Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Energy and Process Engineering

Andreas Brostrøm

Numerical simulation of circulation in boundary layers over sinusoidally modulated walls

Master's thesis in Mechanical Engineering
Supervisor: Simen Andreas Ådnøy Ellingsen
July 2020

NTNU
Norwegian University of
Science and Technology

Andreas Brostrøm

Numerical simulation of circulation in boundary layers over sinusoidally modulated walls

Master's thesis in Mechanical Engineering
Supervisor: Simen Andreas Ådnøy Ellingsen
July 2020

Norwegian University of Science and Technology
Faculty of Engineering
Department of Energy and Process Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

Shear flows over sinusoidally modulated boundary walls are known to produce rotational currents in the fluid, in the form of circulation cell pairs with opposite rotational direction. The rotational axis of the cells are parallel to the streamwise direction of the flow. These cells are dynamically equivalent to Langmuir Circulations, which are rotational cell structures that may form under a water surface in the presence of waves and wind above the surface (Akselsen and Ellingsen 2019). These rotational structures are observed to become unstable for sufficiently high Reynolds numbers. The main objective of the thesis work has been to develop a CFD model in OpenFOAM 7 for studying the transient development of the circulation cells when the flow goes from stable to unstable. To accomplish this, the following research questions are defined:

- In what way does the circulation cells evolve to become unstable?
- What is the time scale for the circulation cells to break down?
- How does different flow and domain parameters affect the instability of the cells?

A body force driven shear flow between mirrored, three dimensional sinusoidal walls is considered to model the problem. Results show that the Langmuir Circulation cells breaks down due to the formation of additional circulation cell pairs in line with the peaks and troughs of the wavy wall boundaries. The additional cell structures have the opposite rotational direction compared to the original cells. When the original cells have disappeared, the new cell structures becomes unstable, and the flow shows signs of transitioning to turbulence. The time scale of the cell structure break down is shown to be slow, and the flow has travelled trough the periodic domain of three streamwise wall periods approximately 120 times before the original structures have vanished completely.

To answer the final research question, a parameter study is performed, where the nominal friction Reynolds number Re_τ and the maximum streamwise wall wave steepness α_x are varied systematically. Results from the parameter study shows that the decrease in friction Reynolds number acts stabilising on the circulation cells. Increasing Re_τ causes a more rapid break down of the cells, and also earlier transitioning away from a fully laminar flow. By sufficiently decreasing the wave steepness α_x, the flow is shown to become steady and the regular structure of the circulation cells is kept. Increasing α_x accelerates the break down of the Langmuir cells, and also the transition to turbulent flow behaviour.

Sammendrag

Skjærstrømninger over sinusoidalt modulerte vegger har vist å produsere roterende strømninger i et fluid i form av sirkulerende cellepar med motsatt roterende retning. Cellenes roterende akse er parallell med strømningsretningen til fluidet. Disse cellene er dynamisk ekvivalente til Langmuirsirkulasjoner, som er roterende cellestrukturer som kan oppstå under vannoverflaten i nærvær av bølger og vind over overflaten (Akselsen and Ellingsen 2019). Cellestrukturene har blitt observert til å bli ustabile for tilstrekkelig høye Reynoldstall. Hovedformålet med denne avhandlingen har vært å utvikle en CFD-modell i OpenFOAM 7 for å studere den transiente utviklingen av sirkulasjonscellene når de går fra stabile til ustabile. For å oppnå dette, blir de følgende forskningsspørsmålene definert:

- På hvilken måte utvikler sirkulasjonscellene seg til å bli ustabile?
- Hva er tidsskalaen for at sirkulasjonscellene skal brytes ned?
- Hvordan påvirker forskjellige strømnings- og domeneparametre instabiliteten til cellene?

En volumkraftdrevet skjærstrømning mellom speilede, tredimensjonale sinusoidale vegger er betraket for å modellere problemet. Resultater viser at Langmuirsirkulasjonscellene brytes ned ved at ytteligere sirkulasjonscellepar oppstår på linje med toppene og bunnene av de bølgete veggene. Disse cellestrukturene har motsatt rotasjonsretning sammenliknet med de originale cellene. Når de originale cellene har forsvunnet, blir de nye cellestrukturene ustabile, og viser tegn på overgang til turbulens. Tidsskalaen for at cellestrukturene skal brytes ned vises å være treg, og strømningen har beveget seg tilnærmet 120 ganger gjennom det periodiske domenet med tre veggperioder i strømningsretningen før de originale strukturene har forsvunnet fullstendig.

For å svare på det siste forskningsspørsmålet, har en parameterstudie blitt gjennomført, hvor det nominelle vegg-Reynoldstallet Re_τ og den maksimale bølgekrappheten i strømningsretningen α_x blir variert systematisk. Resultater fra parameterstudien viser at en reduksjon av vegg-Reynoldstall virker stabiliserende på sirkulasjonscellene. En økning i Re_τ fører til en raskere nedbrytning av cellene, og også en tidligere overgang fra fullt laminær strømning. Ved å minke bølgekrappheten α_x tilstrekkelig, vises det at strømingen blir stasjonær og at de regulære strukturene i sirkulasjonscellene forblir. Økning av α_x akselererer nedbrytningen av Langmuircellene, og også overgangen til turbulent oppførsel i strømningen.

Preface

This master's thesis was carried out during the spring semester of 2020 and was submitted to the Department of Energy and Process Engineering at the Norwegian University of Science and Technology. The thesis is submitted as part of the five year study program Mechanical Engineering, and it comprises of 30 ECTS.

I would like to thank my supervisor Simen A. Ådnøy Ellingsen for his guidance on the thesis work in providing invaluable knowledge regarding the flow problem at hand, and for his supportive and resourceful way of helping me when I have been faced with problems. I would also like to thank my co-supervisor Are Simonsen for our discussions regarding computational fluid dynamics and solution strategies to the problem.

Table of Contents

List of Tables

List of Figures

Nomenclature

Variables and Parameters

$\mathbf{u}$	=	Velocity vector, $[\mathrm{m\,s^{-1}}]$
$\mathbf{x}$	=	Cartesian coordinate directions, $[\mathrm{m}]$
t	=	Time, $[\mathrm{s}]$
H	=	Channel mean height, $[\mathrm{m}]$
L	=	Channel length, subscript specifies coordinate direction, $[\mathrm{m}]$
a	=	Amplitude of wall boundaries, $[\mathrm{m}]$
α	=	Wall wave steepness, subscript specifies coordinate direction, [-]
k	=	Wave number of wall boundaries, subscript specifies coordinate direction, $[\mathrm{m^{-1}}]$
θ	=	Half-angle between wave trains forming the walls, [-]
U_0	=	Maximum streamwise velocity, $[\mathrm{m\,s^{-1}}]$
U_{flat}	=	Velocity profile for plane Poiseuille flow, subscript 0 specifies at centre-plane, $[\mathrm{m\,s^{-1}}]$
u_*	=	Friction velocity, $[\mathrm{m\,s^{-1}}]$
l^+	=	Viscous length scale, $[\mathrm{m}]$
τ_w	=	Wall shear stress, $[\mathrm{N\,m^{-2}}]$
Re_0	=	Reynolds number based on maximum streamwise velocity, [-]
Re_τ	=	Reynolds number based on friction velocity, [-]
ρ	=	Fluid density, $[\mathrm{kg\,m^{-3}}]$
ν	=	Fluid kinematic viscosity, $[\mathrm{m^2\,s^{-1}}]$
μ	=	Fluid dynamic viscosity, $[\mathrm{kg\,m^{-1}\,s^{-1}}]$
P	=	Pressure, $[\mathrm{N\,m^{-2}}]$
$\mathbf{F}$	=	Momentum source term, $[\mathrm{N\,m^{-3}}]$
η	=	Boundary layer similarity variable, [-]
ψ	=	Boundary layer stream function, $[\mathrm{m^2\,s^{-1}}]$
δ	=	Boundary layer thickness, $[\mathrm{m}]$
U	=	Free stream velocity, $[\mathrm{m\,s^{-1}}]$
S	=	Theoretical speed-up from Amdahl's law, [-]
f_p	=	Portion of program benefiting from parallelisation, [-]
n_c	=	Number of processor cores, [-]
C	=	Courant number, subscript denotes average or maximum value [-]
ω	=	Vorticity, subscript specifies coordinate direction, $[\mathrm{s^{-1}}]$
$\dot{m}$	=	Mass flow rate $[\mathrm{kg\,s^{-1}}]$

Superscripts

$*$	=	Nondimensionalised by large scale quantities, e.g. U_0 & H
$+$	=	Nondimensionalised by viscous length scale $l^+ = \nu/u_\tau$

Abbreviations

NTNU	=	Norwegian University of Science and Technology
CFD	=	Computational fluid dynamics
FVM	=	Finite volume method
NS	=	Navier-Stokes
CFD	=	Computational fluid dynamics
LC	=	Langmuir Circulations
LBM	=	Lattice Boltzmann method
GUI	=	Graphical user interface
CAD	=	Computer-aided design
STL	=	Stereolithography
AMI	=	Arbitrary mesh interface
SIMPLE	=	Semi-Implicit Method for Pressure Linked Equations
PISO	=	Pressure-Implicit with Splitting of Operators
BDF2	=	Backward differentiation formula of second order
GAMG	=	Geometric algebraic multigrid
HPC	=	High performance computing
SSH	=	Secure Shell
SFTP	=	Secure File Transfer Protocol
ODE	=	Ordinary differential equation
CPU	=	Central processing unit
i.e.	=	id est, meaning ”that is”
e.g.	=	exempli gratia, meaning ”for example”

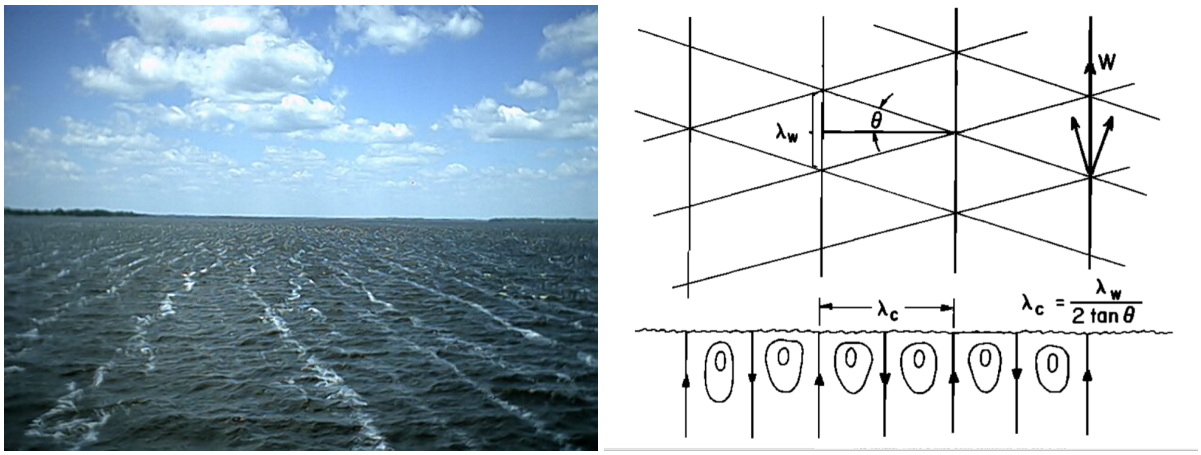
Chapter 1

Introduction

This chapter will present the background for the thesis work. Following will be the motivation for why it is an important topic of research. Research questions will be defined in order to provide the foundation for the analysis to be conducted. Finally, a chapter outline is given, where the structure of the thesis is explained.

