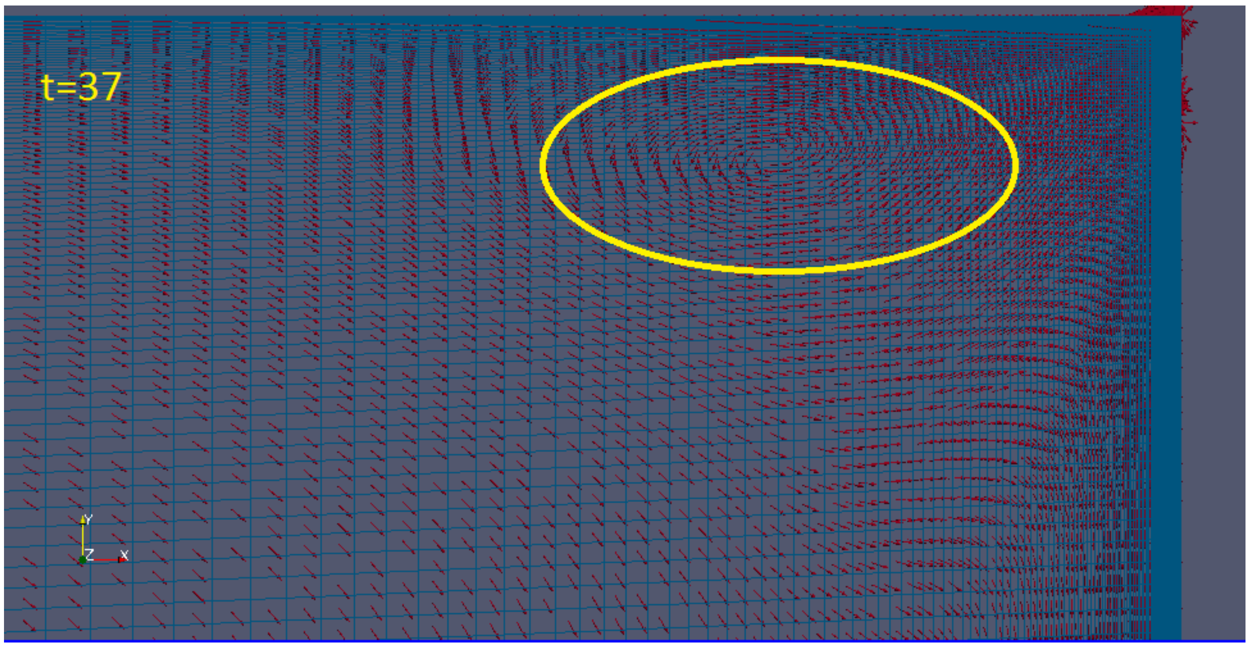

Figure A.1: Time 37

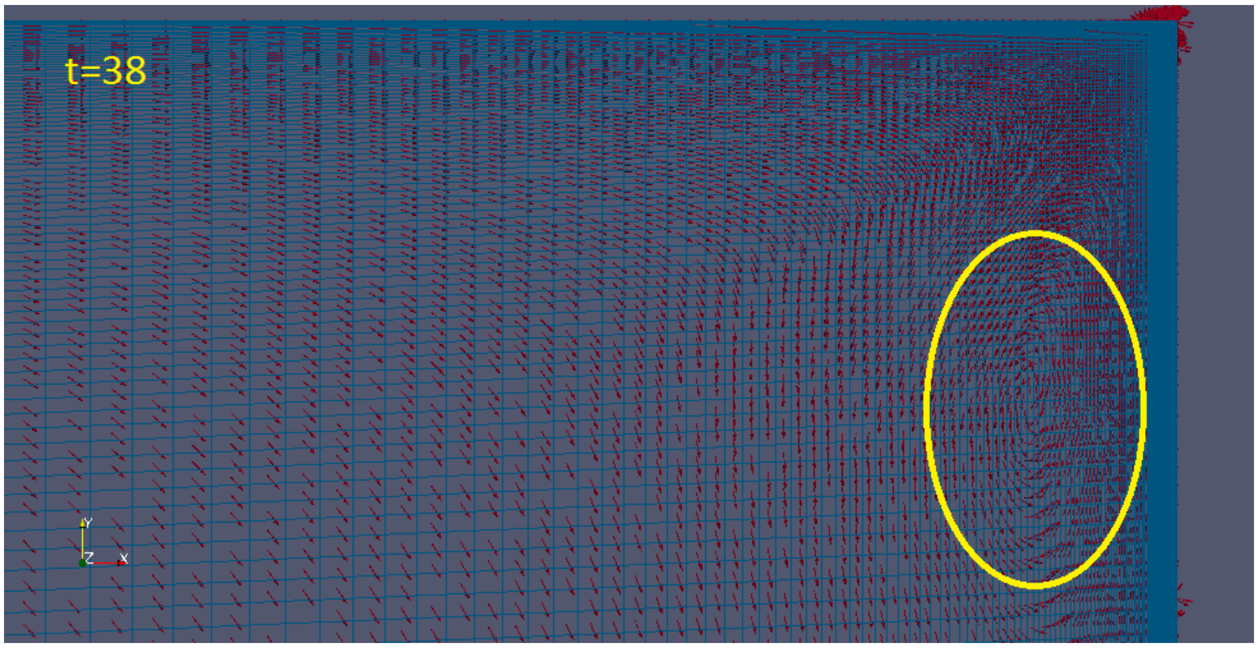

Figure A.2: Time 38

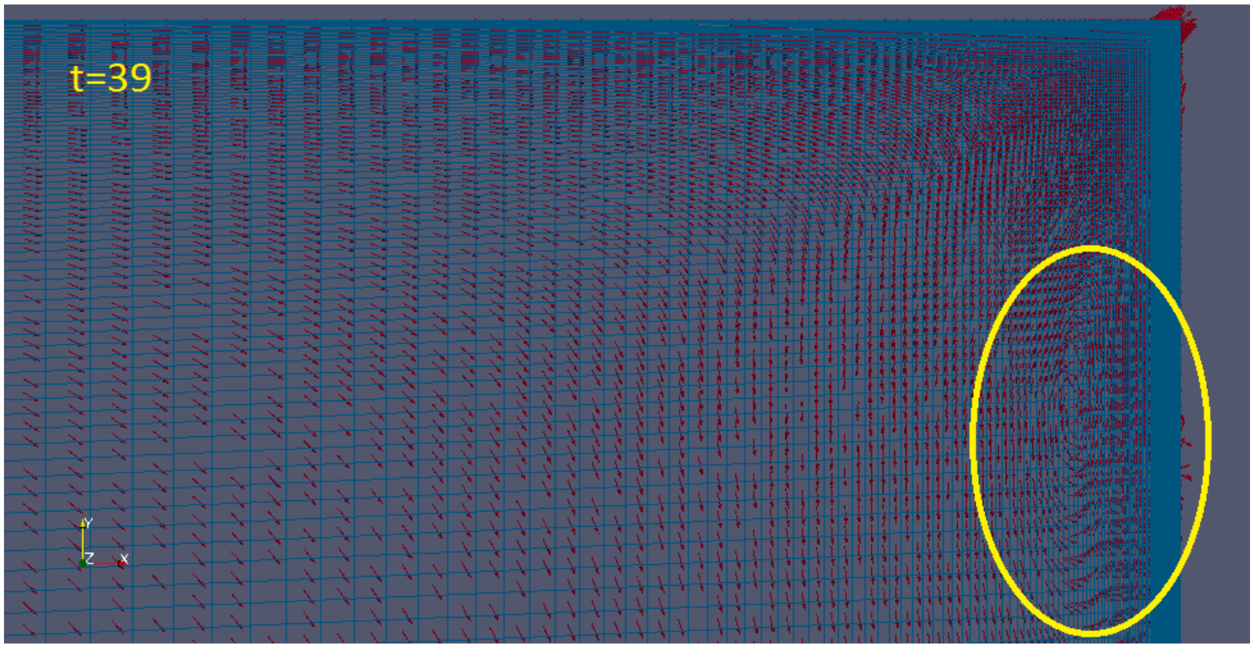

Figure A.3: Time 39

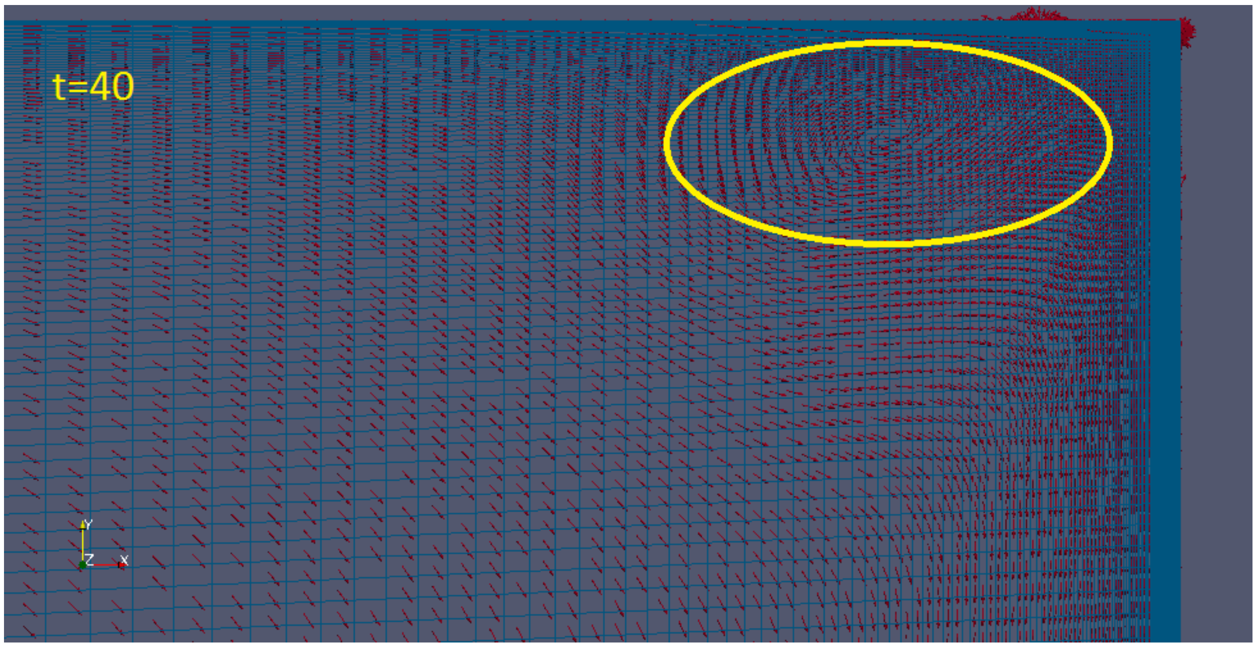

Figure A.4: Time 40

A.2 One-phase circulation at different Reynolds number

Re=50:

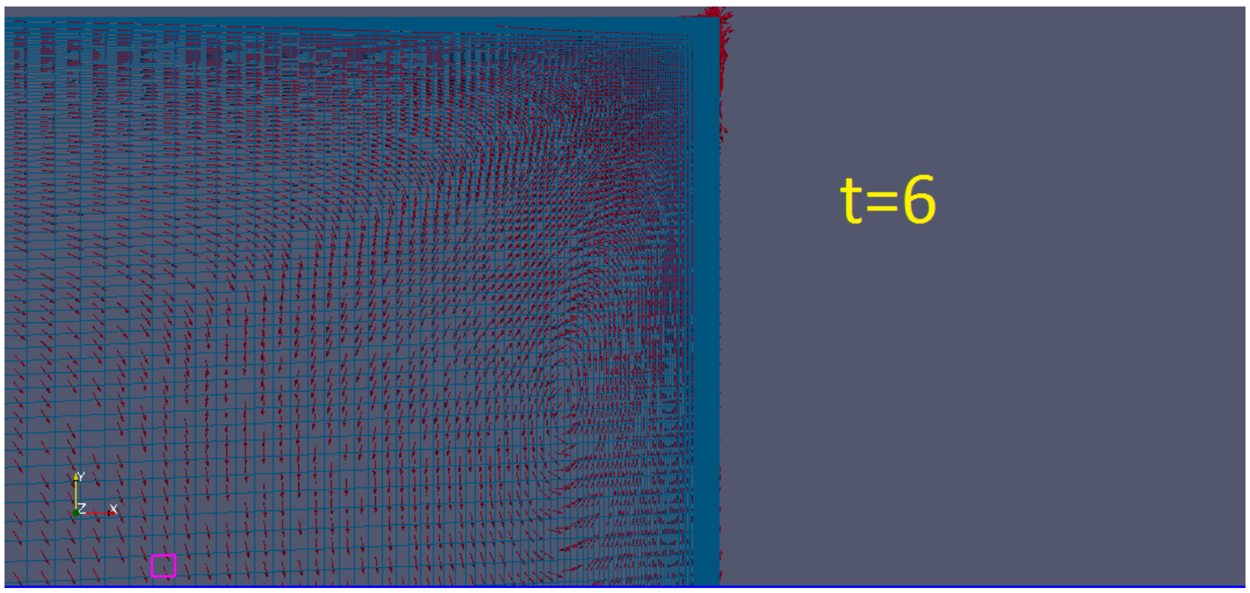

Figure A.5: Re=50 Time 6

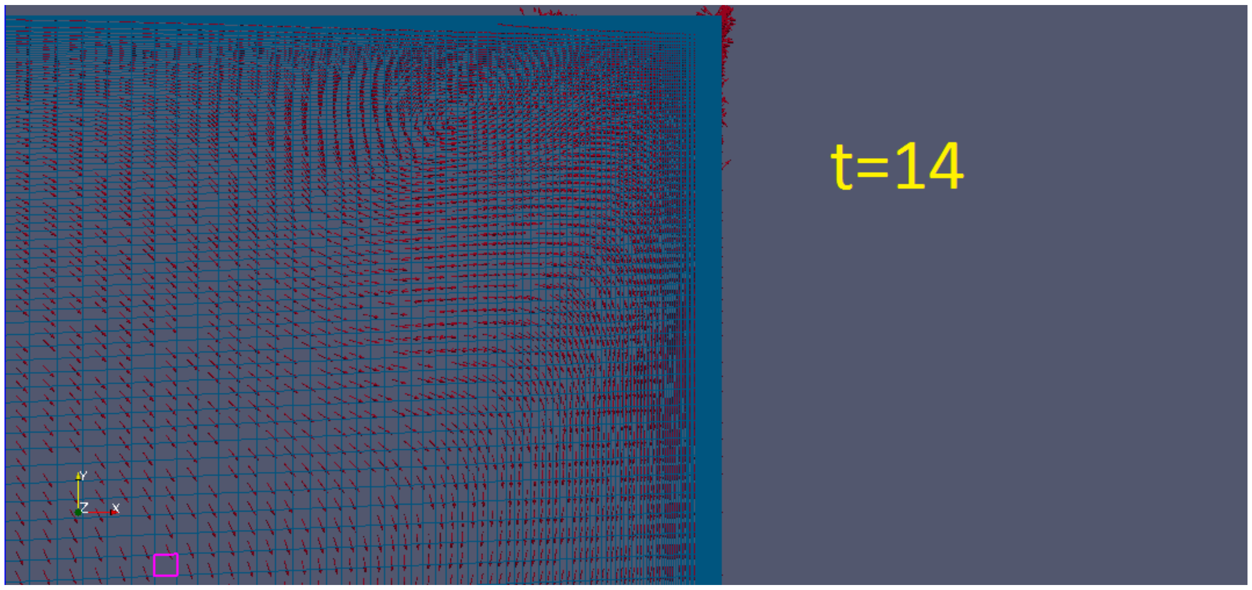

Figure A.6: Re=50 Time 14

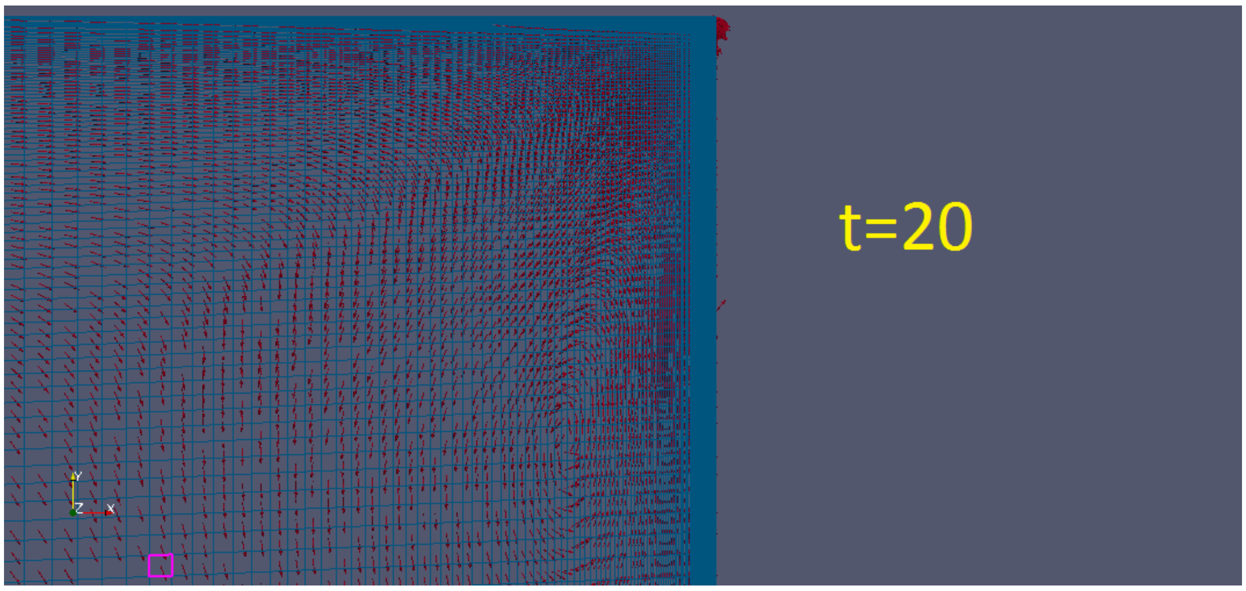

Figure A.7: Re=50 Time 20

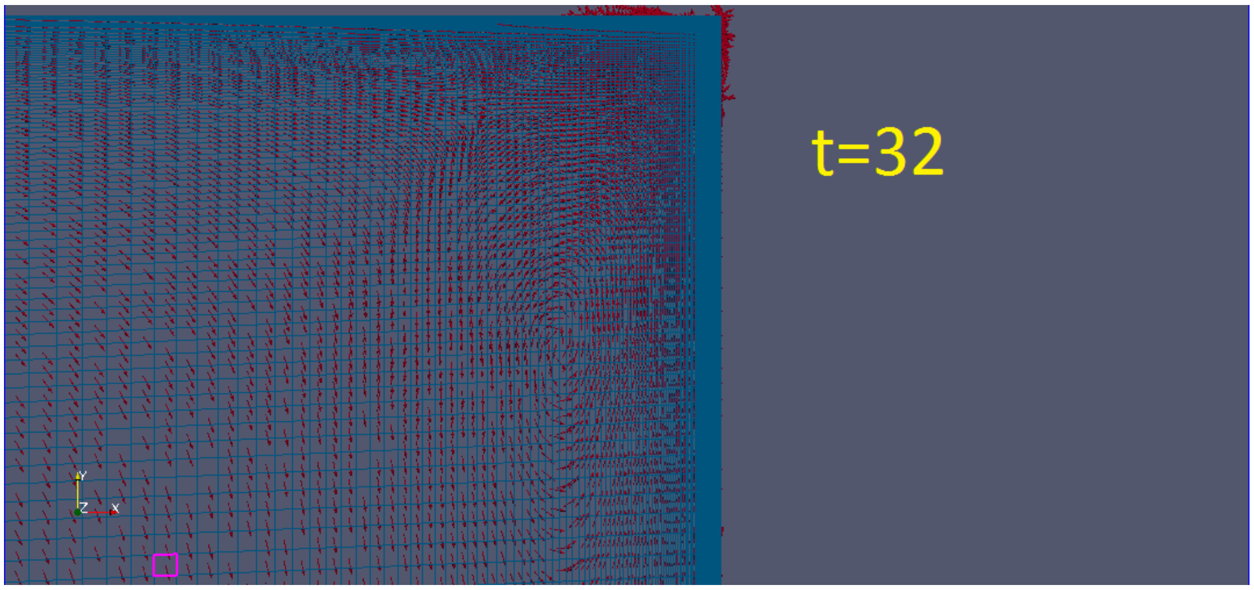

Figure A.8: Re=50 Time 32

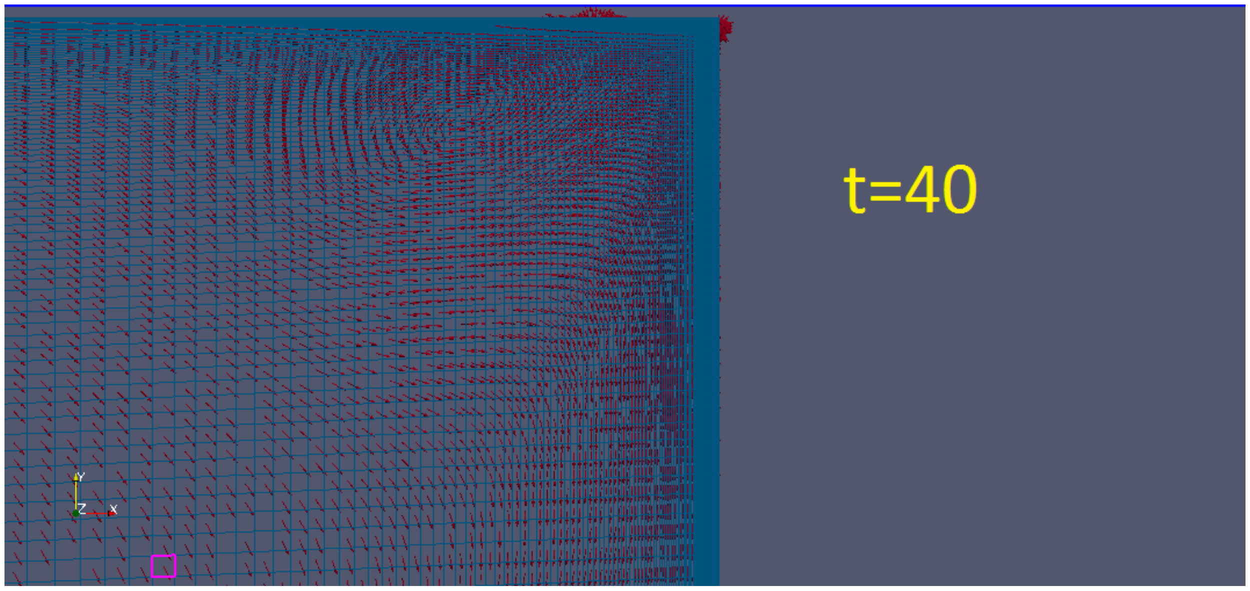

Figure A.9: Re=50 Time 40

Re=100:

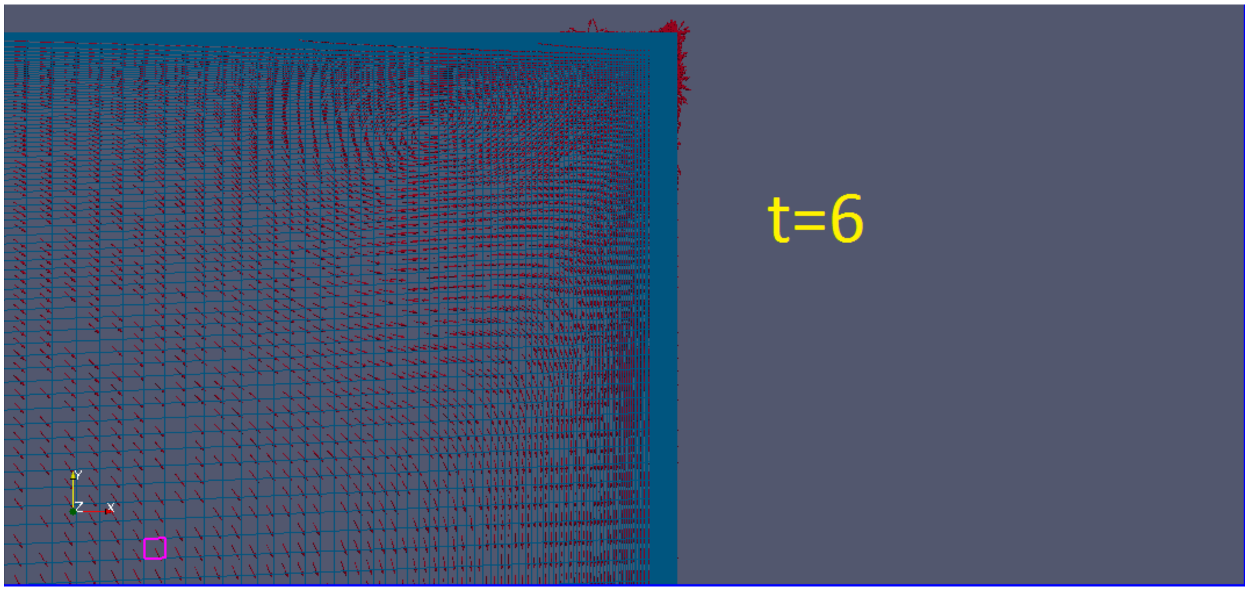

Figure A.10: Re=100 Time 6

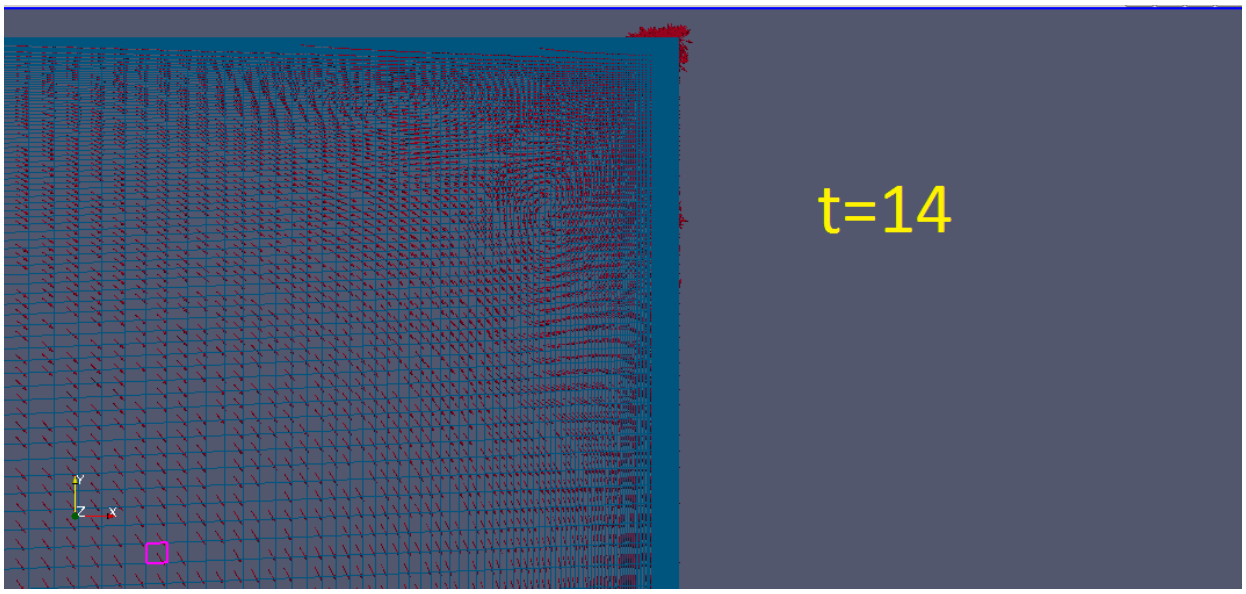

Figure A.11: Re=100 Time 14

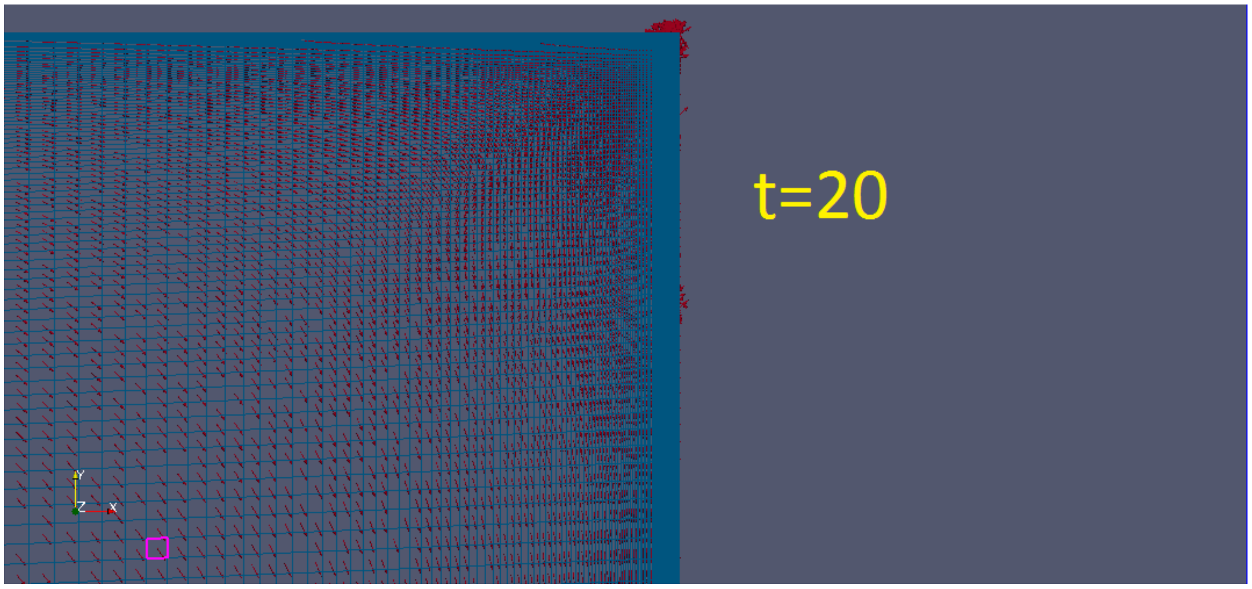

Figure A.12: Re=100 Time 20

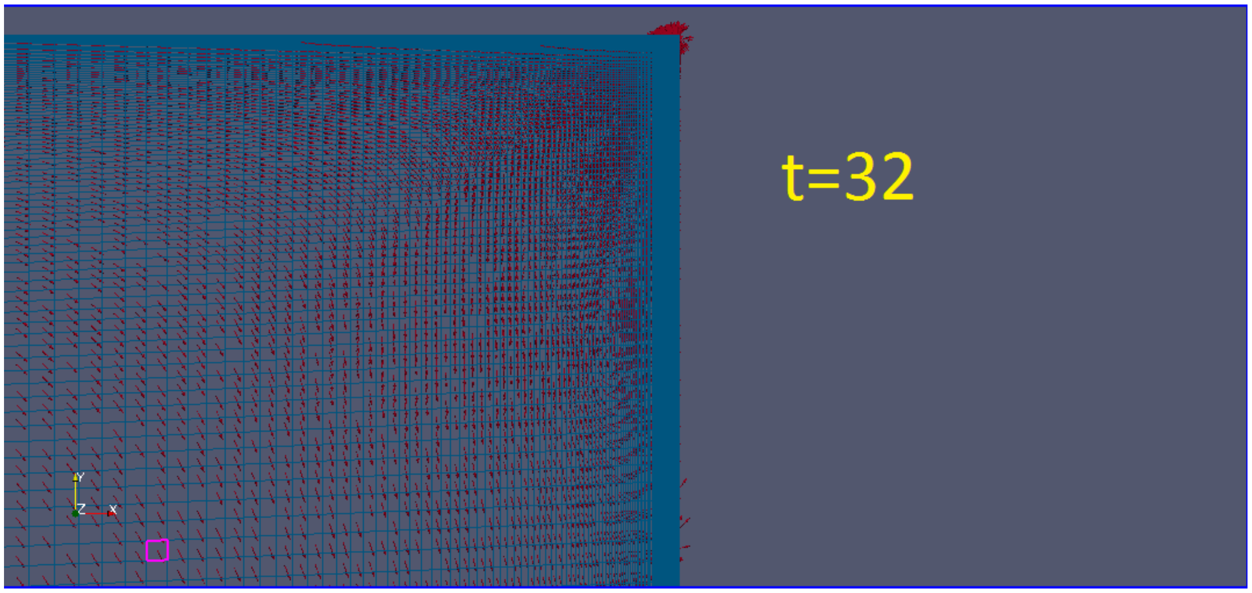

Figure A.13: Re=100 Time 32

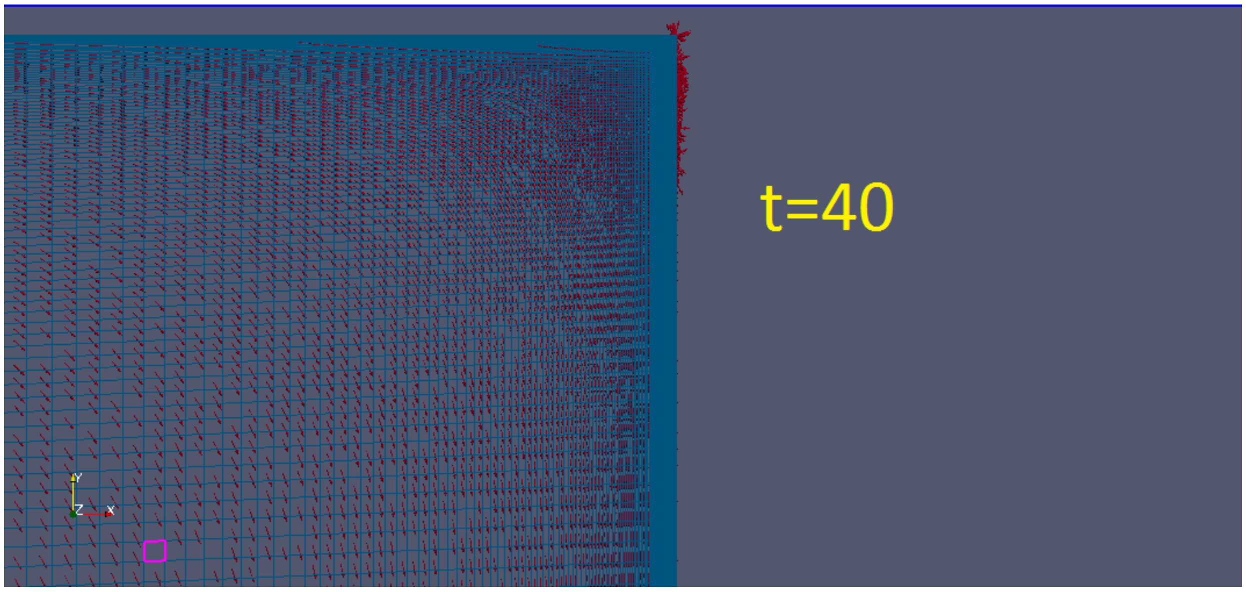

Figure A.14: Re=100 Time 40

Re=150:

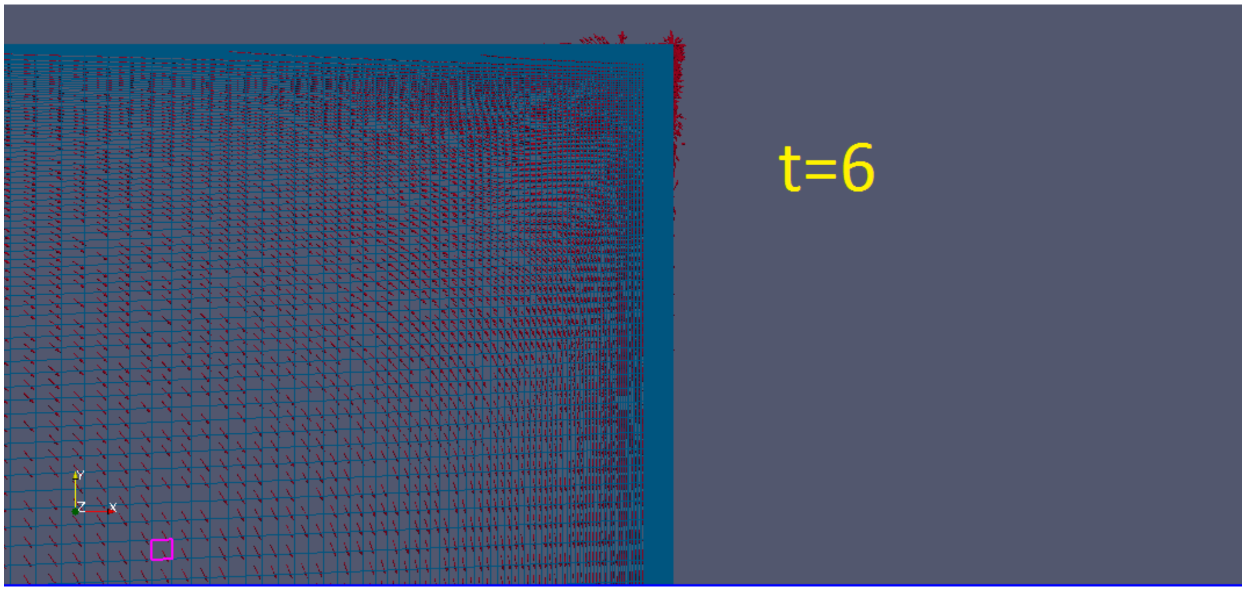

Figure A.15: Re=150 Time 6

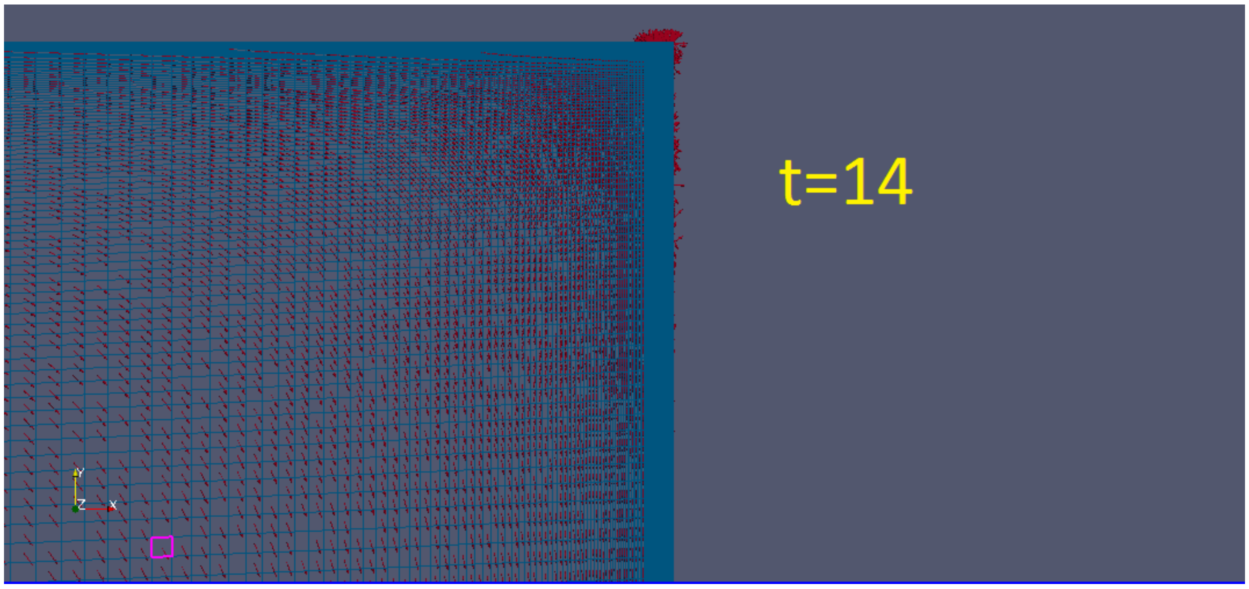

Figure A.16: Re=150 Time 14

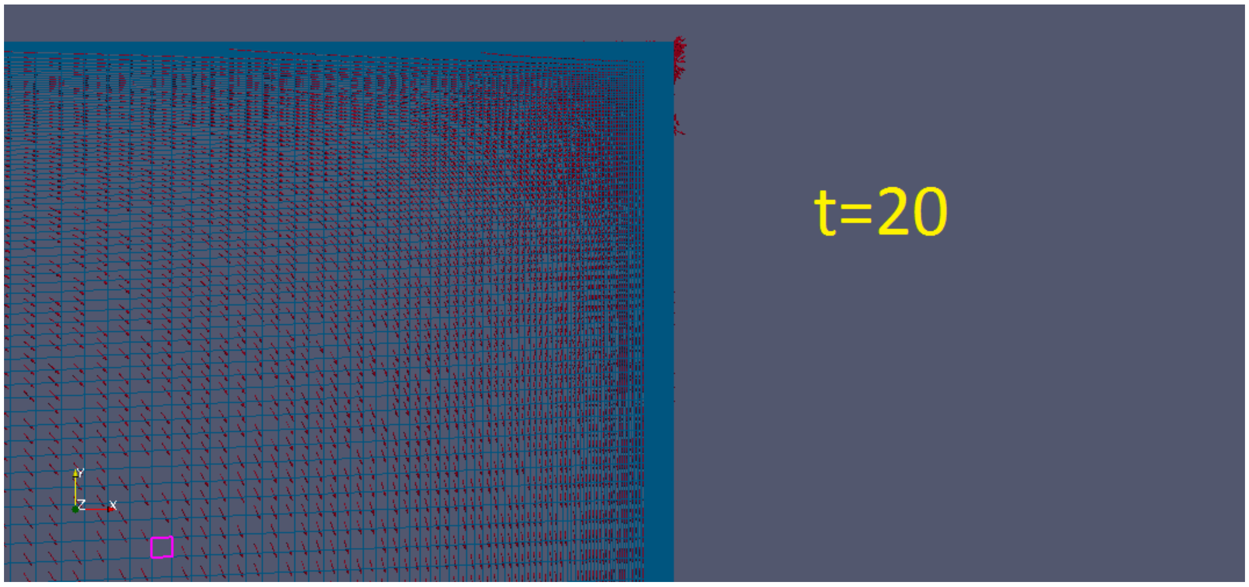

Figure A.17: Re=150 Time 20

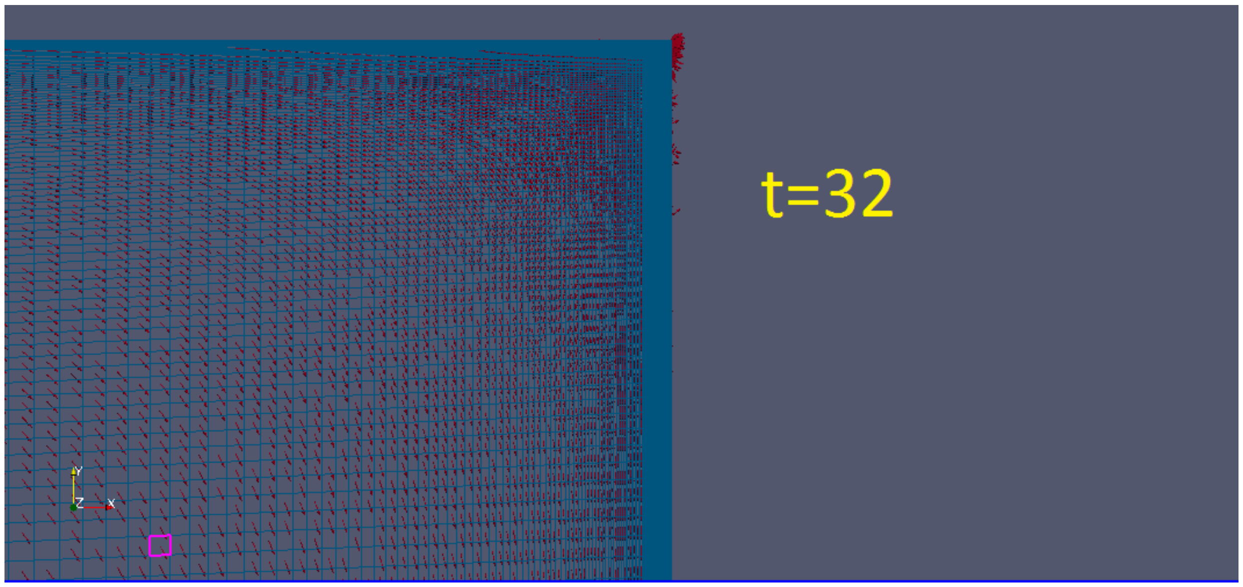

Figure A.18: Re=150 Time 32

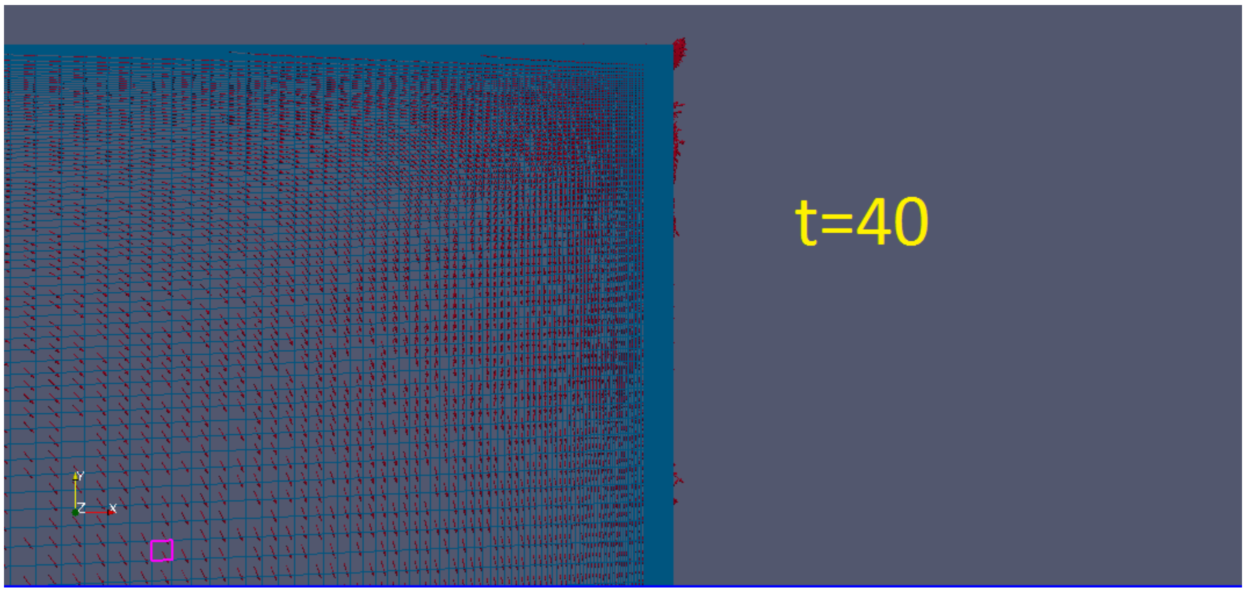

Figure A.19: Re=150 Time 40

A.3 Close-up on interruption region

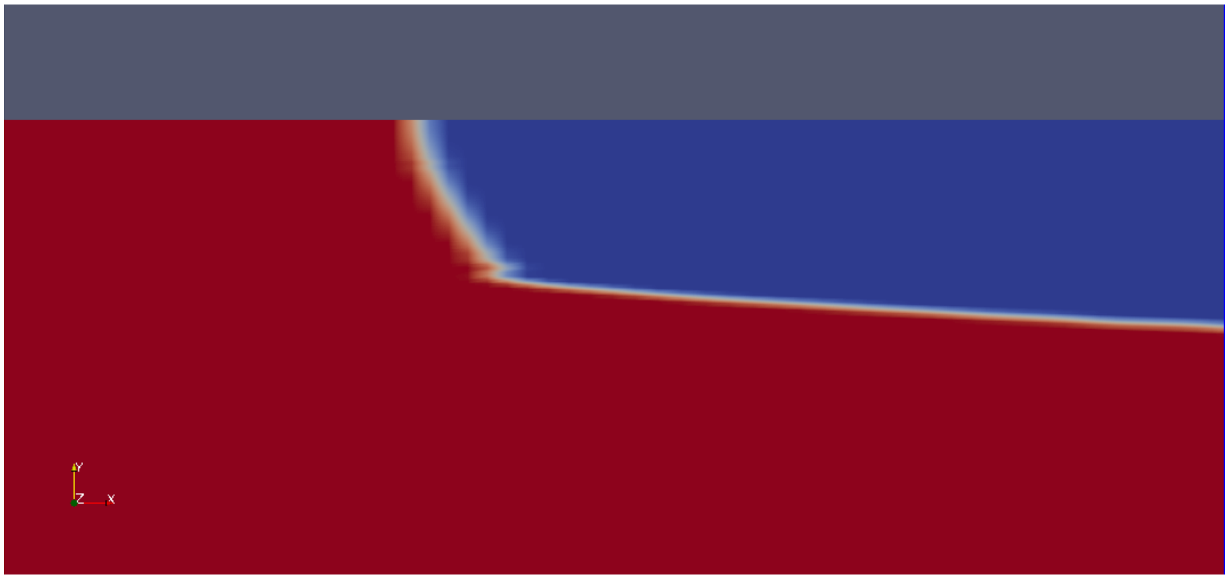

Figure A.20: Interruption

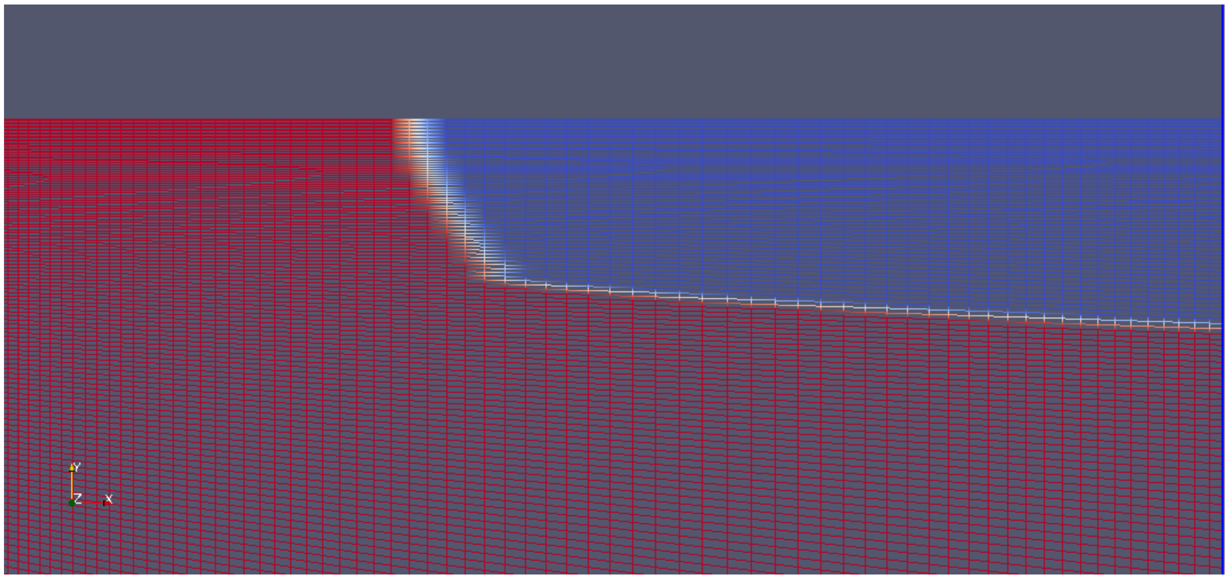

Figure A.21: Interruption, with mesh