1.1 Background

When looking out over the ocean on a windy day, one might observe multiple distinct lines of oil, foam or other small debris forming as parallel streaks on the surface of the water, as can be shown by picture in figure 1.1a. These streaks are known as windrows, and they arise as a consequence of a physical phenomenon called Langmuir Circulations (LC). Langmuir Circulations are rotating cells of fluid forming close to the surface. The rotational direction alternates, creating pairs of counter rotating cells. Their rotational axes aligns nearly parallel to the wind direction above the surface, with some deviation (Faller 1964). Langmuir Circulations are driven by the wind, as was first concluded by Irving Langmuir in his pioneering paper from 1938 (Langmuir 1938). The windrows are formed in the converging sections in between the rotating cells, i.e. where two cells both cause a downwelling current. This is illustrated in the lower part of figure 1.1b. On the opposite side of the cells, the combined upwelling current of both cells causes a diverging section which transports particles upwards, across the top the cells, and towards the converging sections where the windrows becomes visible.



(a) **(b)**

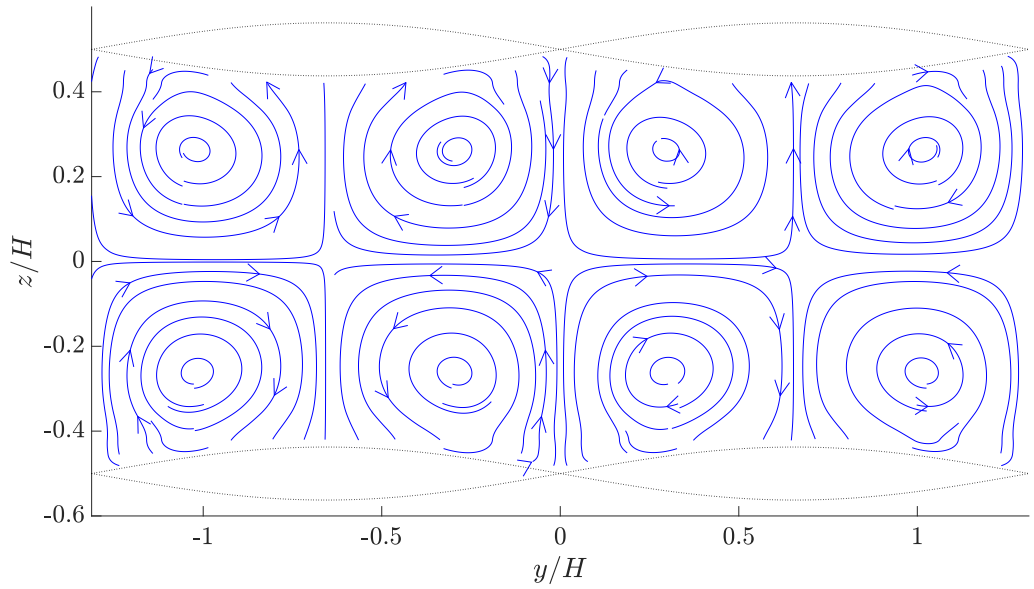
Figure 1.1: (a) Picture showing the formation of windrows above Langmuir Circulations. Taken from (Thurnherr 2002). (b) Above: Schematic figure depicting the two wave trains and the wind direction. Below: Cross section of the circulation cells creating upwelling and downwelling current. Taken from figure 1 in (Faller and Caponi 1978)

Although the wind is recognised as the driving force of the phenomenon, this does not explain the physical mechanism creating the circulation patterns. A suggested answer to this is a mechanism referred to as CL1, first introduced in (Craik 1970), and further developed in (Craik and Leibovich 1976). The model is based on the interaction between a crossed-wave pattern arising from superposition of oblique plane wave trains, and a shear flow driven by the wind stress on the water surface (Faller and Caponi 1978).

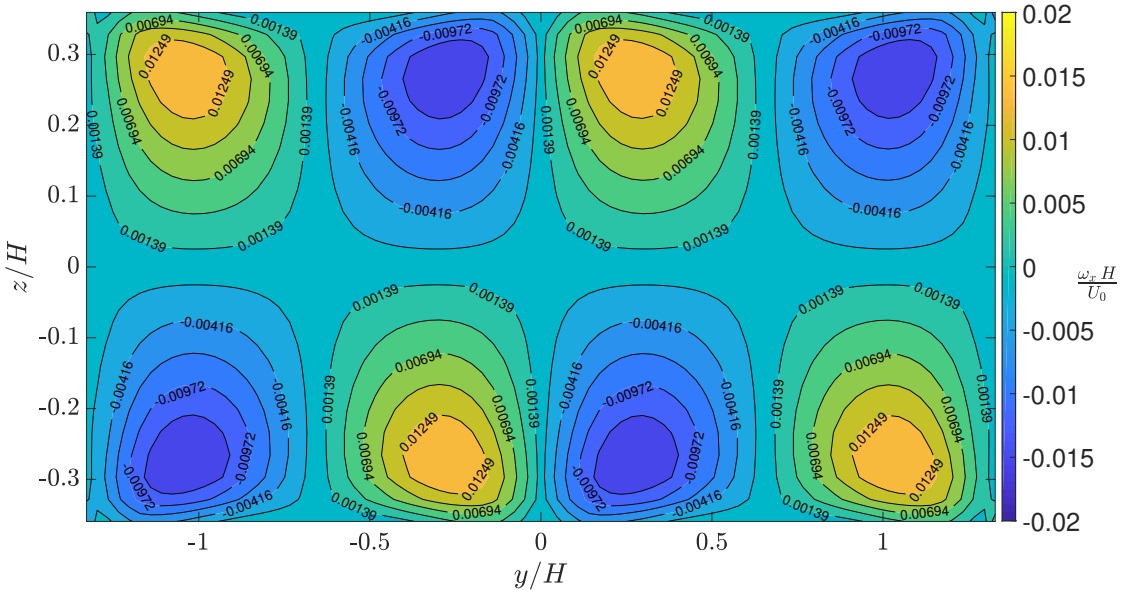
A recent theory by (Akselsen and Ellingsen 2019) [1] has established that a change of reference frame, making the bathymetry a certain symmetric, three-dimensional stationary boundary, will also induce the CL1 mechanism when a shear flow is driven over it. Instead of the wave patterns on the fluid surface, an undulating wall will enforce the wave motion onto the shear flow. The concept is illustrated in figure 1.3. To supplement their theoretical findings, they have done numerical simulations using the lattice Boltzmann method (LBM), with a code written in MATLAB. LBM has been proved successful in modelling flows involving complex boundaries, however it is not very efficient in terms of computational cost. Unlike conventional computational fluid dynamics (CFD) schemes, where macroscopic continuum equations are discretized, the LBM method uses microscopic models and mesoscopic kinetic equations (Chen and Doolen 1998). This method models the flow as inherently compressible, however, due to low flow speeds in the study, the compressible effects are considered negligible (Rathakrishnan 2019). Formation of circulation cells have previously been observed over hilly terrain (Gong, Taylor, and Dörnbrack 1996), and its linkage to Langmuir Circulations are supported through another driving mechanism, namely the CL2 mechanism (Craik 1977; Leibovich 1977; Phillips,

[1] Since the paper is not yet published, it is available from the authors by contacting simen.a.ellingsen@ntnu.no

Wu, and Lumley 1996). The CL2 mechanism requires no coherent surface-wave structures, but rather explains the circulations as a result of the vortex force that stems from an inviscid instability in the current (Leibovich 1983). Secondary circular structures in the flow over wavy geometry has been observed using direct numerical simulations (Chan et al. 2018). In this study, pipe flow was considered, and at fully turbulent Reynolds numbers.



(a)



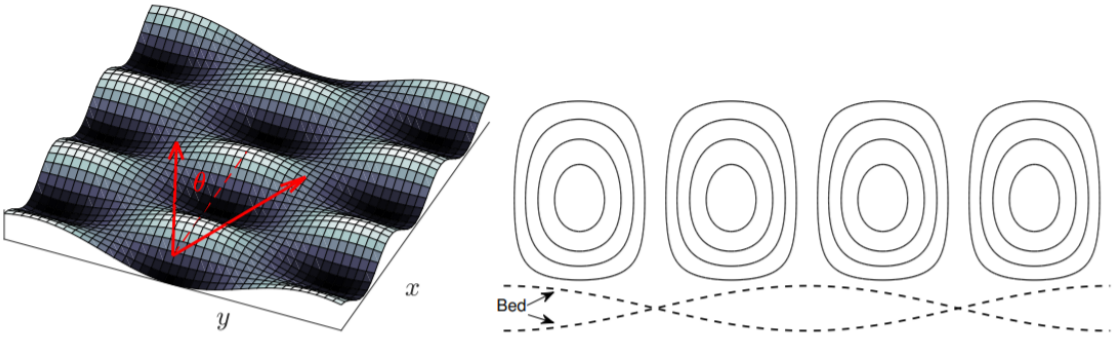
(b)

Figure 1.2: Visual representation of the steady state Langmuir Circulations predicted by (Akselsen and Ellingsen 2019). (a) Streamwise averaged streamline results, (b) Streamwise averaged x vorticity results. The figures are taken from (Brostrøm 2019)

In the project work conducted in the autumn semester (Brostrøm 2019), the analogous phenomenon to the Langmuir Circulations in a closed channel with wavy walls were studied using a more conventional CFD tool, namely ANSYS Fluent. In this study, the steady state solution of the flow was analysed and compared with results from (Akselsen and Ellingsen 2019).
Figure 1.2a shows streamwise-averaged results for the normal velocity fields, visualised using streamlines. Figure 1.2b shows the streamwise-averaged results for the streamwise vorticity field. Both figures are taken from the project work (Brostrøm 2019), and are results from a given set of parameters. The results give indication of distinct rotational structures in the flow, supporting the claims from (Akselsen and Ellingsen 2019). A parameter study was conducted where the nominal friction Reynolds number Re_τ of the flow, and the half-angle θ between the wave trains forming the wavy wall boundaries, were altered systematically. Here, Re_τ is based on the friction velocity u_*, the mean half-height of the channel $H/2$ and the fluid kinematic viscosity ν, while θ can be visualised from the left part of figure 1.3. The parameter study was done to investigate how these parameters affected the formation of Langmuir cells in the flow, except for some deviations in value amplitudes, the results showed convincing correspondence with those from (Akselsen and Ellingsen 2019). An interesting result from the study was that for sufficiently high Reynolds numbers, $Re_\tau > 40$, the stable circulation patterns which for lower Re_τ converged to a steady state solution, began to break down. At $Re_\tau = 40$ for a plane channel flow, the Reynolds number based on the centre-plane velocity is $Re_0 = 1600$ exactly. This Reynolds number is based on the full channel height H. Results from (Orszag 1971) show that the critical Reynolds number of a plane Poiseuille flow is 5772. However, this is when the half-height $H/2$ is used in the definition, meaning the critical Reynolds number in the present case would be $Re_{0,c} = 11544$. Because of this, the break down of the circulation cells is not suspected to be caused by the flow becoming turbulent, but possibly by an instability in the physical mechanism responsible for creating the cells structures. This transient break down of the Langmuir cells will be investigated throughout this thesis, using CFD methodology.

1.2 Motivation

Other than creating fascinating patterns on the surface, the physical implications of Langmuir Circulations in the ocean are also of importance. Evidence suggest that the circulations may be a principal component of upper layer mixing in the ocean (Craik 1970; Leibovich 1983). This makes the circulation currents an important part in the development of climate and weather models, due to a contribution to ocean-air fluxes of heat, mass and momentum (Kukulka et al. 2009). It is noted that it is difficult to quantify this contribution, because it is difficult to separate the effect of Langmuir Circulations from ordinary turbulence (Scott et al. 1969). The implications of the undulating wall analogy to the conventional Langmuir Circulations could be many. Temperature exchange between the fluid and the solid boundary through enhanced



.

Figure 1.3: The undulating wall with wave vectors $(k_x, \pm k_y)$ and the circulation cells resulting from the shear flow over it, as illustrated by Akselsen and Ellingsen. This is figure 2 from their paper (Akselsen and Ellingsen 2019)

mixing would be one of them. In addition, transport of turbulence generated near the wall into the bulk flow may be of interest. This would affect the drag on the boundary (Akselsen and Ellingsen 2019), and studying it could give valuable insight in understanding how certain roughness patterns affect friction on walls from the fluid flow. To be able to utilise such a physical phenomenon effectively, it is important to understand its behaviour, and of course its inherent limitations. Therefore it is important to know what kinds of flow conditions that are necessary for the circulation structures to exist.

1.3 Research objectives

The main objective of this thesis is to develop a CFD model for simulating a fluid flow in between sinusoidally modulated boundary walls. This model will be used to investigate the transient behaviour of the Langmuir Circulation cells when flow and geometrical properties are set to be in the regime where the cells are expected to break down. In order to systematically say something about this, three research questions questions are stated as follows:

- In what way does the circulation cells evolve to become unstable?
- What is the time scale for the circulation cells to break down?
- How does different flow and domain parameters affect the instability of the cells?

These questions are to be considered in the structuring of the analysis, and will be answered in the end of the thesis. To answer the first two questions, a base case will be defined and analysed. In order answer the last question, a parameter study will be conducted where the friction Reynolds number will be varied within the range were cell break down is expected to

occur. In addition, streamwise wave steepness of the wall boundary will also be varied, and the transient effects on the circulation structures will be analysed.

Version 7 of the CFD software `OpenFOAM` will be utilised to create the model and run the simulations. A major benefit in choosing OpenFOAM as the simulation software, as opposed to e.g. Fluent or Star CCM+, is the fact that OpenFOAM is an open source software, i.e. the source code is public and the software is free. This means that the user is given the opportunity to both study the solver implementation, and to make custom modifications to the solver if needed.

1.4 Chapter outline

Chapter two will present the flow problem in detail, and provide the mathematical descriptions for the boundaries of the flow domain, as well as formal definitions of the flow and geometrical properties. Chapter three will describe some theoretical background for the flow, including analytical derivations for laminar channel flow, needed to systematically vary the nominal friction Reynolds number and body forcing of the flow. Chapter four contains the methodology for the analysis, and will provide details about the discretisation of the domain, the boundary conditions used, and other parts of the implementation and solving in OpenFOAM. This chapter also includes the verification of the model, and validation of the CFD code. Chapter five will present the results of the simulations in a systematic way, and aim to show meaningful visualisations of the solution data, which will be used to classify and explain the behaviour of the flow structures. Chapter six contains the discussion of the results, where the different solutions from the simulations will be compared in light of the research questions defined in chapter one. Finally, in chapter seven, the conclusion to the research questions will be given.

Chapter 2

Flow problem description

This chapter will present the flow problem studied in the present work. Mathematical definitions for the parametrization of the boundary walls, as well as necessary flow and domain properties will be given.

In the study done by (Akselsen and Ellingsen 2019), both a free surface and a mirrored, shifted version of the lower boundary was utilised as the upper boundary of the flow domain. In this thesis, the analysis is limited to the mirrored and non-shifted version of the domain. If the analysis were to include the alternative of having a free surface at upper boundary of the fluid, additional modelling of that free surface would have to be included. This could be done by using the Volume of fluid method (Hirt and Nichols 1981), but it would increase the complexity of the problem, and also the computational cost of solving it. Both (Akselsen and Ellingsen 2019) and (Brostrøm 2019) found clear formations of circulation cells with the two-wall formulation, so using this setup is considered reasonable for studying the stability of the circulations as well. By choosing this, the problem is turned into a closed off flow in between two wavy walls, almost a channel flow, only with wavy walls. The mean height of walls are are defined to be a distance H apart.

The wavy walls are sinusoidally corrugated in two dimensions, and are mathematically parameterized as a surface $z_b(x, y)$, given by

$$z_b(x,y) = \begin{cases} +H/2 - a\,\sin(k_x\,x)\,\sin(k_y\,y) & \text{at upper wall,} \\ -H/2 + a\,\sin(k_x\,x)\,\sin(k_y\,y) & \text{at lower wall,} \end{cases} \tag{2.1}$$

where z is the flow normal, vertical direction, and $z = 0$ describes the midplane between the two walls. The streamwise and transverse coordinates are denoted by x and y respectively, and k_x and k_y are the corresponding wavenumbers for the boundary walls. The a denotes the amplitude of the wavy walls from their mean z locations of $\pm H/2$. Periodic boundary conditions are used in both the streamwise and transverse directions to greatly reduce the domain size and therefore

the computational cost of solving the problem. Three periods of the wavy wall are used in both directions to define the domain. This is further discussed in Chapter four, in the section about boundary and initial conditions. Figure 2.1 illustrates the flow problem geometry.

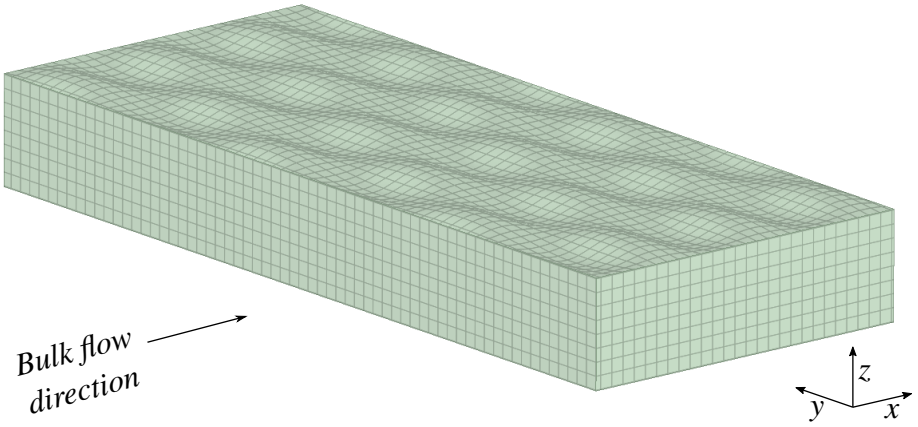


Figure 2.1: Picture of fluid domain, with coordinate directions indicated. Grid lines are added to help show the curvature of the wavy walls, and must not be mistaken for the mesh.

Two different Reynolds numbers are defined in order to describe the flow problem, namely

$$Re_0 = \frac{U_0 H}{\nu}, \quad Re_\tau = \frac{u_* H/2}{\nu} \tag{2.2}$$

where Re_0 is based on the maximum velocity U_0 in the vicinity of the midplane, the mean channel height H and the fluid kinematic viscosity ν. The nominal friction Reynolds number Re_τ is based on the friction velocity u_* and the channel half-height $H/2$. A mathematical definition of the two velocities is given by

$$U_0 = \max_{\Omega}\{u(x,y,z)\}, \quad u_* = \sqrt{\frac{\tau_w}{\rho}}, \tag{2.3}$$

where u is the streamwise velocity component, Ω is the fluid domain, τ_w is the wall shear stress and ρ is the fluid density. The nominal friction Reynolds number will be varied systematically to study its effect on the vortex cell stability. In addition, the effect of varying the maximum wave steepness of the wall in the stream wise direction $\alpha_x = k_x a$, while the wave steepness in the transverse direction α_y is kept constant, will also be studied. This is equivalent to changing the half-angle θ between the two oblique wave trains with wave vectors $\mathbf{k} = [k_x, \pm k_y]$ which together forms the undulating boundary wall. The angle θ is visualised in figure 1.3 in the

introduction. An implicit definition of this angle can be expressed as

$$\tan(\theta) = \frac{k_y}{k_x}. \tag{2.4}$$

Chapter 3

Theory

In this chapter, some theoretical background for the fluid flow problem will be presented, including analytical derivations for laminar channel flow needed to systematically vary the nominal friction Reynolds number and body forcing of the flow.

3.1 Governing equations

The flow is modelled by the unsteady, incompressible Navier-Stokes equations, consisting of the continuity equation, given by

$$\nabla \cdot \mathbf{u} = 0, \tag{3.1}$$

and the momentum equations, written as

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\,\nabla^2 \mathbf{u} + \frac{\mathbf{F}}{\rho}, \tag{3.2}$$

where $\mathbf{u}$ is the velocity vector field, ρ is the fluid density, p is the pressure field, ν is the fluids kinematic viscosity, and $\mathbf{F}$ is the momentum source term driving the flow, given in $[\mathrm{N\,m^{-3}}]$. The ∇ is a vector operator and is defined by

$$\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right). \tag{3.3}$$

Only the streamwise momentum component will have a source term, meaning $\mathbf{F} = (F, 0, 0)$. Together with no-slip boundary conditions on the walls, F will induce a boundary layer shear flow, providing the necessary conditions for the CL1 mechanism to create the Langmuir Circulations. Details on how to determine the forcing term in a systematic way will be discussed in the last section of this chapter.