A.4 Channel flow, structured grid

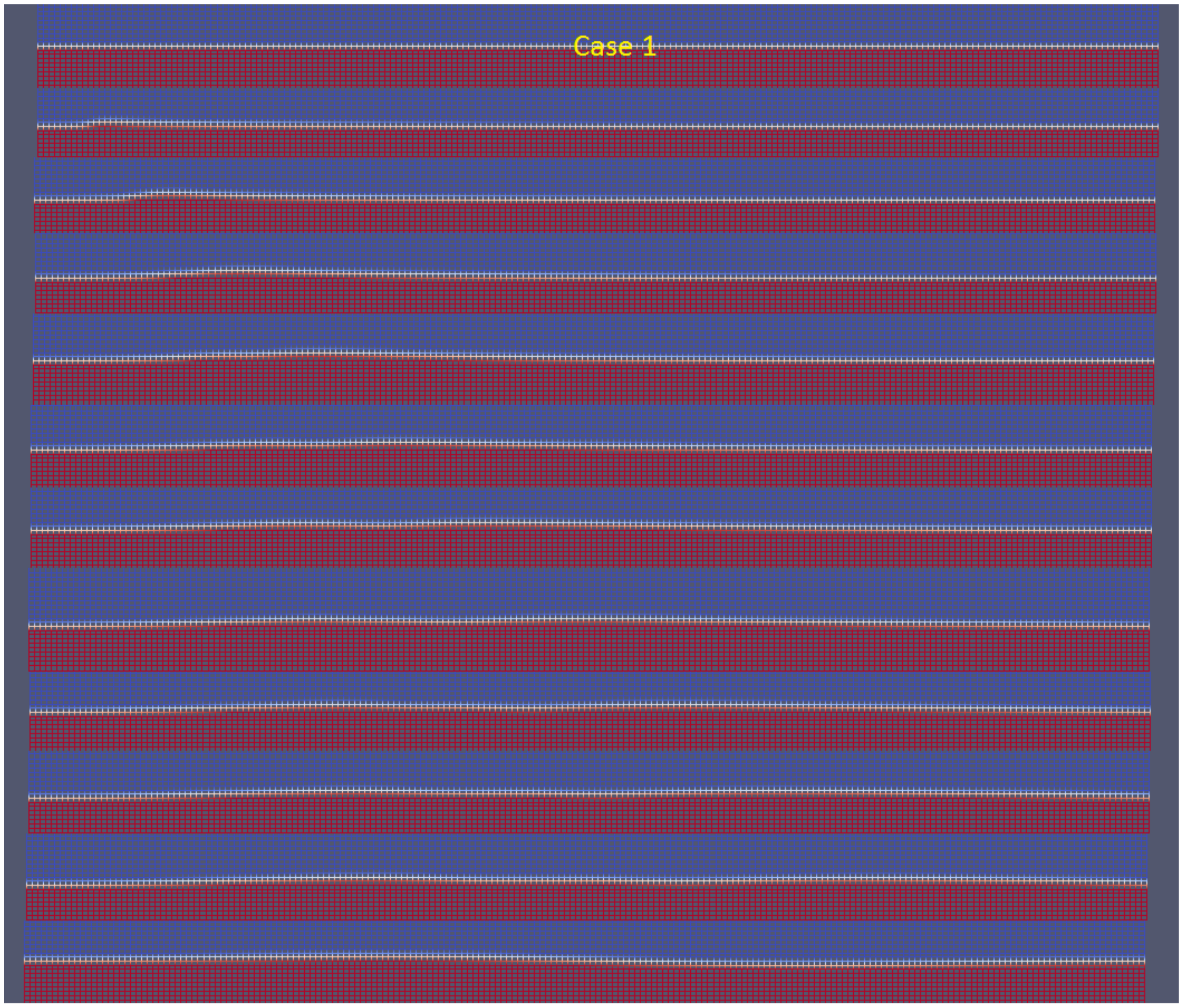

Figure A.22: Case 1

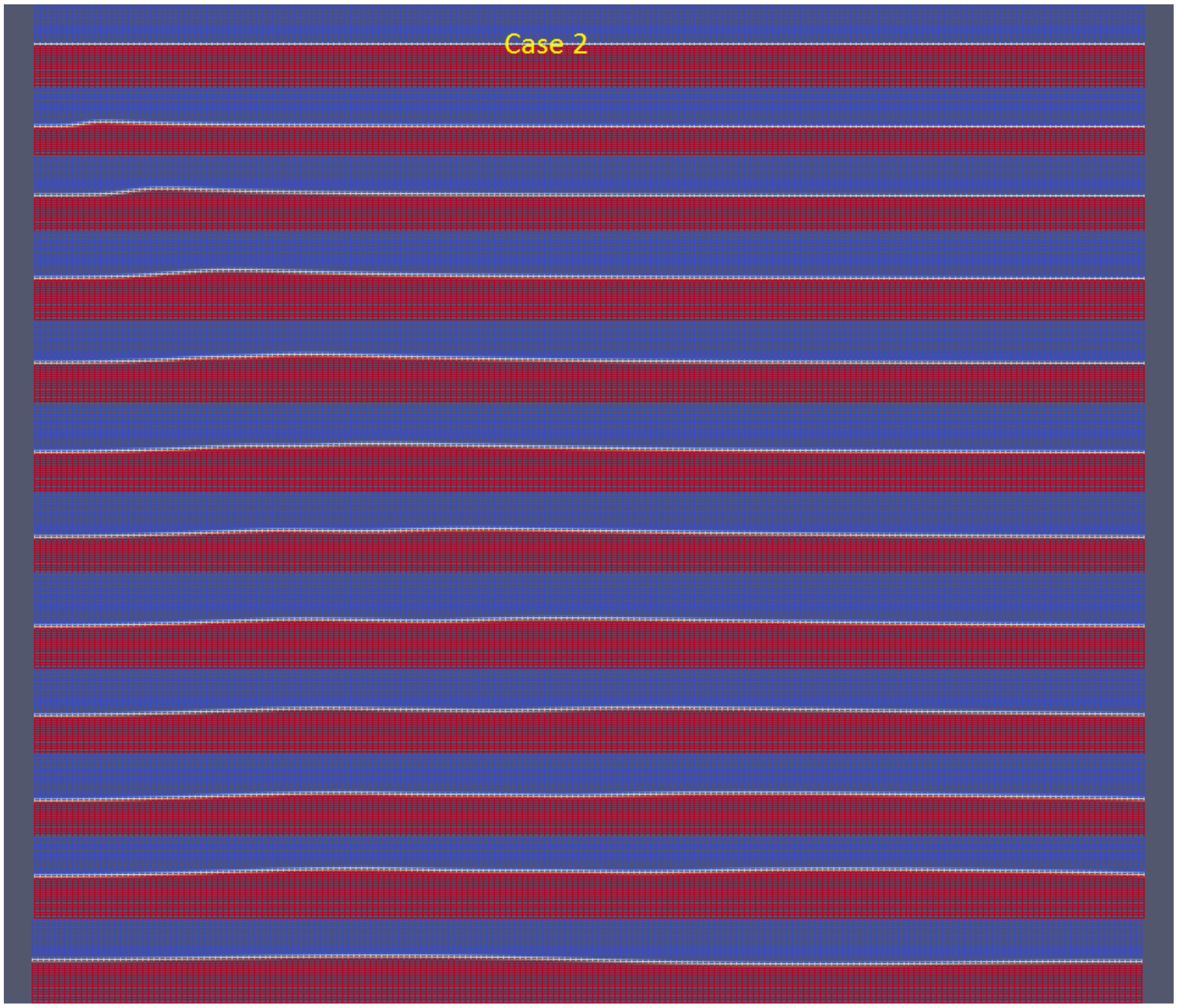

Figure A.23: Case 2

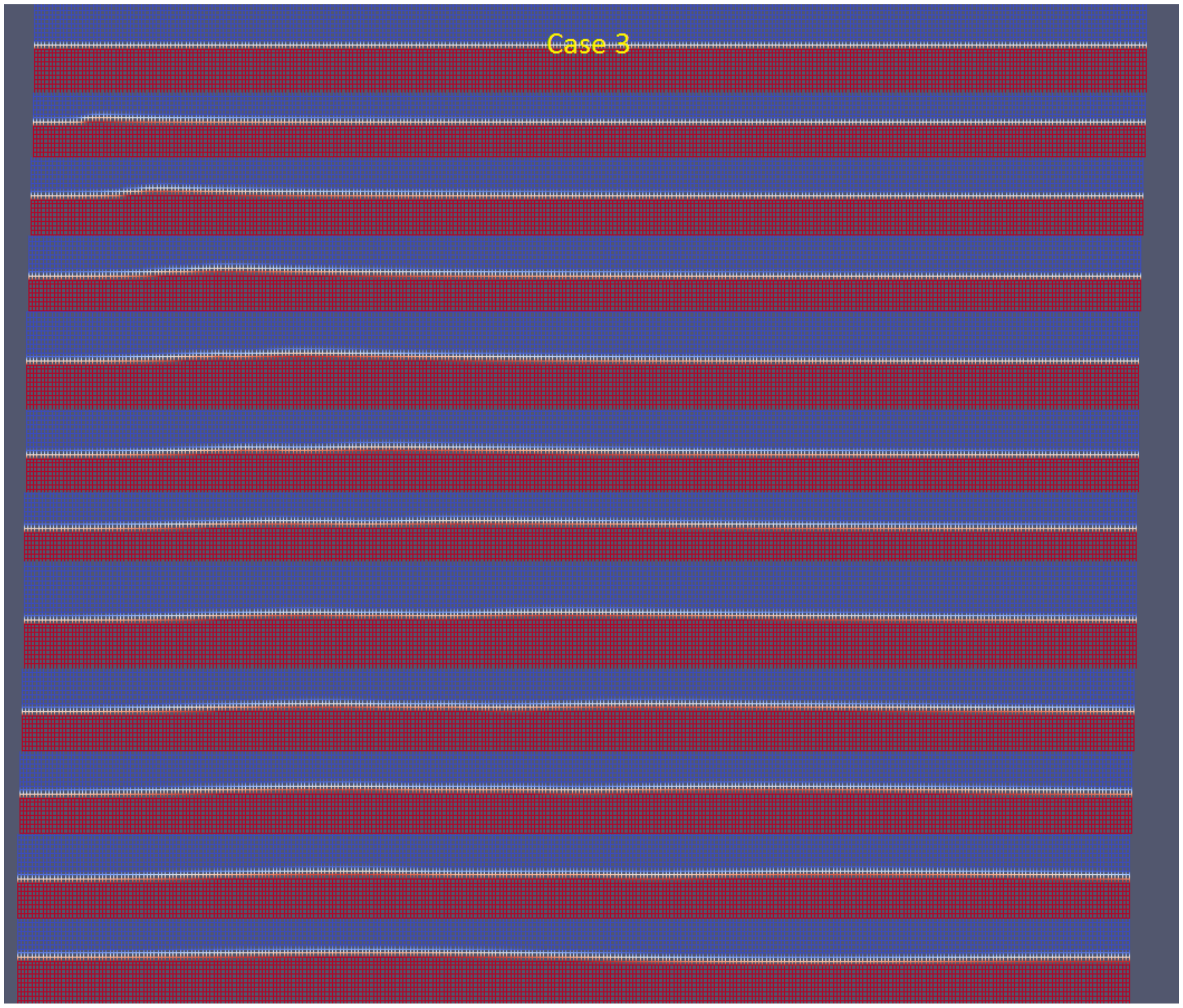

Figure A.24: Case 3

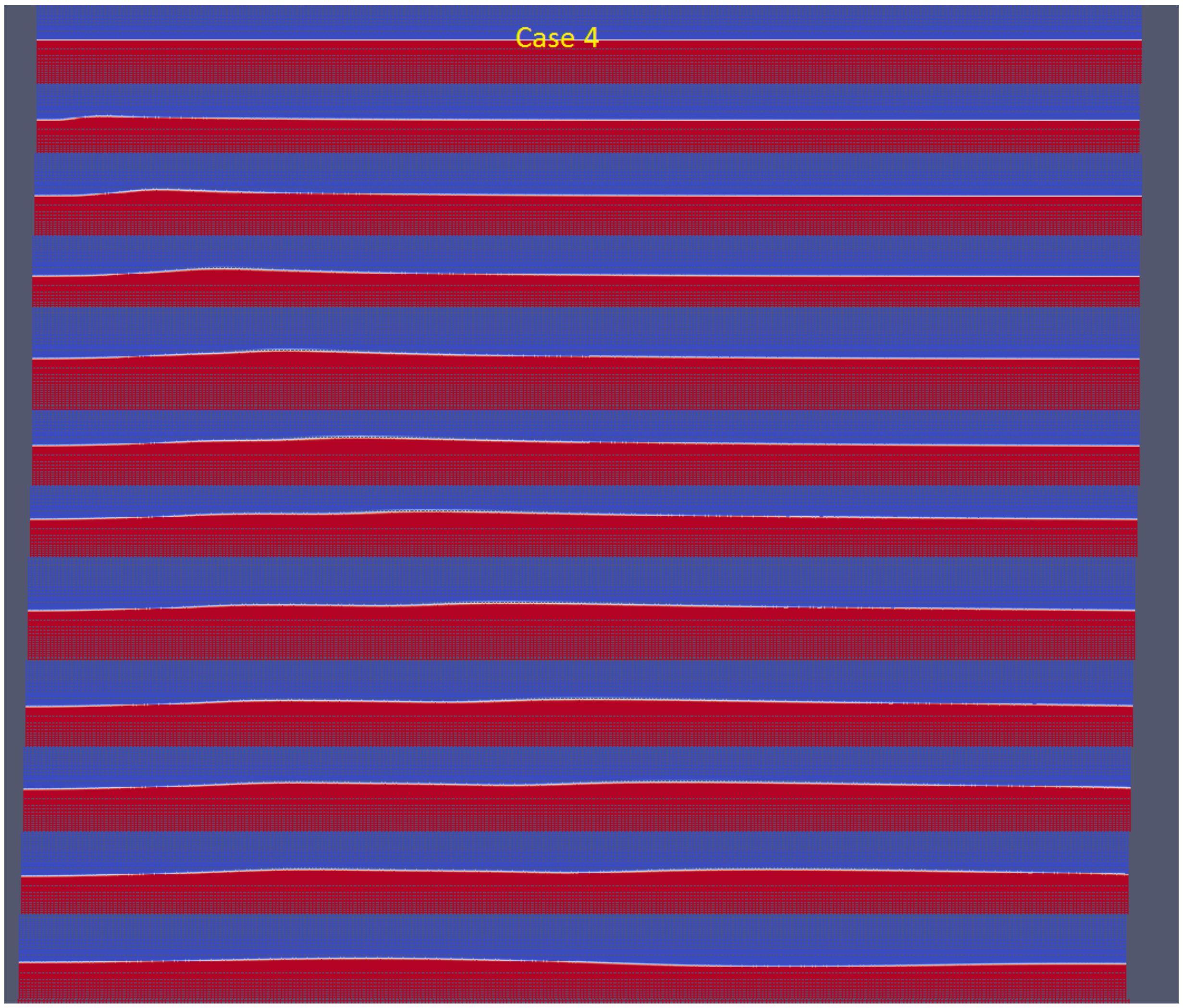

Figure A.25: Case 4

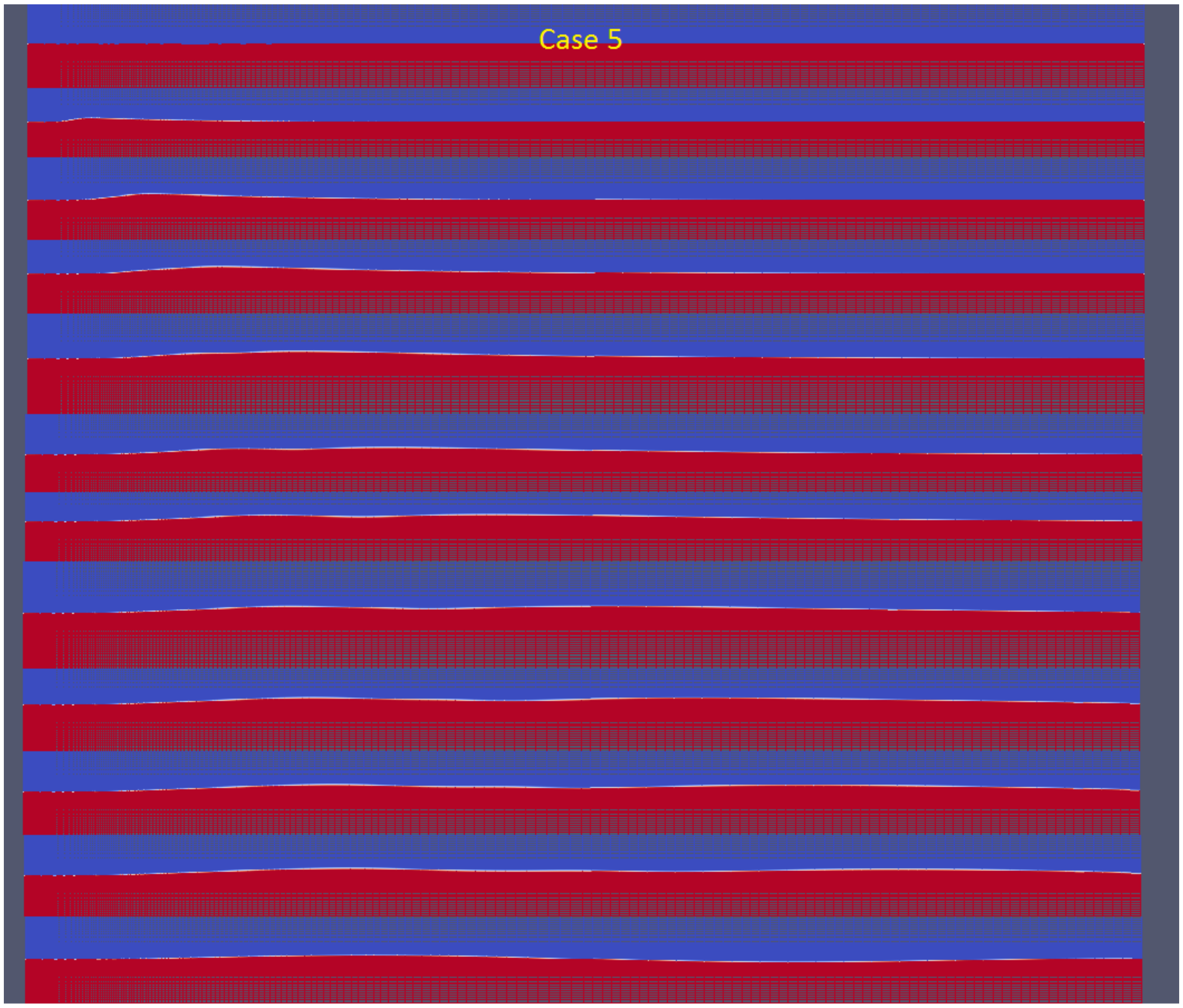

Figure A.26: Case 5

A.5 Evolving nummerical diffusion

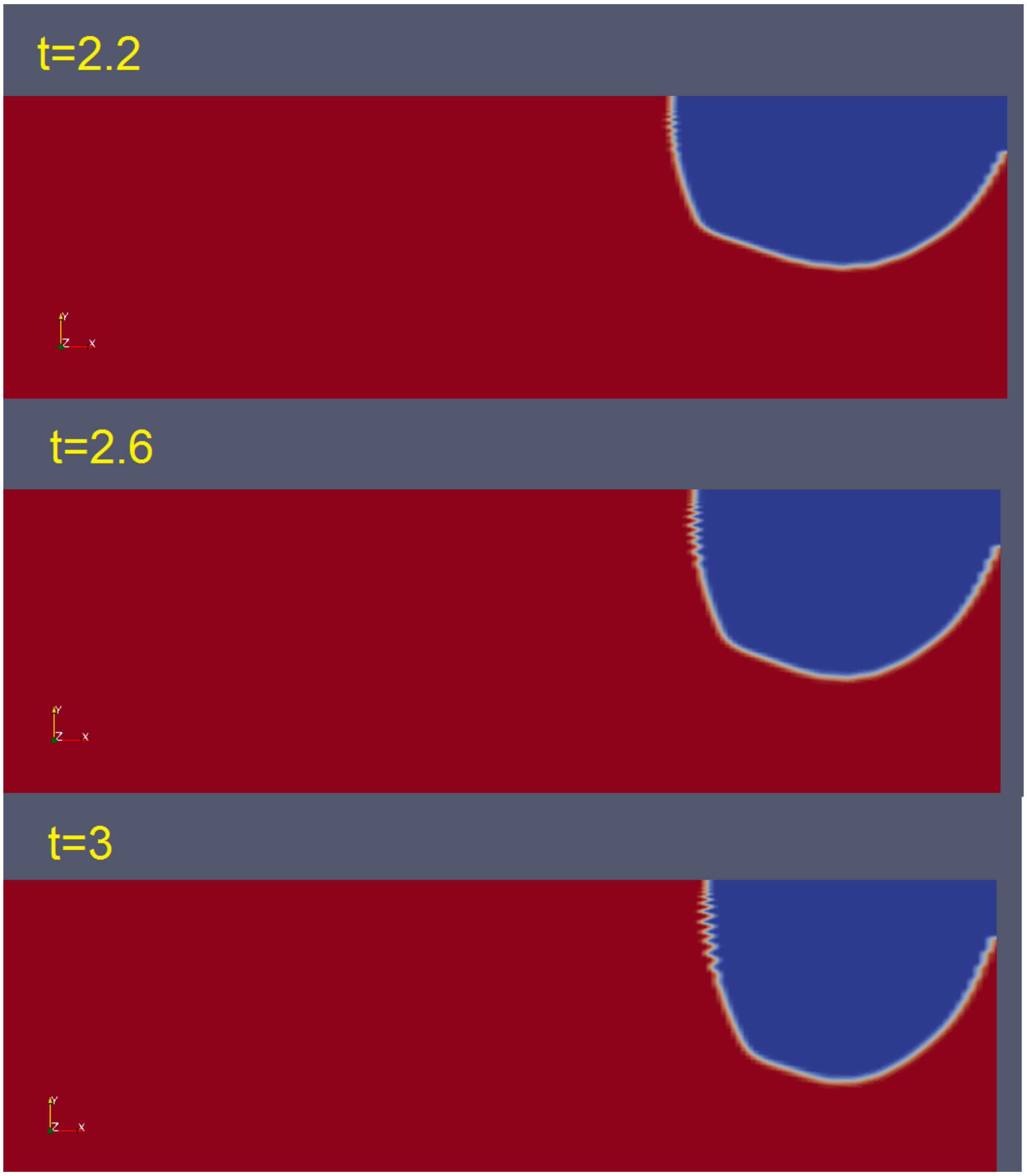

Figure A.27: Case 8 evolving numerical diffusion

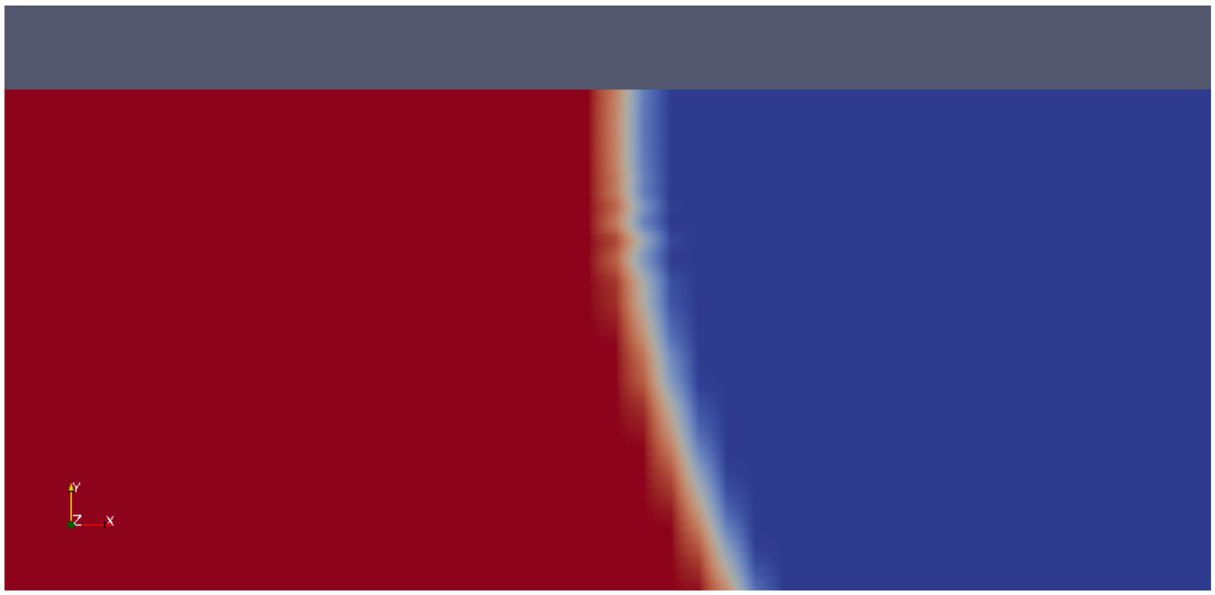

Figure A.28: Case 1 zoomed in at t=1.8