The Navier-Stokes equations will be solved in its dimensional form, however nondimensional quantities are defined in order to present the result in a more general form. The di-

mensionless quantities are defined using the maximum flow velocity U_0 and the average height between the wavy walls H, and are given by

$$\mathbf{x}^* = \frac{\mathbf{x}}{H}, \quad \mathbf{u}^* = \frac{\mathbf{u}}{U_0}, \quad t^* = \frac{t\,U_0}{H}, \quad \mathbf{k}^* = \mathbf{k}\,H, \quad p^* = \frac{p}{\rho\,U_0^2}, \tag{3.4}$$

where t is the time and superscript "$*$" denotes dimensionless quantities based on the mentioned large scale quantities.

Details of the theoretical approach for studying the analogous Langmuir Circulation phenomenon is complicated, and considered outside the scope of this thesis, as this study focuses on the computational fluid dynamics of the flow problem. The theoretical details will therefore not be discussed in the present study. The interested reader is referred to (Akselsen and Ellingsen 2019) for insight into the theoretical solution procedure.

3.2 Boundary layers

The formation of a boundary layer is considered an essential part in the CL1 description on why Langmuir Circulations are formed, due to the fact that this is where the shear of the flow is dominant. For Langmuir Circulations near the ocean surface, a boundary layer is formed by the shear stress that the wind exerts on the water surface. For the channel flow analogy, the boundary layer is formed due to the no-slip boundary condition enforced for viscous fluids at solid walls. Since velocity shear is considered so important for the formation of the circulation structures, a CFD code used to study them should be able to accurately calculate the shear inside a boundary layer. A test case will therefore be included as part of the validation of the CFD code used in OpenFOAM. The test case will be the development of a laminar boundary layer over a semi-infinite flat plate, as illustrated in figure 3.1. It is noted that the coordinate y represents the vertical direction in this test case, however, for the original three dimensional problem, it defines the transverse direction. In a boundary layer, the momentum equations in the steady Navier-Stokes equations reduce to

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = U\frac{dU}{dx} + \nu\frac{\partial^2 u}{\partial y^2}, \tag{3.5}$$

$$0 = -\frac{1}{\rho}\frac{\partial p}{\partial y}, \tag{3.6}$$

where U is the free stream velocity above the boundary layer. Only two spatial directions are included, as the solution is independent of the transverse direction for a laminar boundary layer over the flat plate, making it a two dimensional problem. In (3.6), all terms but one disappears since the normal velocity v is negligible near the wall, and this means p can only be a function of x. The $U\frac{dU}{dx}$ term in (3.5) has replaced the $-\frac{1}{\rho}\frac{\partial p}{\partial x}$ from the Navier-Stokes equations. This

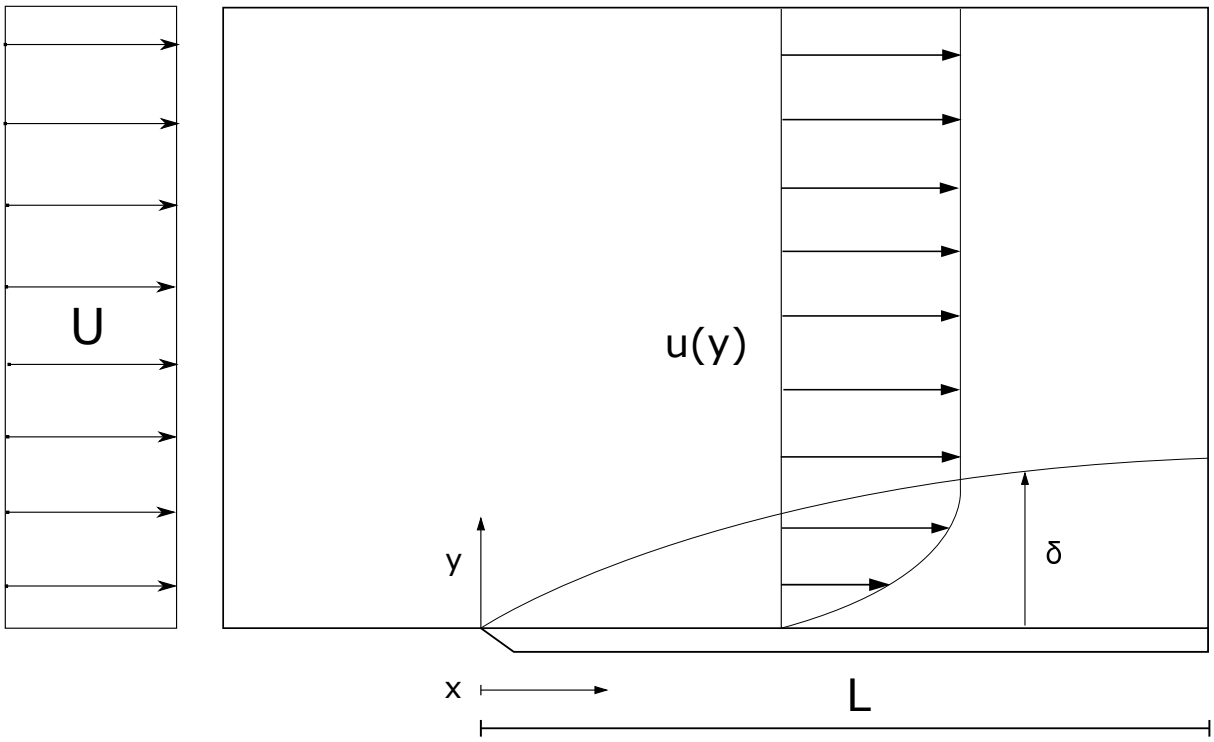


Figure 3.1: Laminar boundary layer development over a flat plate

can be justified with the Bernoulli equation, which states that

$$p + \frac{1}{2}\rho U^2 = \text{constant} \tag{3.7}$$

in the free stream. Since p only can be a function of x, and U is assumed to be at most a function of x, taking the x-derivative of (3.7) shows that the two terms are interchangeable. The $\nu\dfrac{\partial^2 u}{\partial x^2}$ term from NS is neglected due to small variations in x compared to y. To further simplify (3.5), U is assumed to be constant, such that the equation reduces to

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\frac{\partial^2 u}{\partial y^2}. \tag{3.8}$$

Now, a similarity variable η and a stream function ψ is introduced and defined as

$$\eta = y\sqrt{\frac{U}{\nu x}}, \tag{3.9}$$

$$\psi = \sqrt{\nu U x} f(\eta) \tag{3.10}$$

respectively. Here, $f(\eta)$ is some unknown function of η. The goal is to be able to describe the boundary layer shape only in terms of η. This is done by first setting $u = \dfrac{\partial \psi}{\partial y}$ and $v = -\dfrac{\partial \psi}{\partial x}$,

giving

$$u = Uf'(\eta), \tag{3.11}$$

$$v = \sqrt{\frac{\nu U}{x}}(\eta f'(\eta) - f(\eta)). \tag{3.12}$$

All terms in (3.8) can now be calculated and inserted back into the equation. The resulting equation can, with some reordering be written as

$$f''' + \frac{1}{2}ff'' = 0. \tag{3.13}$$

Since f is only a function of η, then so is its derivatives, and therefore the goal to describe the boundary layer only in terms of η is achieved. The equation to solve is now an ordinary differential equation (ODE) and it can be solved numerically. Since this is a third order ODE, three boundary conditions are needed. These can be found from (3.11) and (3.12). The no slip condition at the wall makes $f'(0) = 0$ and the fact that u should tend to U in the free stream gives $f'(\infty) = 1$. Finally, v should be equal to zero at the wall, so from (3.12) it is necessary that $f(0) = 0$ must also hold. One boundary condition is at infinity, so a shooting method can be used iteratively to solve (3.13) in order to find $f(\eta)$ and its derivatives. This will not be done in the present study, so the CFD data from the test case will be compared to results of (3.13) from (Lal and Paul M 2014).

3.3 Momentum source term

Since the flow will be specified by a nominal friction Reynolds number Re_τ, some connection between this number and the momentum source term has to be made in order to correctly drive the flow. In order to do this, an ordinary Poiseuille flow between two parallel plates with a distance H apart is considered. This connection to the original problem is justified by having the amplitude small compared to the channel height. It is noted that the true $Re_{\tau,t}$ will vary in different parts of the channel, due to the variation of τ_w over the wavy structures on the walls. In addition, the average Re_τ will be lower than for the flow between flat plates. This is due to the fact that some of the forcing on the flow will be balanced by pressure drag on the walls, since the wall normal vector generally will have a component in the bulk flow direction, and stagnation of the flow causes higher pressure on the upstream sides of the wall peaks than on the downstream sides. This will reduce the average τ_w, in turn reducing the average u_*, which Re_τ is linearly dependent of.

Placing the coordinate system origin on the mid-plane between the two plates, an analytical

expression for the streamwise velocity U_{flat} as a function of normal position z is given by

$$U_{\text{flat}}(z) = \frac{F}{8\,\mu}\left(H^2 - 4z^2\right), \tag{3.14}$$

where $\mu = \rho\,\nu$ is the dynamic viscosity of the fluid. Squaring both sides of the last expression in (2.2) and inserting the last expression in (2.3) gives

$$Re_\tau^2 = \frac{\tau_w H^2}{4\,\rho\,\nu^2}. \tag{3.15}$$

An expression for τ_w can be found by taking the derivative of (3.14) at $z = -H/2$ and multiplying the result by μ, yielding

$$\tau_w = \frac{FH}{2}. \tag{3.16}$$

One additional equation for F is needed, and can be found by considering (3.14) at $z = 0$, giving the centre-plane speed $U_{0,\text{flat}}$ as

$$U_{0,\text{flat}} = \frac{FH^2}{8\,\mu}. \tag{3.17}$$

Then (3.16) and (3.17) are combined and substituted into (3.15), giving

$$Re_\tau^2 = Re_0, \tag{3.18}$$

by recognising Re_0 as defined in (2.2). Thus, by using the definition of Re_0, an expression for the viscosity needed to describe the flow is found to be

$$\nu = \frac{U_{0,\text{flat}} H}{Re_\tau^2}. \tag{3.19}$$

Finally, (3.17) can be solved for F in order to find the expression for the momentum source term needed to drive the flow, reading

$$F = \frac{8\,\rho\,\nu\,U_{0,\text{flat}}}{H^2}. \tag{3.20}$$

Chapter 4

Methodology

The following chapter contains the solution strategy used for the analysis, and will describe the development of the CFD model used to analyse the flow problem in detail. This includes how the flow domain geometry is created, how the meshing is conducted, and how the boundary conditions are specified. Additionally, the verification of the model and the validation of the code used will also be presented here, as essential steps towards producing realistic and trustworthy results.

4.1 Computational fluid dynamics

Computational fluid dynamics (CFD) is a field of science consisting of solving fluid flow problems using numerical analysis. The general concept of CFD is to divide the large and complex fluid problem into lots of smaller and simpler problems that are solved simultaneously. This is done through discretisation of the fluid domain, most commonly dividing it into many small cells. All the cells are treated as control volumes, and the discretized integral versions of the Navier-Stokes equations are solved for every cell. This approach is called finite volume method (FVM), and is the method used in OpenFOAM. CFD is used in a large range scientific and industrial applications, from studying the flow inside blood vessels, to calculating the aerodynamic forces on a rocket flying through the atmosphere. Although analytical solutions to the Navier Stokes equations exist for many flow problems, they are often limited to relatively simple flows, and by that not generally applicable for the complex flows often occurring in the real wold. Therefore, CFD is a necessary tool in studying and understanding fluid flow.

4.2 CFD software

The CFD software used for the analysis in this thesis is OpenFOAM. OpenFOAM, short for Open-source Field Operation And Manipulation, is the leading, free open source software for

computational fluid dynamics (CFD-Direct n.d.). It is a toolbox written in `C++` consisting of libraries with applications for solving a wide range of flow problems and processing data. While OpenFOAM do exists as a Windows program, it is originally made for Linux, where the user will get the most benefit from all the functionality. The use of OpenFOAM mainly consists of editing configuration files, which gives input to a compiled and executable program, all contained and run within the file structure from the command line in Linux. Examples of all relevant configuration files will be included in appendix A for reference. OpenFOAM is different from commercial type CFD software such as ANSYS Fluent, where a graphical user interface (GUI) is used for input of solver settings and other information from the user. From a research point of view, having access to the source code is a major benefit. It enables the user to study the code in detail, and even make changes if needed. Commercial CFD software are generally so-called ”black box”, meaning the user inputs some parameters into the solver and in return is handed some results, but has no way of knowing what is actually happening inside the solver. An advantage of using OpenFOAM in terms of an educational point of view, is that the user is forced to be aware of everything needed to set up a CFD case, as every file has to be modified manually. A disadvantage of using OpenFOAM, is the lack of a GUI. This means that the workflow is often more difficult to get used to and learn, as no visual feedback or guiding is given when setting up the case by navigating through the file tree. In addition, the fact that OpenFOAM is operated in Linux means that the user will have to learn a new operating system in addition to the CFD software itself. Together, this means that OpenFOAM will generally take more time to learn than commercial alternatives.

4.3 Geometry

A three dimensional model of the complete fluid domain is needed in order to discretize the problem for CFD analysis. This is done using a computer-aided design (CAD) software, namely ANSYS SpaceClaim. SpaceClaim is used since it has a built in function that can create mathematically parameterised surfaces. This is ideal for the present case, as the wall boundaries are described mathematically using (2.1). An alternative to such a function could be to create the surface in e.g. MATLAB and exporting it as a stereolithography-file (STL), connecting the points using triangles to form a complete surface. Some problems needs to be addressed when creating the geometry where periodic boundary conditions are to be implemented. A whole number of periods is needed for the end surfaces of the domain to match up, however, small round-off errors in the wave numbers used to define the walls may add up at the ends, creating a mismatch between them. Caution is therefore needed when creating the geometry, and the end surfaces should be measured for dissimilarities. An issue related to SpaceClaim in ANSYS 19.2 is a bug where the `equation driven surface` tool is not able to create sinusoidal surfaces with a whole number of periods in two directions greater than one. To work around

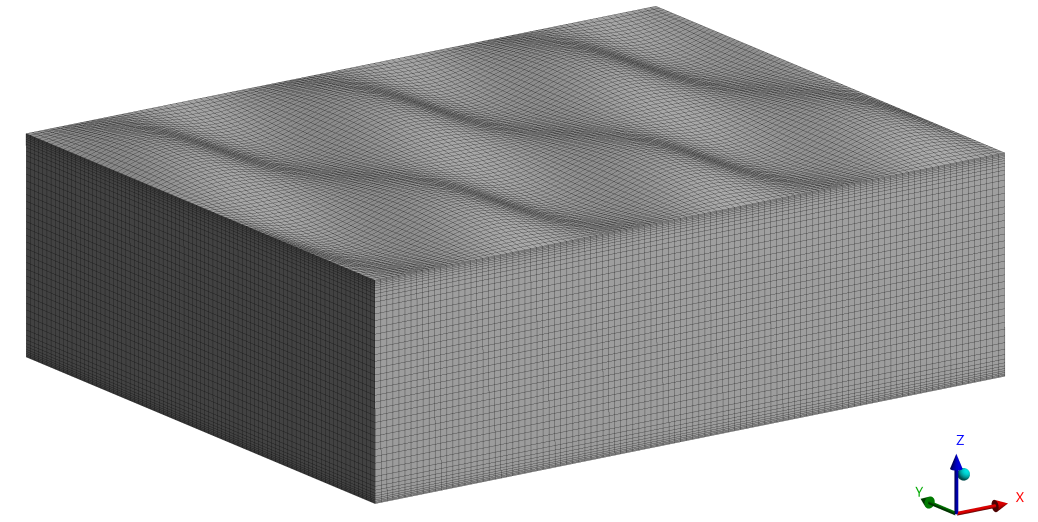
this, the surface is defined with slightly more than the whole number of periods needed, and then cut to size using other CAD functionality. The issue has been reported, so this is hopefully fixed for later versions of the software. The last step is to form a volume in between the surfaces. In SpaceClaim, this can be done using the `Blend` tool.

4.4 Discretization

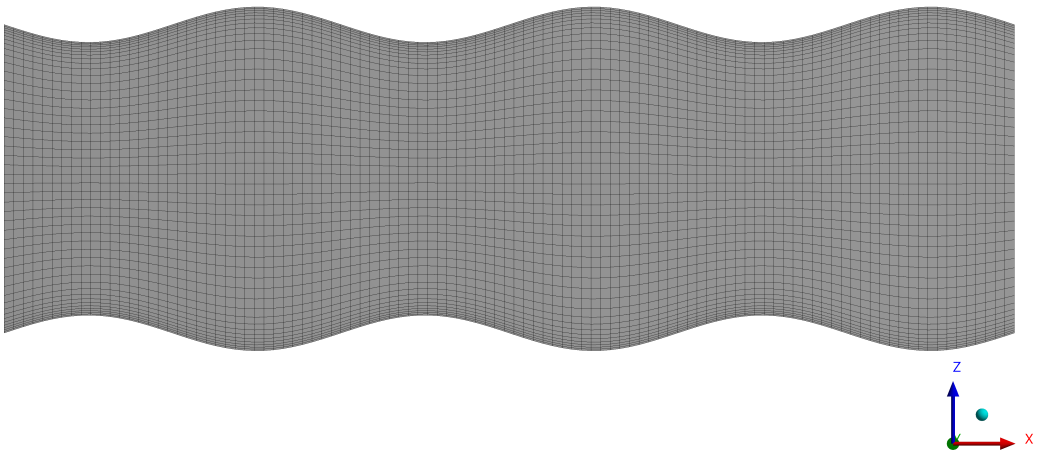
4.4.1 Creating the mesh

One third of the mesh used to discretize the fluid domain is depicted in figure 4.1a, and it is a structured mesh that conforms to the wavy top and bottom walls. The full mesh consists of two additional identical sections, one on each side in the y direction.

Figure 4.1b shows an arbitrary y normal cross section of the mesh, and how the structured nature of the mesh is present inside the domain as well. Whenever the geometry allows for a structured mesh without creating to much skewness or other nonuniformity, it is generally the preferred mesh to use. Skewness is a measure of how a cell differs from a equilateral cell of equivalent volume, and highly skewed cells can decrease accuracy and destabilise the solution (ANSYS 2009a). Convergence rates and accuracy are generally better for the structured mesh over the unstructured one (Ali, Tucker, and Shahpar 2017). This is because a structured mesh generally produces cells with better quality, as long as the geometry is sufficiently regular, but also due to a natural alignment of the cells with respect to the flow. Additionally, the cell placement with respect to each other is inherent in the structured mesh, whereas for the unstructured mesh, the cell connectivity needs to be explicitly specified, taking up additional memory usage (Thompson, Soni, and Weatherill 1998). The principal difference between the two mesh types is illustrated in figure 4.2. Where the unstructured mesh prevails, is on complex geometries, due to its high geometrical flexibility. This allows for the mesh to conform to complex and irregular boundaries, where a structured mesh could fail to generate or create cells with poor quality. The unstructured mesh also has the ability to concentrate the cells where needed, and relaxing the mesh density in less sensitive regions (Luettich 2018). As the geometry in the present study is relatively simple, with smoothly changing boundaries, the benefits of using a structured mesh are considered superior.



(a)



(b)

Figure 4.1: (a) Middle part of the mesh used in the simulations, (b) Arbitrary y normal cross section of the mesh

The mesh is generated using ANSYS Meshing. Inherent compatibility between the CAD tool and the meshing tool, where the two are coupled together inside the ANSYS Workbench environment, is the main reason for choosing ANSYS Meshing as the mesh generation tool. The mesh is generated as a structured mesh using the `Multizone` method, with all hexahedral cells. Inflation layers are inserted at both boundary walls in order to capture the velocity gradients at the boundaries. A smooth transition between the inflation layer cell size and the interior domain cell size is ensured, as abrupt changes in cell size may lead to inconsistencies and therefore large local errors in the solution (Müller 2018). A growth rate of 1.2 is used throughout the inflation layers. Figure 4.3 gives a visual representation of the inflation layer size, and how it blends with the interior parts of the domain. Since the fluid domain is modelled as periodic in

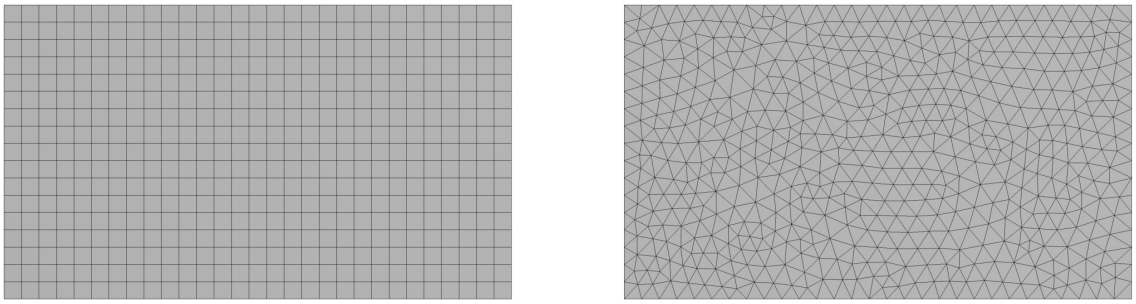


Figure 4.2: Example of a) structured and b) unstructured mesh on a simple 2D surface