A.6 First time-step of diffusion at different MULES and fvSchemes

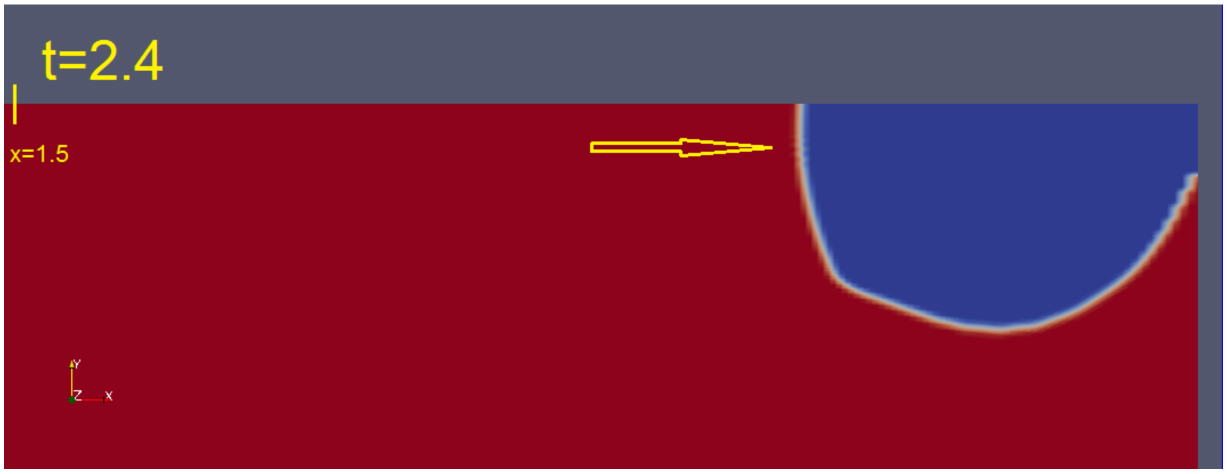

Figure A.29: Case 2 first time-step of diffusion

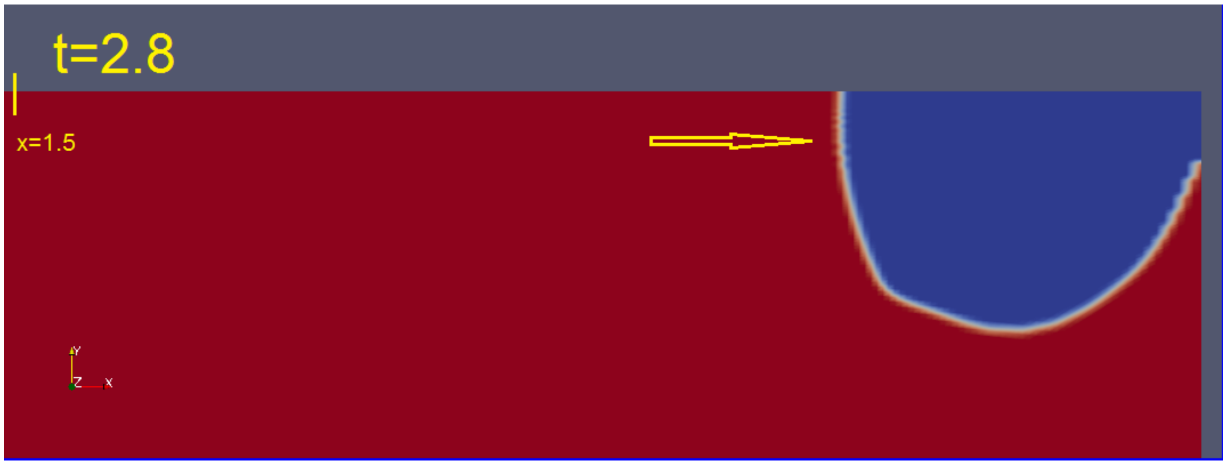

Figure A.30: Case 3 first time-step of diffusion

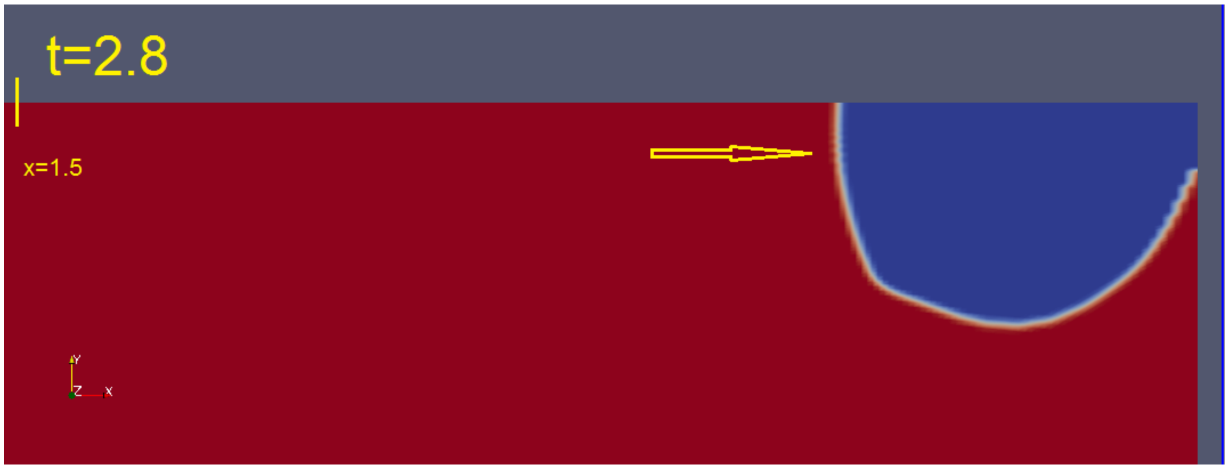

Figure A.31: Case 4 first time-step of diffusion

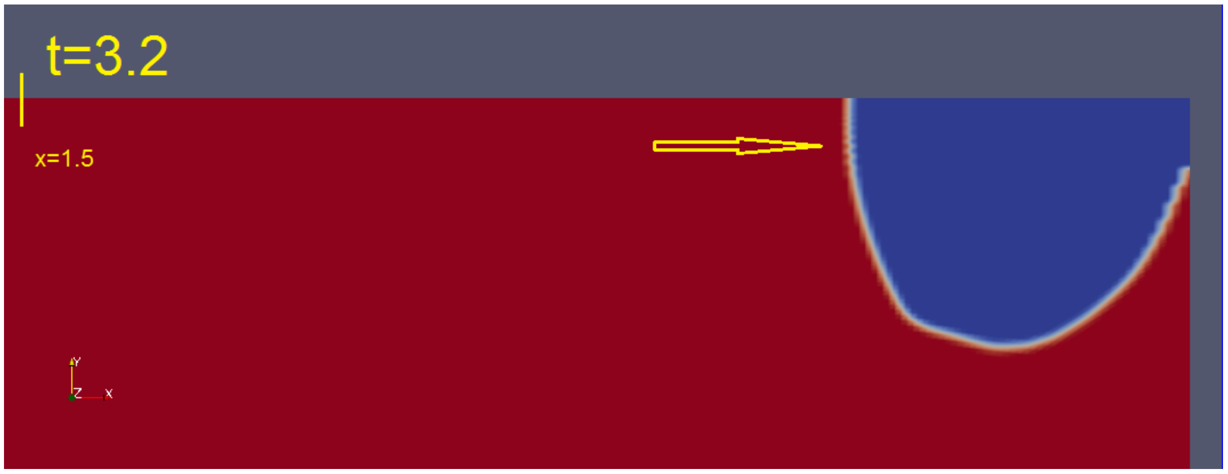

Figure A.32: Case 5 first time-step of diffusion

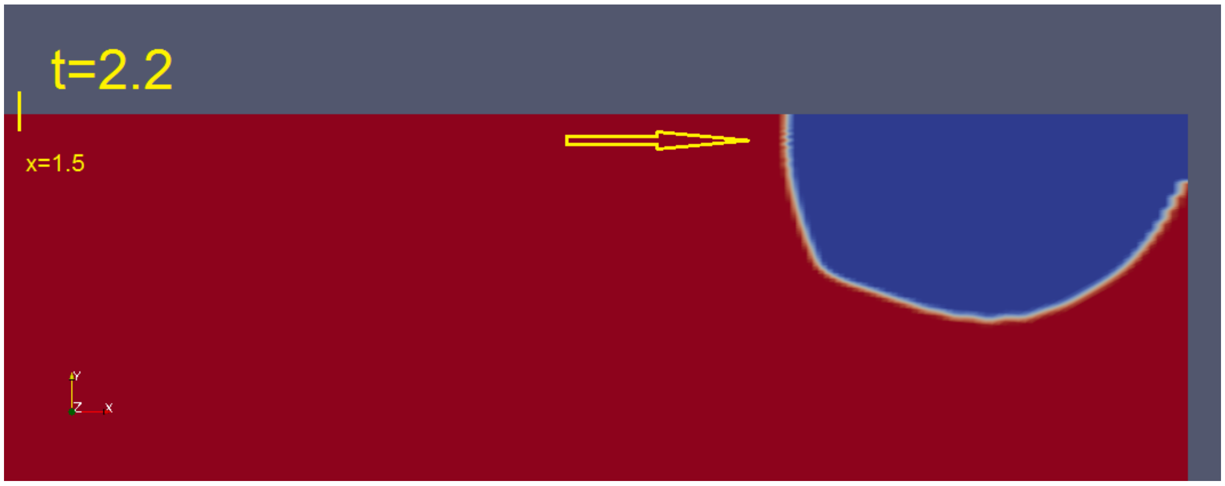

Figure A.33: Case 6 first time-step of diffusion