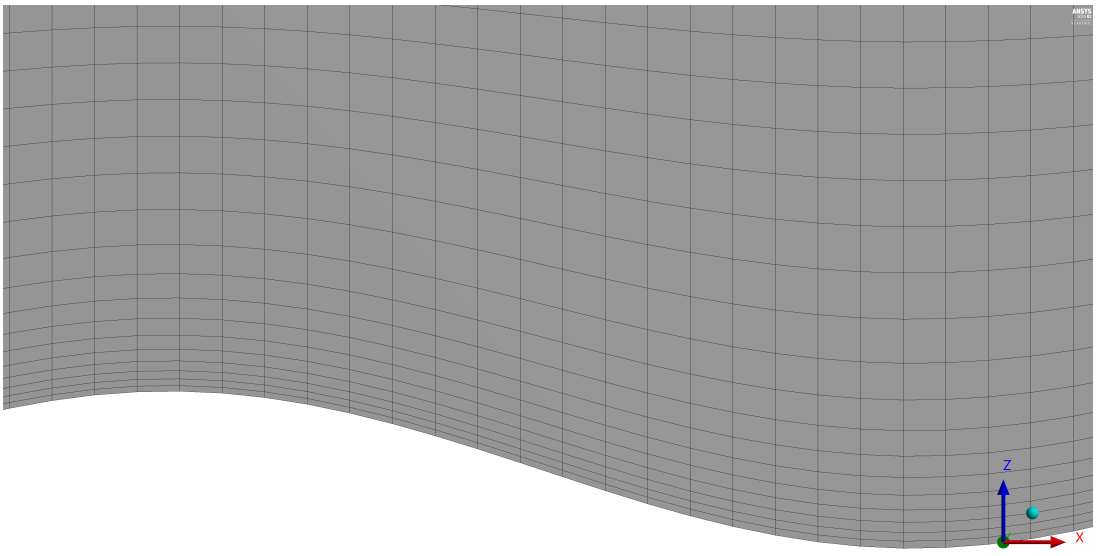


Figure 4.3: Inflation layer at the walls

both the streamwise and transverse directions, it is important that the surface mesh of the two face pairs match each other. This means that the mesh on the inlet side must match that on the outlet side, within a tolerance, and similarly for the two other sides. Small differences in the surface mesh on the face pairs due to numerical errors in the geometry or in the mesh generation, can become significant, especially when the cells become small. The `Match control` option is utilised to ensure proper coupling of the mesh between periodic faces, by slightly altering the cell nodes on one side. Problems occur when trying to specify this option for several faces that share an edge. Therefore, this option is used for the inlet and outlet sides of the domain, as this is the most dominant direction in terms of the dynamics of the system. In addition, the domain is specified with an extra period of the wavy walls compared to the domain to be analysed, in both positive and negative transverse direction. This is done to hopefully reduce possible forcing effects on the dynamics from the periodic boundary conditions. Measures are anyhow taken in OpenFOAM to ensure a correct coupling between these faces as well, and will

be discussed in the section on boundary and initial conditions.

The finished mesh is exported as a `.msh` file, and the work done in ANSYS is now completed. Further analysis continues within OpenFOAM. A built in function named `fluentMeshToFoam` is used to import the mesh from the `.msh` file. It is noted that the mesh is now stored as if it was an unstructured mesh, meaning all neighbours are specified for each cell. This means that the benefit of a structured mesh in terms of cell storage is lost, but the other advantages discussed still remain. When the mesh is imported into OpenFOAM, the correct boundary type for each surface patch must be set in the `boundary` file located in the `constant/polyMesh` directory, such that OpenFOAM can interpret the boundary conditions that will be specified for pressure and velocity. OpenFOAM has a built in function for evaluating mesh quality. It is called `checkMesh`, and this utility is run on all meshes used in the thesis work. The function outputs statistics such as cell and face count, and types of cells used in the mesh. Individual quality parameters such as max cell aspect ratio and skewness is also reported, giving an indication to the user whether the mesh is acceptable in terms of the quantitative measures, or if more adjustments should done to improve the mesh quality. The aspect ratio is the ratio between the longest and shortest side length of a cell, and should generally be kept as close to one as possible in the bulk of the flow. Near the wall, where the flow is more or less parallel to the wall, a higher aspect ratio is tolerated. This is needed to resolve the boundary layer. The `checkMesh` utility also checks the correspondence of the surface mesh on faces coupled together through periodic boundary conditions. A failed mesh check is not tolerated for any of the meshes used, as a measure to ensure good quality meshes.

4.4.2 Mesh refinement and temporal convergence study

Several sources of errors are present when performing CFD analysis. (Slater 2008b) classifies the errors into two main categories, which is acknowledged and unacknowledged errors. Acknowledged errors include physical approximation error which refer to the correctness of the physical model equations used to describe the system, computer round-off error due to finite number of decimals in floating points, iterative convergence error as the linear system solvers use iterative schemes which need some stopping criterion, and finally, discretization error resulting from the discretization of the flow problem in space and time. Unacknowledged errors include programming errors and user errors. Programming errors are of most relevance if the code is self-written or insufficiently tested, and user errors are reduced through experience and training. Of the acknowledged errors, discretization errors are of most concern for a CFD user for an application, as the solution often can show significantly wrong result due to poor mesh quality or time step for transient simulations (Slater 2008b). This is especially true for a code that is widely used. Because of this, steps are taken to ensure that the mesh, together with the solver and schemes used, produce mesh independent results to an acceptable level. This is done

by conducting a mesh refinement study. The study is divided into three parts, where one is for the steady state problem, and two are for the transient problem. Together, these three studies form the foundation for the choice of mesh and time step used for the analysis.

Table 4.1: Mesh parameters used for the mesh refinement study, where Δh is the inner domain cell size which is the same in all three directions, Δh_{min} is the first cell height, and n is the number of cells normal to the wall in the inflation layers.

Mesh name	$\Delta h/H$	$\Delta h_{\text{min}}/H$	n
Coarse	0.0906	0.0375	5
Medium coarse	0.0781	0.0156	9
Medium	0.0469	0.0094	9
Medium fine	0.0313	0.0047	11
Fine	0.0167	0.0022	13

Steady state mesh refinement

Five different meshes are considered in the study, ranging from coarse to fine. Mesh parameters used for the different meshes can be seen in table 4.1. The computational domain is restricted to a single period of the wavy walls in both directions, reducing the computational cost of conducting the mesh refinement study. An amplitude height $a = 0.0625H$, a wave angle $\theta = \pi/8$ and the magnitude of the nondimensional wall wave vector $|\mathbf{k}^*| = 2\,\pi$ is used to fully define the domain. This corresponds to $\alpha_x = 0.39$ and $\alpha_y = 0.15$. The procedure will be to first run simulations for a nominal $Re_\tau = 25$, for which a steady state solution with formed Langmuir cells has been shown to exist (Akselsen and Ellingsen 2019; Brostrøm 2019). This is done for two main reasons. Firstly, a minimum requirement of the mesh used should be that the simpler, steady state solution converges. Secondly, doing this provides an initial velocity and pressure field for further transient convergence analysis. Velocity profiles, wall forces and mass flow rates are extracted from the data and compared for the different meshes.

Figure 4.4a shows the relative difference between the mass flow rate into the domain for the first four meshes, compared to that of the finest one from table 4.1. Apart from a small increase for the medium coarse mesh, the flow rate converges for increased mesh refinement, down to an error of 0.1% for the medium fine mesh. In figure 4.4b, the difference between the flow rate into and out of the domain is presented. These values are equal down to the sixth decimal values extracted from OpenFOAM for all meshes. This is an expected, but also reassuring result, as the faces are coupled together with periodic boundary conditions and therefore should experience the exact same flow rate.

Figure 4.5a and 4.5b show the relative differences between the viscous wall force F_{v} and pressure wall force F_{p} for the first four meshes, compared to the fine mesh. Both figures indicate

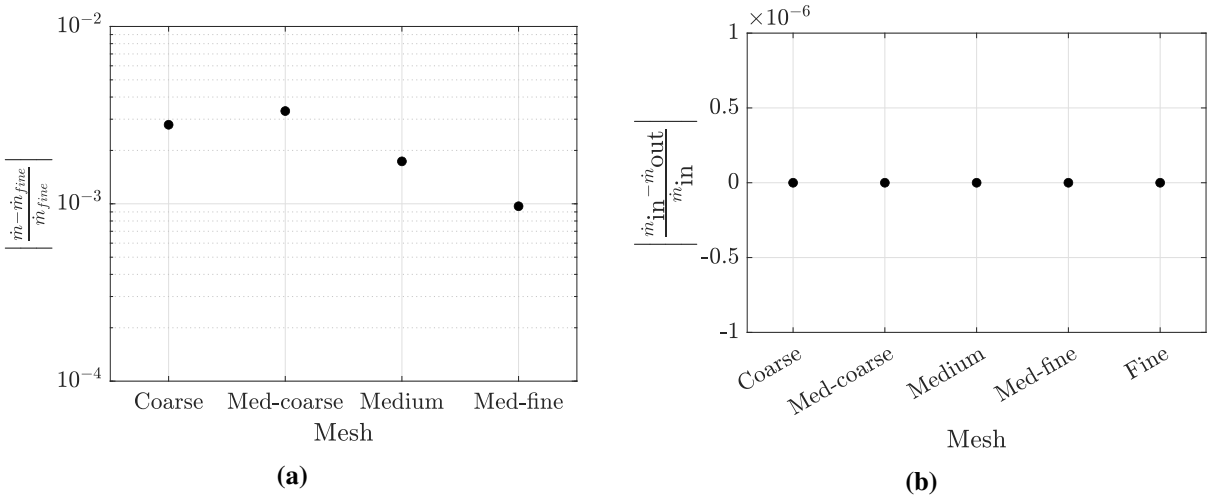


Figure 4.4: Steady state: (a) Relative difference between flow rate into the domain for the first four meshes compared to the finest, (b) Relative difference in mass flow rate in and out of the domain in the x direction for all meshes.

converging values for both quantities, and the medium fine mesh has a relative error of 0.05% and 0.4% for the viscous and pressure forces respectively, compared to the fine mesh. In figure 4.5c, the relative difference between the total wall forces and the force on the fluid from the momentum source term is presented. This value represents the error of the force balance on the flow in the steady state solution. This value is actually increasing when the mesh is refined, except for a significantly lower error for the medium coarse mesh. The reason for this increase is not known, but could be caused by inaccuracies in the way OpenFOAM extracts the wall forces, or maybe due to round-off error in the momentum source term. However, this error is lower than 0.1% for all meshes, and therefore accepted.

Velocity profiles are probed over lines in different regions of the domain for all meshes, and the results are presented in figures 4.6. The normal velocities to three different line probes are shown; two lines are in the z direction and one line is in the y direction. By inspection of the plots, it is concluded that at least the medium mesh is needed to capture the velocity profiles. This is especially clear when looking at figure 4.6b where the two coarser meshes exhibits oscillatory behaviour. Figure 4.6e shows that the qualitative shape of the velocity profile is significantly wrong for the two coarsest meshes. Velocity profiles on two additional line probes are extracted, but as they show very similar behaviour, they are left out of the study, and included in appendix C.1 and C.2. Both the medium and the medium fine meshes seem to produce acceptable results in terms of convergence. Since sufficiently large computational resources are available, the finer of the two is chosen for further analysis, and it is indicated with a red solid line in all plots.

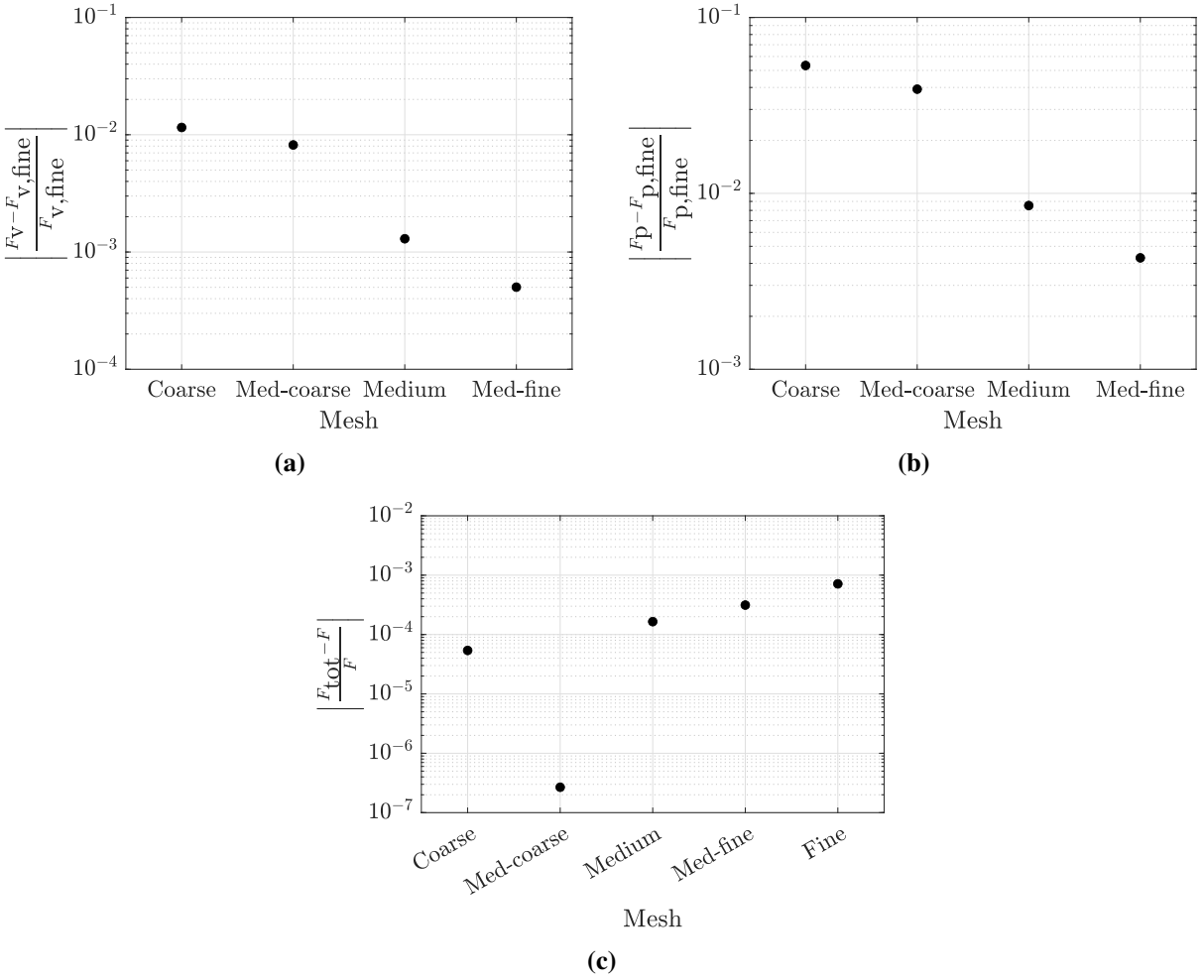
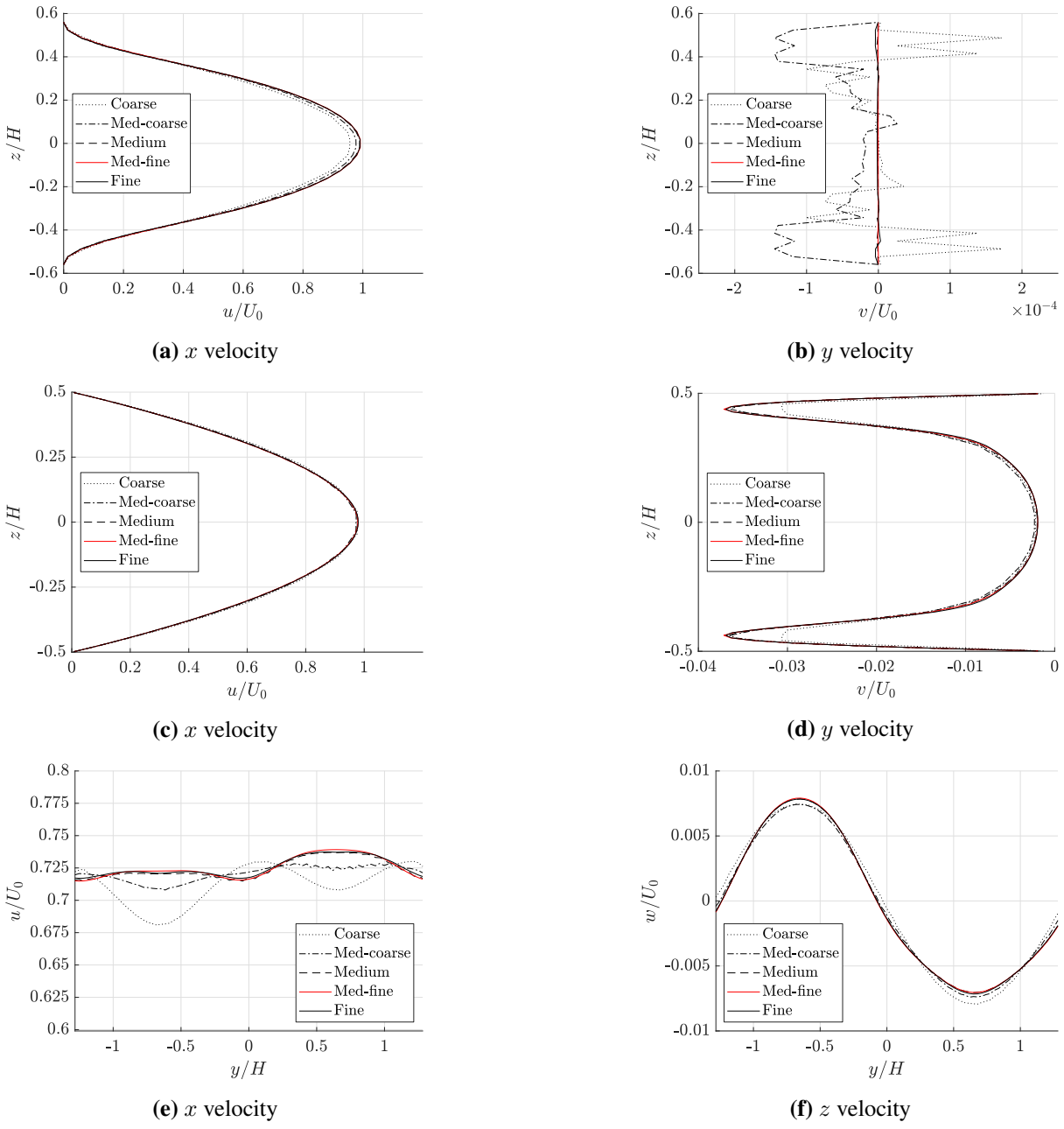


Figure 4.5: Steady state: (a) Relative difference between viscous wall force F_{v} for the first four meshes compared to the finest, (b) Relative difference between pressure wall force F_{p} for the first four meshes compared to the finest, (c) Relative difference between the the total wall forces F_{tot} and the force F on the fluid from the source term.



(a) x velocity

(b) y velocity

(c) x velocity

(d) y velocity

(e) x velocity

(f) z velocity

Figure 4.6: Steady state: Probed steady velocity profiles in different parts of the domain: (a) & (b) Vertical line in the z direction from wall to wall in between wave troughs, (c) & (d) Vertical line in the z direction from wall to wall at $x = y = 0$, i.e. in between saddle points, (e) & (f) Horizontal line in the y-direction at $x = 0$ and $z/H = 0.25$, i.e. half way in between the centre plane and upper the mean wall location.

Transient mesh study for different meshes and equal time step

The steady state solutions from the previous study are used as the initial conditions for transient simulations with a reduced ν on the same five meshes, running to approximately $t^* = 17$, where $t^* = tU_0/H$. During this time, the simulation is able to capture some dynamic effects as the flow adjusts to the sudden change in Reynolds number. The same time step of $dt^* = 0.009$ is used for all meshes, and the maximum Courant number C_{max} for the finest mesh is measured to be 0.53. The Courant number is defined as

$$C = \frac{\delta t \, |\mathbf{u}|}{\delta x}, \tag{4.1}$$

where δt is the time step, $|\mathbf{u}|$ is the velocity magnitude in the cell, and δx is the cell size in the direction of the velocity.