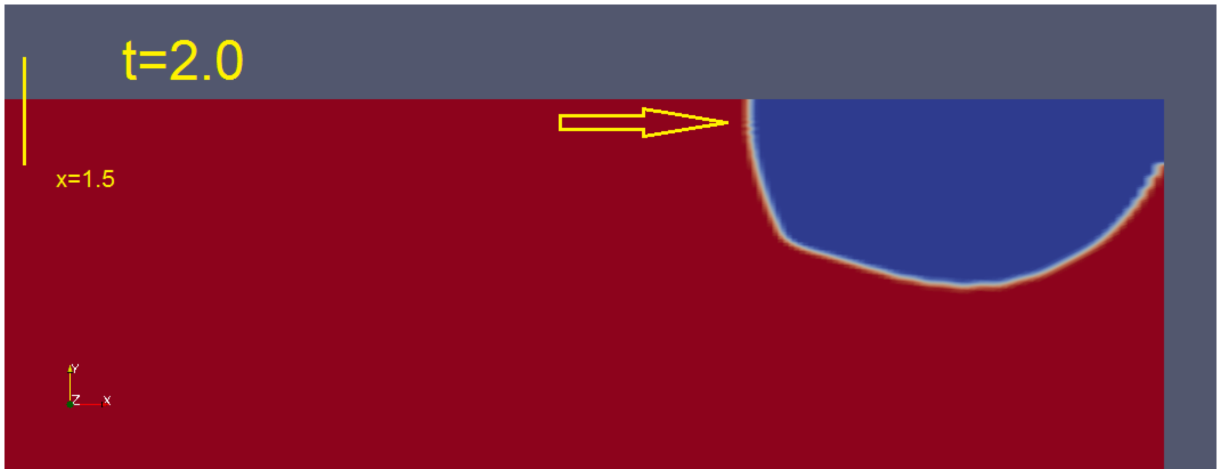

Figure A.34: Case 7 first time-step of diffusion

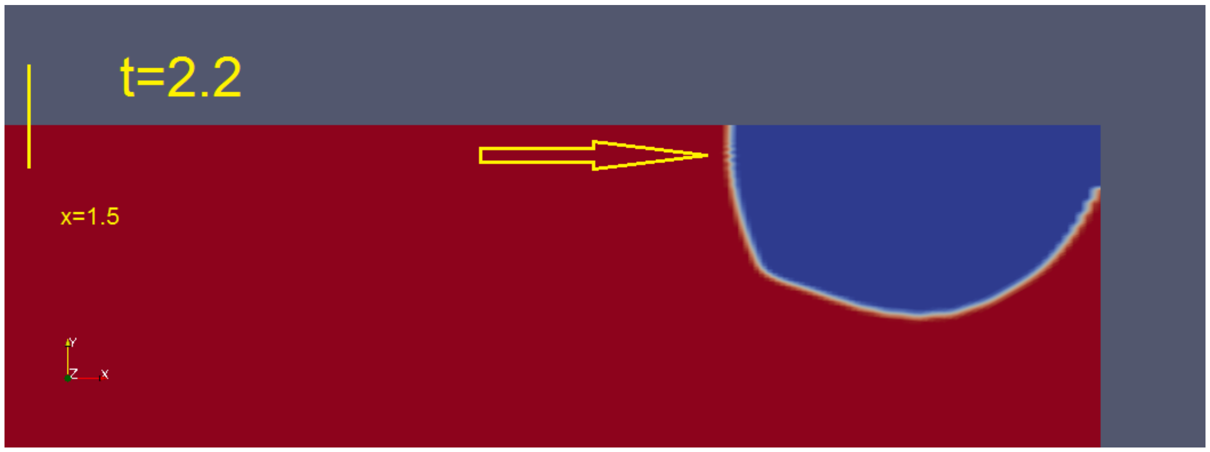

Figure A.35: Case 8 first time-step of diffusion

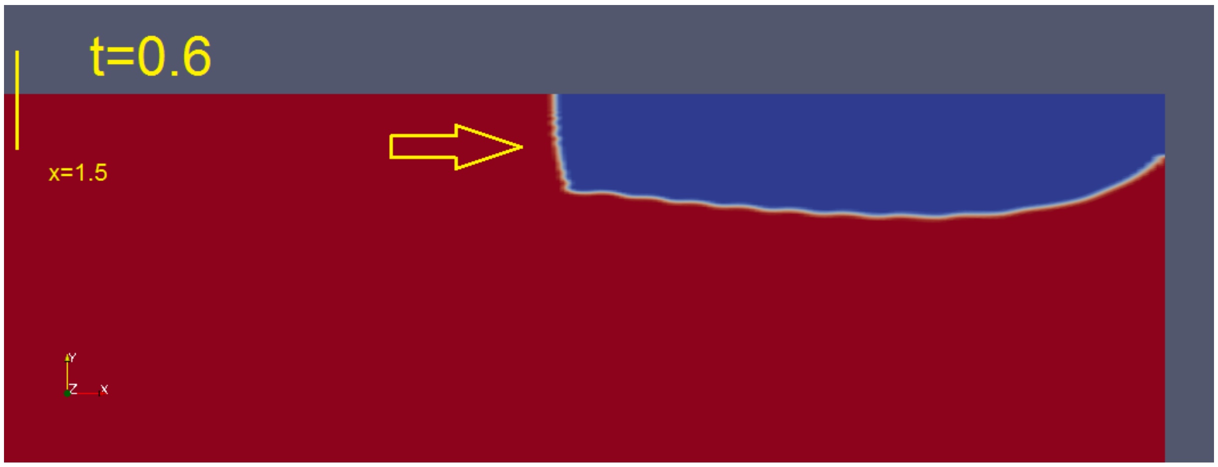

Figure A.36: Case 1 first time-step of diffusion, with PIMPLE instead of PISO

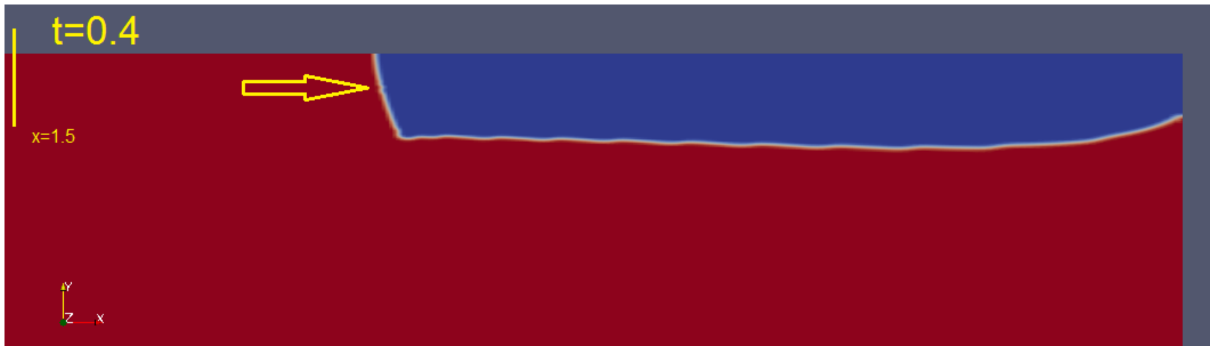

Figure A.37: Case 1 first time-step of diffusion, with PISO and nCorrectors=10

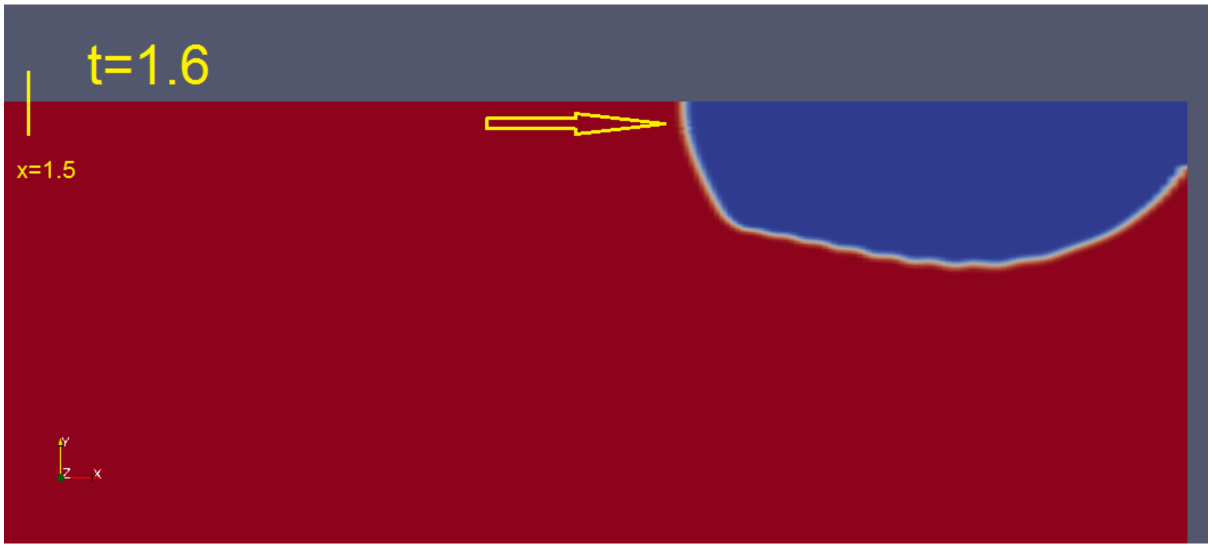

Figure A.38: Case 1 first time-step of diffusion, with Gauss interfaceCompression

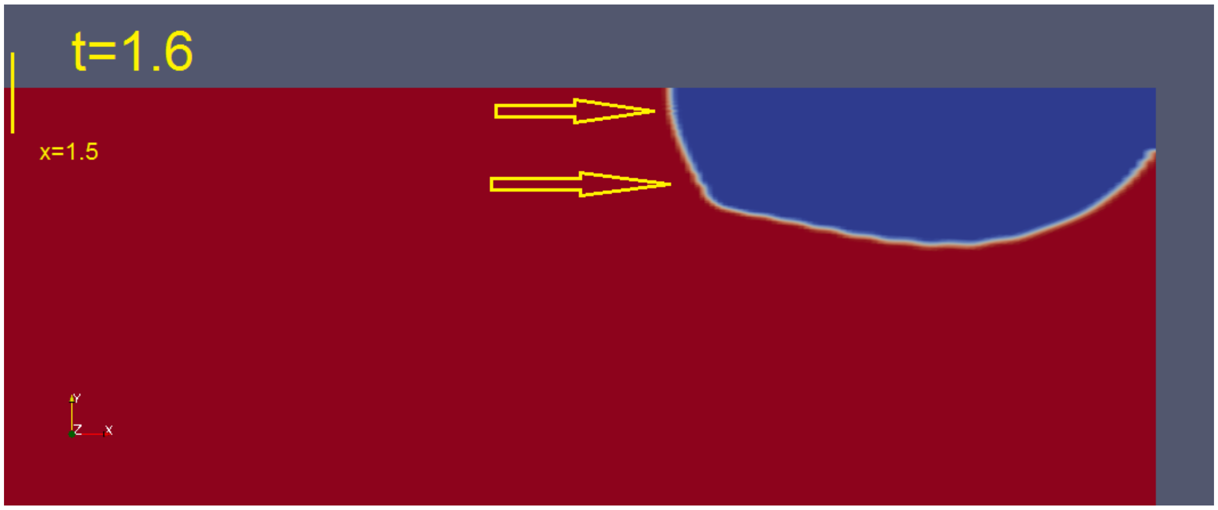

Figure A.39: Case 1 first time-step of diffusion, with Gauss limitedLinear 1

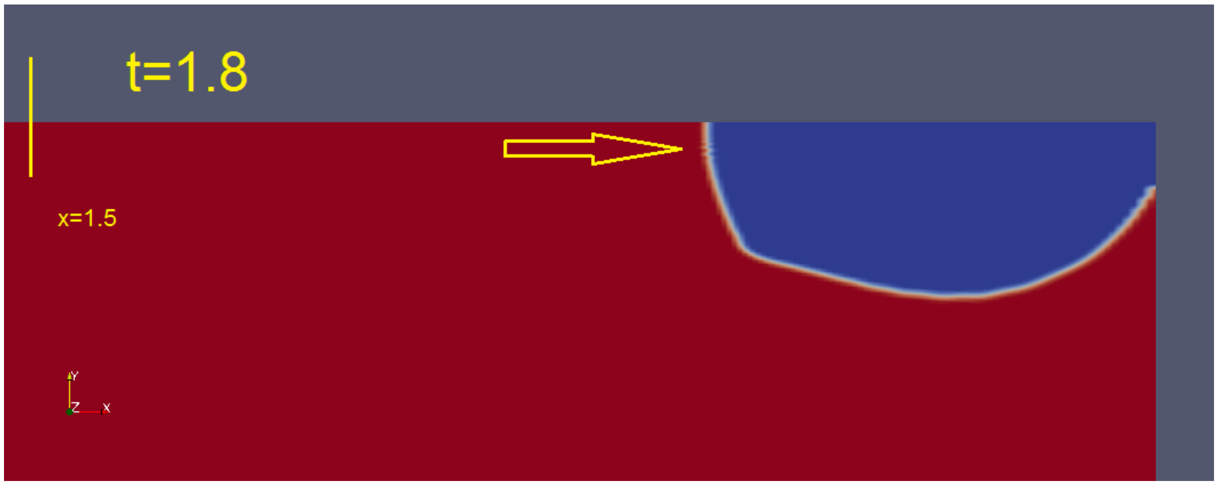

Figure A.40: Case 1 first time-step of diffusion, with Minmod and SFCDV

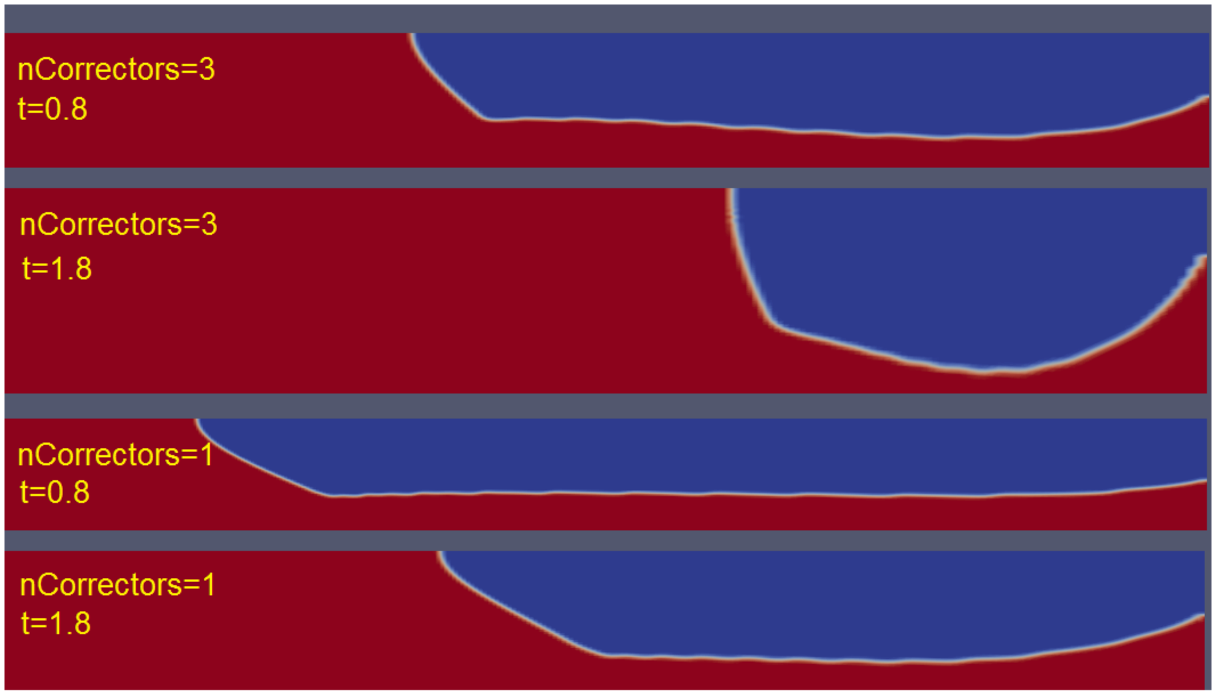

Figure A.41: Comparison of same case with PISO nCorrectors=1 and nCorrectors=3

A.7 Additional cases for alternating density

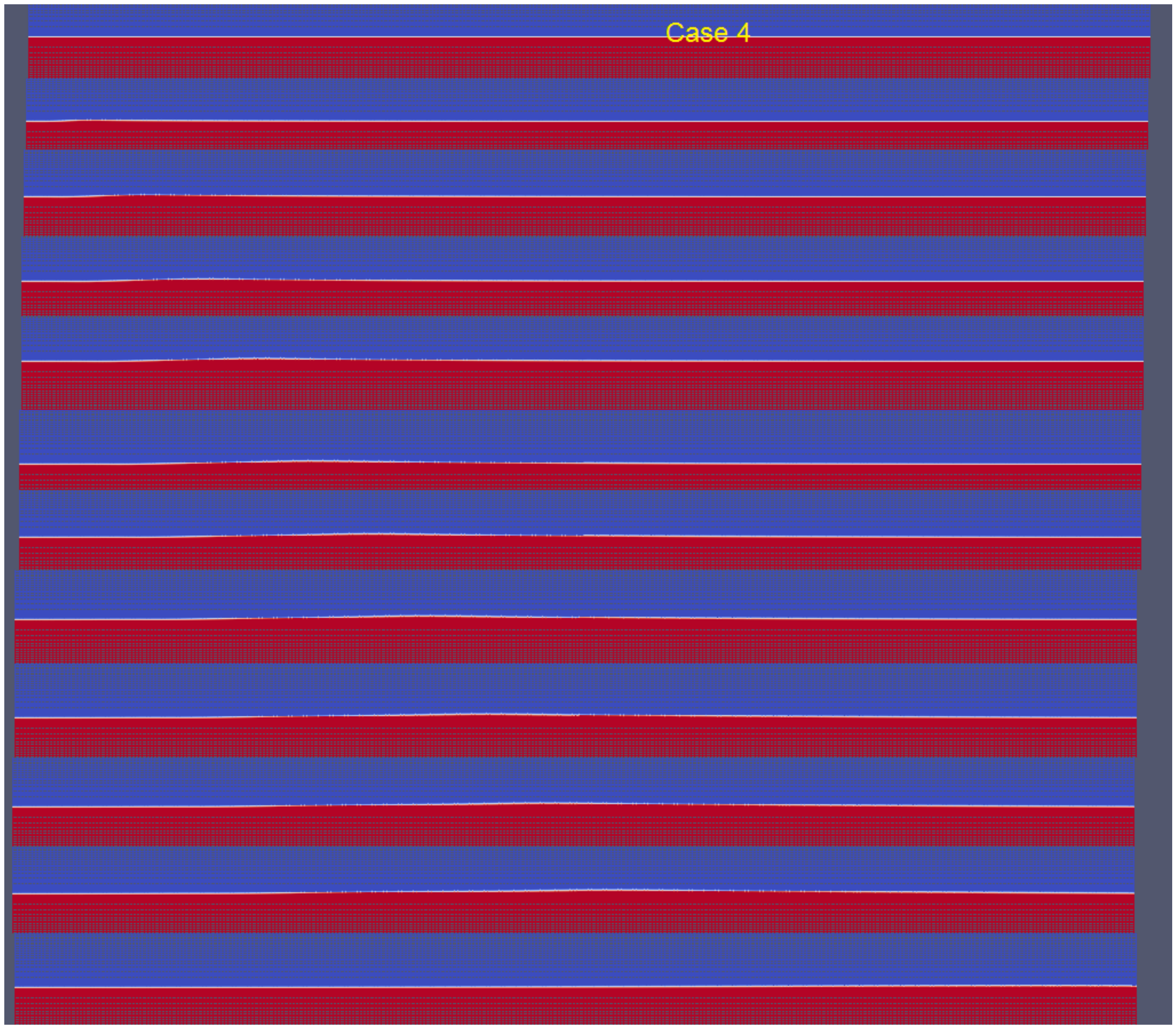

Figure A.42: Flow pattern for Case 4

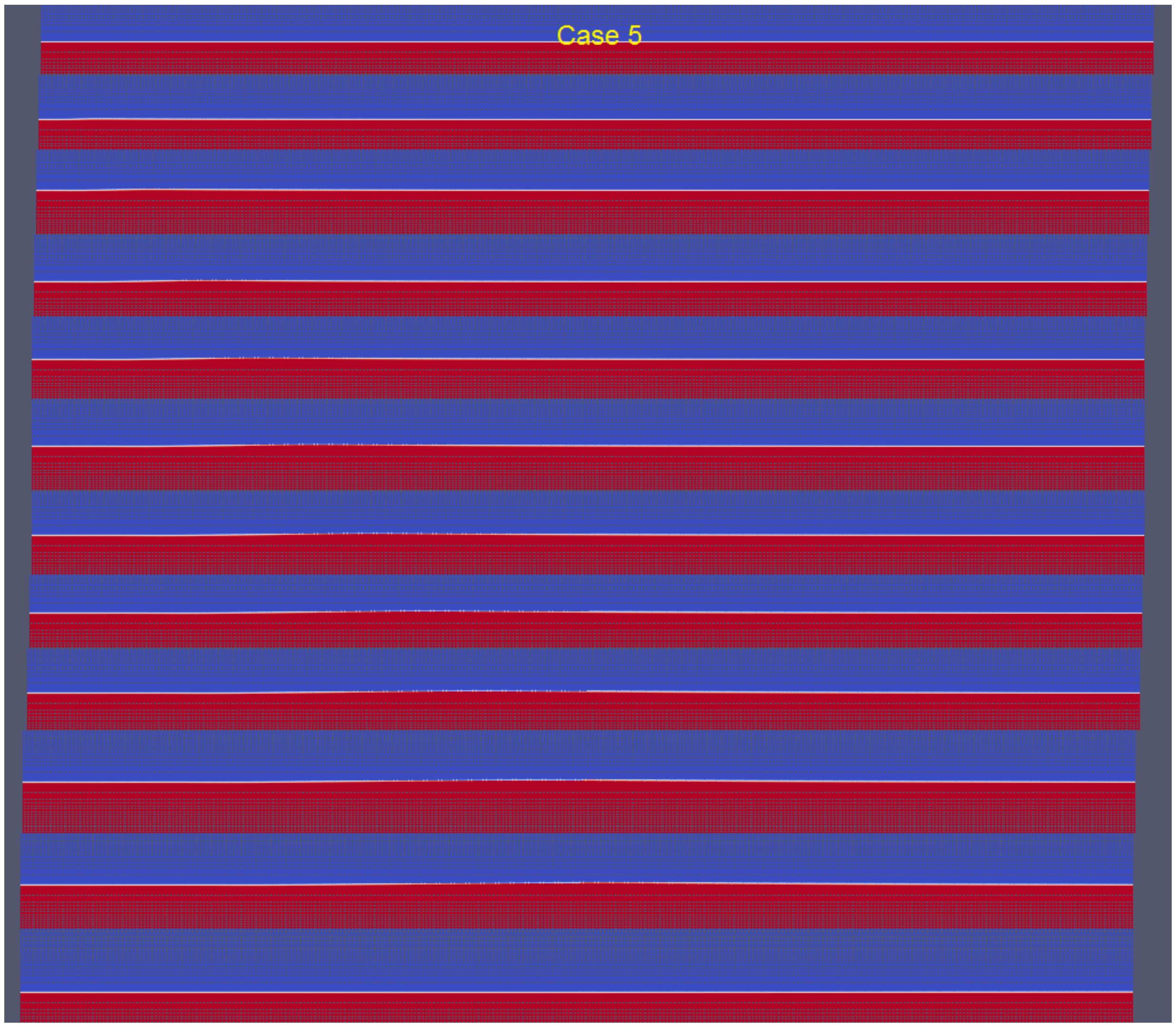

Figure A.43: Flow pattern for Case 2

A.8 Skewness fail

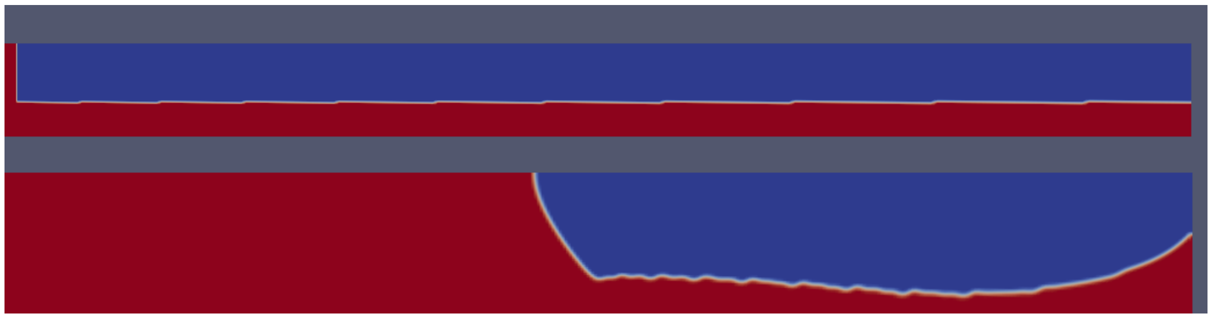

Figure A.44: Skewness imposed by increments impose failure in interface

A.9 Structured grid zoomed

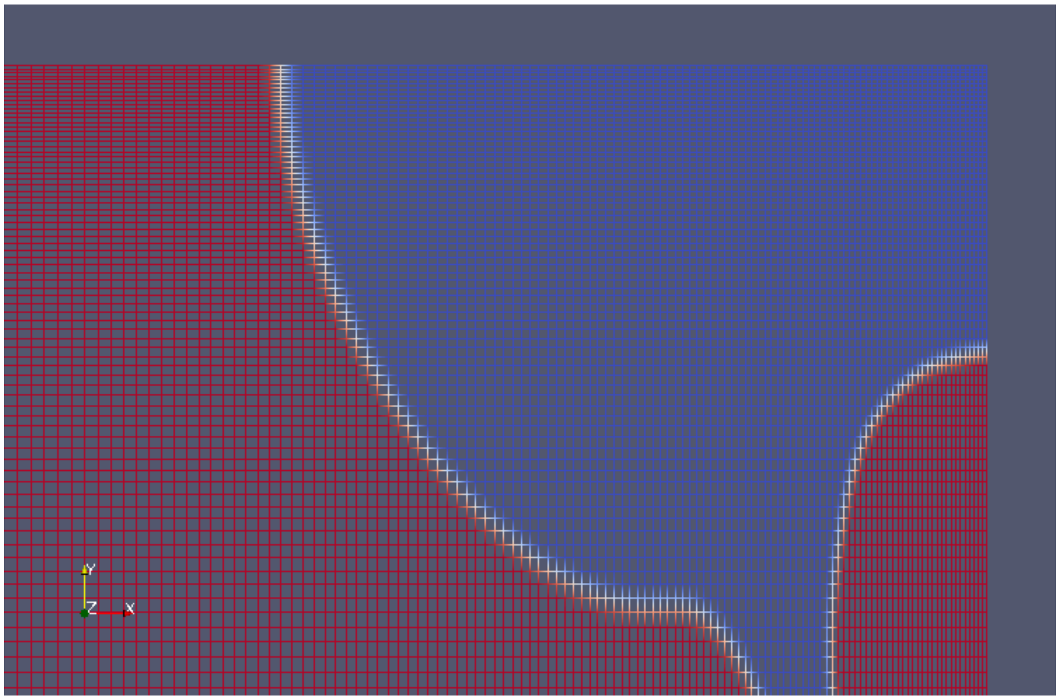

Figure A.45: t=20, structured grid, zoomed in

A.10 Flow pattern with Re=100

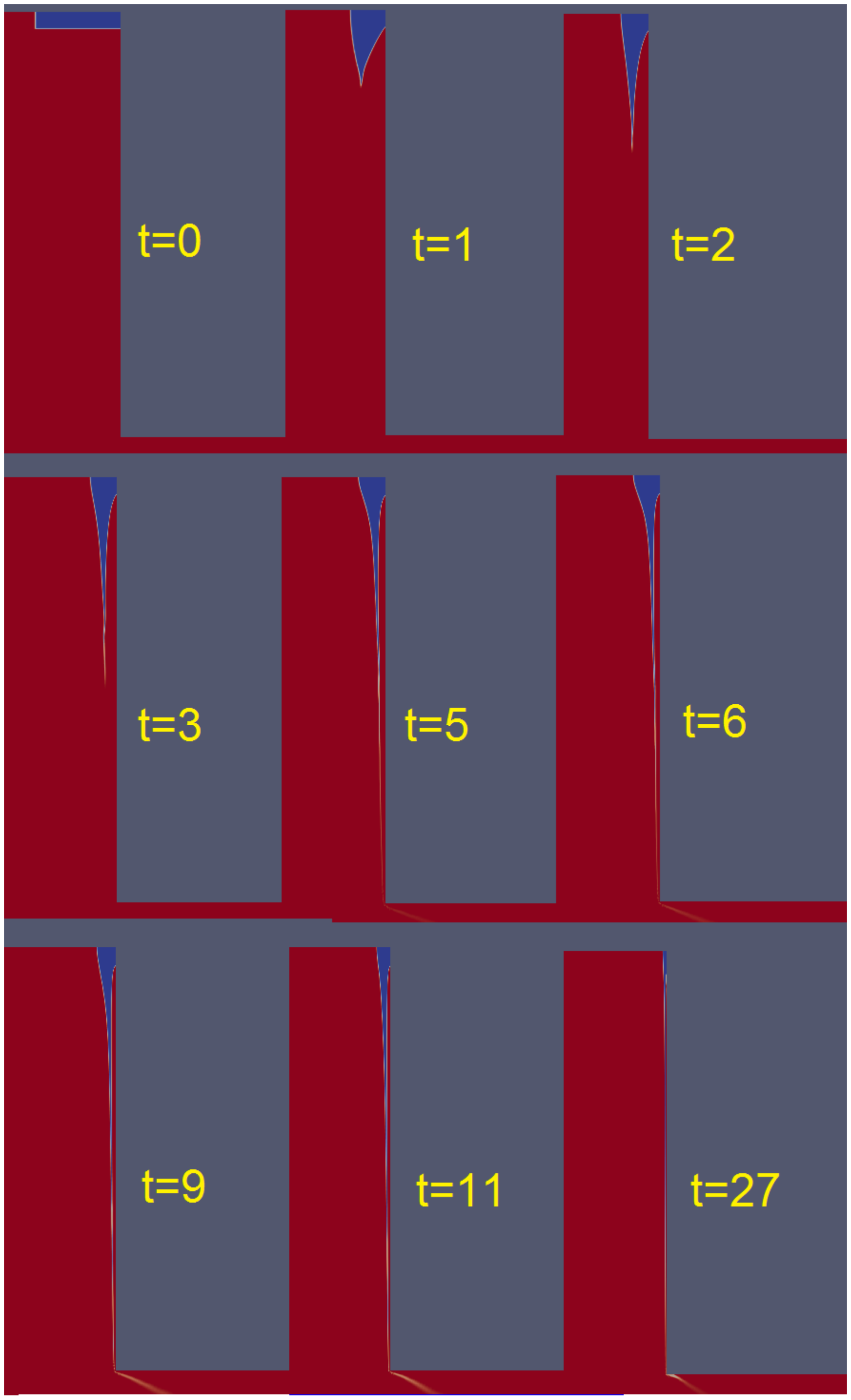

Figure A.46: Flow pattern with Re=100

A.11 Higher viscosity

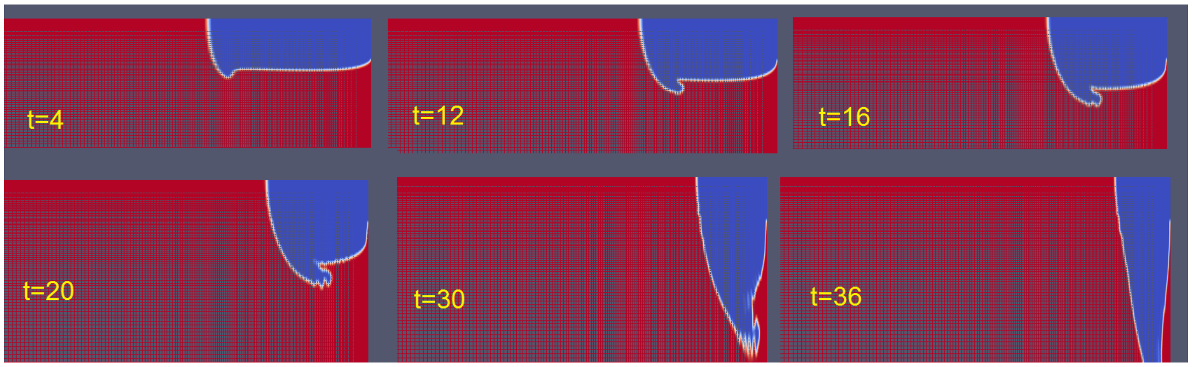

Figure A.47: $\nu_{oil} = 1$, Re=100

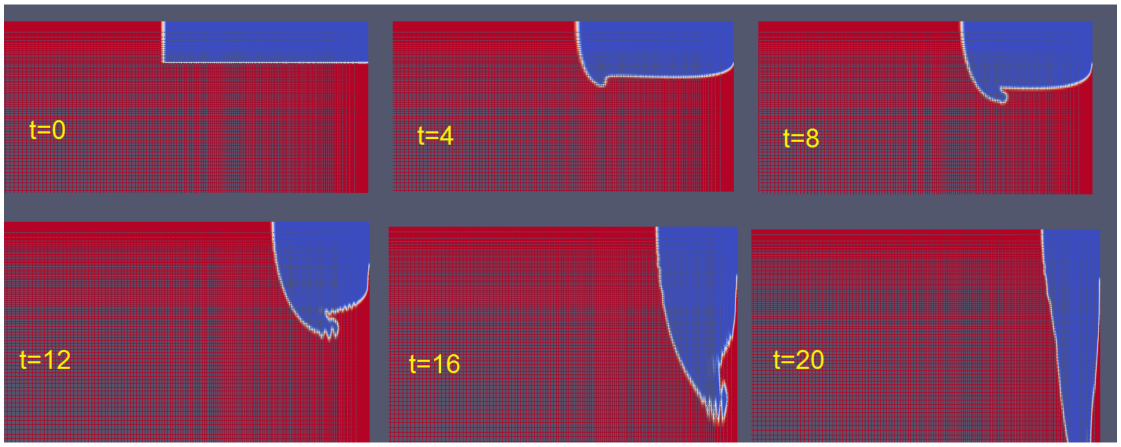

Figure A.48: $\nu_{oil} = 1$, Re=150

Appendix B

OpenFOAM control files

B.1 transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases (water oil);
```

```
water
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 0.01;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1030;
    CrossPowerLawCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 0.01;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 0.01;
        m               m [ 0 0 1 0 0 0 0 ] 1;
        n               n [ 0 0 0 0 0 0 0 ] 0;
    }

    BirdCarreauCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 0.01;
        k               k [ 0 0 1 0 0 0 0 ] 99.6;
        n               n [ 0 0 0 0 0 0 0 ] 0.1003;
    }
}

oil
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 1;
    rho             rho [ 1 -3 0 0 0 0 0 ] 900;
    CrossPowerLawCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 0.03;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 0.03;
        m               m [ 0 0 1 0 0 0 0 ] 1;
        n               n [ 0 0 0 0 0 0 0 ] 0;
    }
    BirdCarreauCoeffs
    {
        nu0             nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
        nuInf           nuInf [ 0 2 -1 0 0 0 0 ] 0.03;
        k               k [ 0 0 1 0 0 0 0 ] 99.6;
        n               n [ 0 0 0 0 0 0 0 ] 0.1003;
    }
}

sigma           sigma [ 1 0 -2 0 0 0 0 ] 0.013;


// ************************************************************************* //
```

B.2 controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     interFoam;
```

```
startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         2;

deltaT          0.001;

writeControl    adjustableRunTime;

writeInterval   1;

purgeWrite      0;

writeFormat     ascii;

writePrecision  12;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable no;

adjustTimeStep  yes;

maxCo           1;
maxAlphaCo      0.15;

maxDeltaT       1;
```

```
functions
{
    forceCoeffs
    {
        type        forceCoeffs;
        functionObjectLibs ( "libforces.so" );
        outputControl timeStep;
        outputInterval 1;

        patches     ( walls );
        pName       p;
        UName       U;
        rhoName     rhoInf;
        log         true;
        rhoInf      1000;
        liftDir     (0 1 0);
        dragDir     (1 0 0);
        CofR        (0 0 0);
        pitchAxis   (0 0 1);
        magUInf     1;
        lRef        0.5;
        Aref        1.0;
    }
    forces
    {
        type        forces;
        functionObjectLibs ( "libforces.so" );
        outputControl timeStep;
        outputInterval 1;

        patches     ( walls );
        pName       p;
        UName       U;
        rhoName     rhoInf;
        log         true;
        rhoInf      1000;
        origin      ( 0 0 0 );
        CofR        ( 0 0 0 );
    }
}
// ************************************************************************* //
```

B.3 fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         CrankNicolson 0.95;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    div(rhoPhi,U)  Gauss linearUpwind grad(U);
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha) Gauss linear;
    div((muEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p_rgh;
    pcorr;
    alpha.water;
}


// ************************************************************************* //
```

B.4 fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.water.*"
    {
        nAlphaCorr      10;
        nAlphaSubCycles 1;
        cAlpha          1;

        MULESCorr       yes;
        nLimiterIter    20;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-20;
        relTol          0;
    }
    pcorr
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-5;
        relTol          0;
    }

    p_rgh
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-07;
        relTol          0.05;
    }

    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0;
    }
}

PIMPLE
{
    momentumPredictor   no;
    nOuterCorrectors    1;
    nCorrectors          3;
    nNonOrthogonalCorrectors 0;
    pRefCell             0;
    pRefValue            0;
}

relaxationFactors
{
    fields
    {
    }
    equations
    {
        ".*" 1;
    }
}


// ************************************************************************* //
```