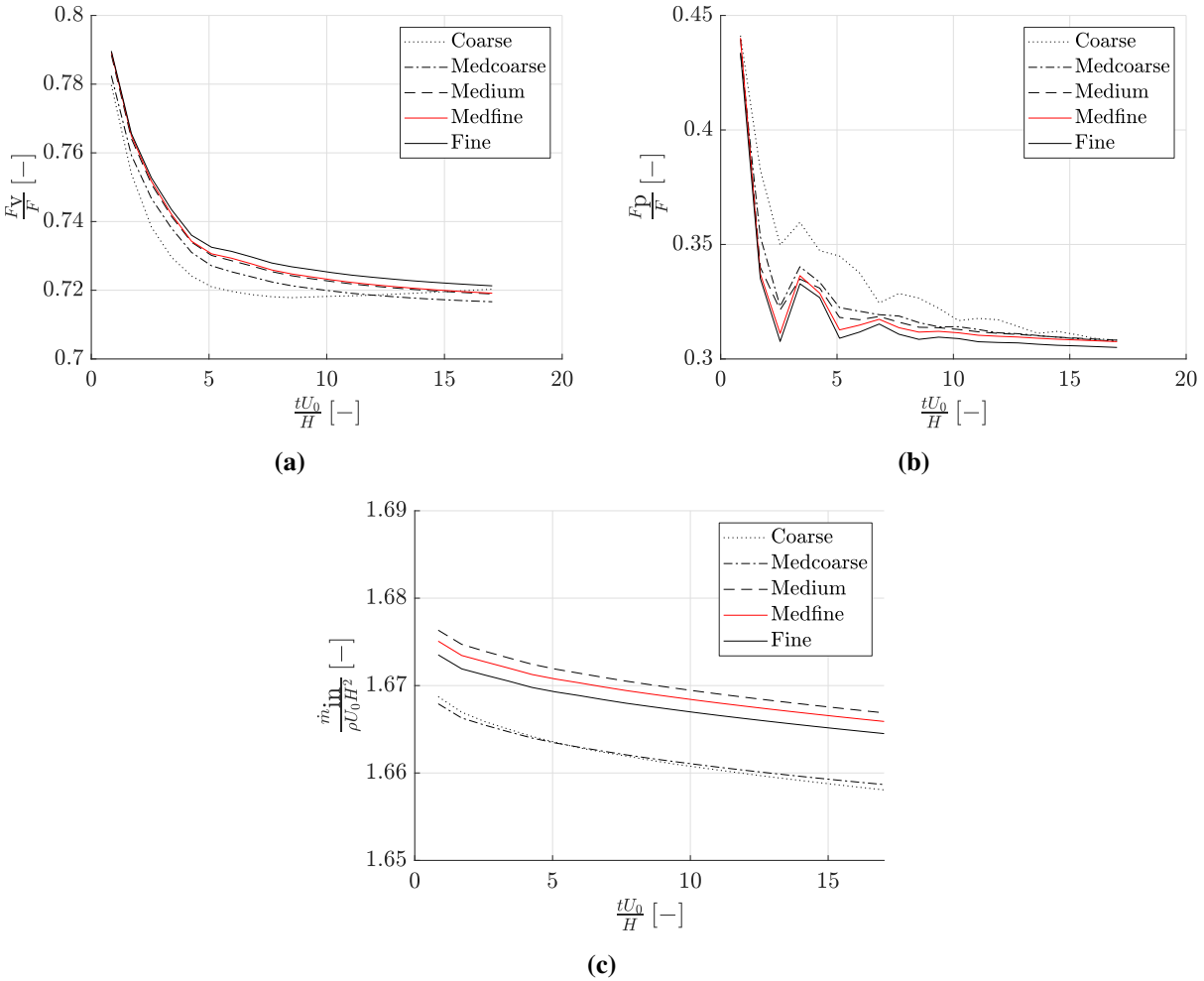


Figure 4.7: Equal time step: Transient development of (a) Viscous wall force F_v and (b) Pressure wall force F_p for the different meshes. Both F_v and F_p are made dimensionless with the total momentum forcing term F. (c) Transient development of mass flow rate $\dot{m}_\text{in}$ into the domain, made dimensionless with $\rho U_0 H^2$.

It is a measure of how far the fluid travels locally within a time step, compared to the mesh cell size. In general, $C < 1$ is needed for temporal accuracy. For explicit time schemes, a sufficiently small C, often $C < 1$ is a requirement for stability, and therefore convergence of a transient simulation. Velocity probes and integral quantities are studied over time and compared between the different meshes.

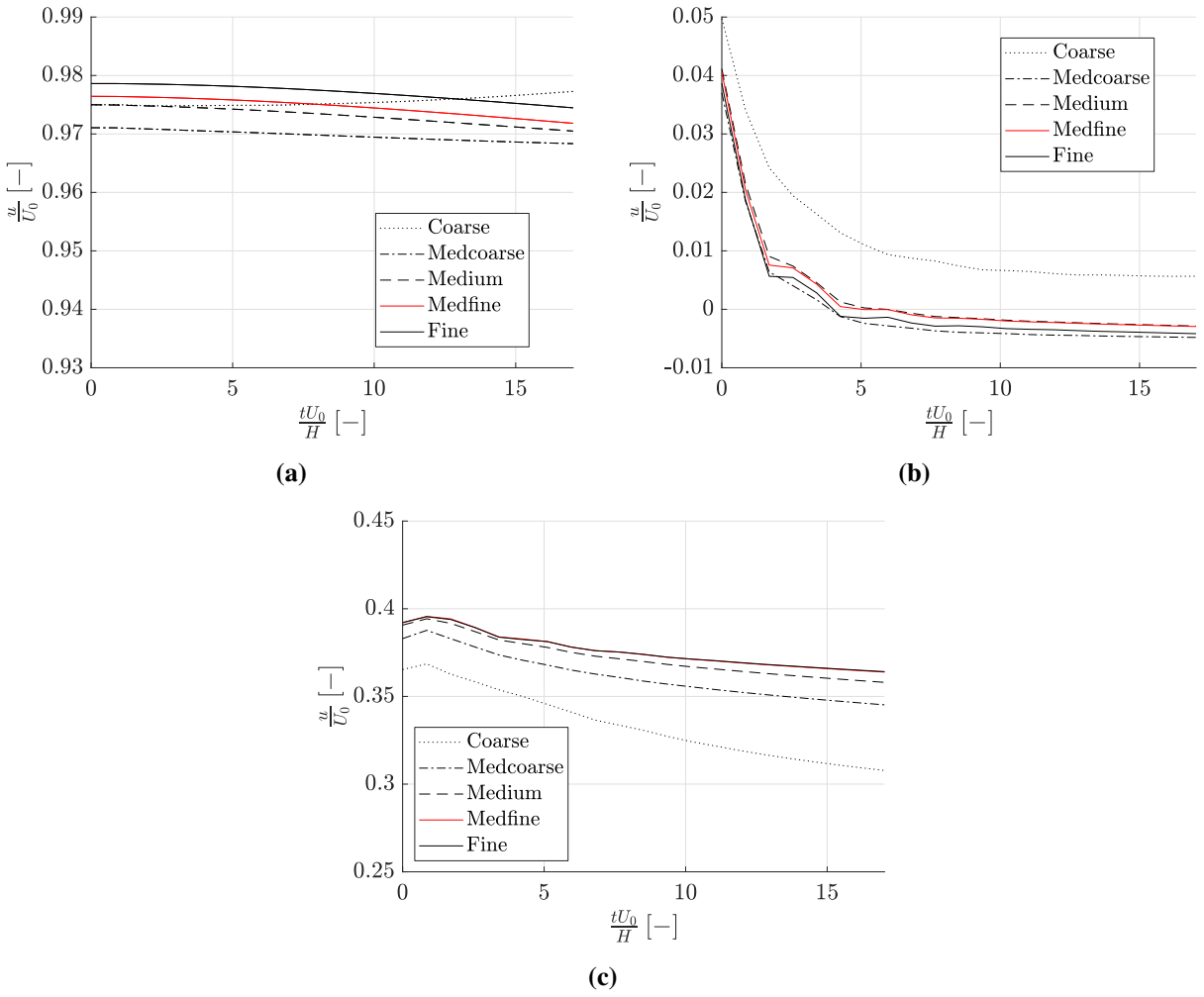
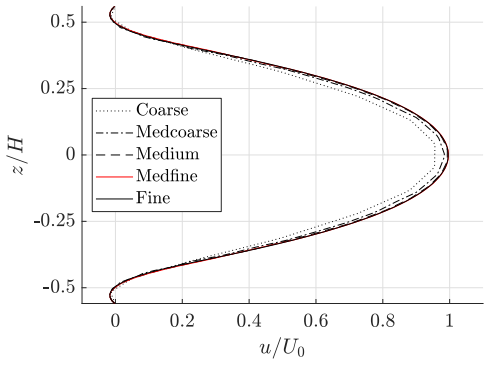


Figure 4.8: Equal time step: Probed for streamwise velocity placed at different locations in the domain, at (a) $x = y = z = 0$, which is above a saddle point, (b) $x/H = 0.2734$, $y/H = -0.6563$ and $z/H = 0.5$, which is located a wall amplitude a below the top wall in a trough, (c) $x/H = 0.2734$, $y/H = 0.6563$ and $z/H = 0.375$, which is located a wall amplitude a below the top wall on a peak

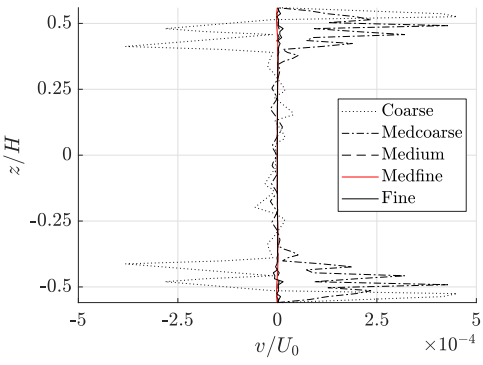
From figure 4.7, the initial development of the wall forces and mass flow rate is presented. As the initial conditions contain the steady state error from the previous study, some quantitative differences between the curves are expected. Nevertheless, the medium fine mesh, indicated with a solid red line, manages to produce the qualitative shape of the fine mesh results well, while also keeping the numerical error between them comparable to the initial error as time advances. Figure 4.8 shows the streamwise velocity values probed at points of interest inside the domain, i.e. in between saddle points, and near peaks and troughs of the wall boundaries.

The plots give similar indications as figure 4.7, i.e. that even though some small initial errors from the steady simulations are present, the medium fine mesh is able to capture the shape of the fine mesh curve well, without the error increasing noticeably.

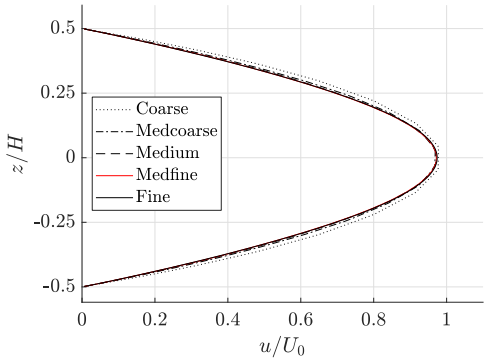
Figure 4.9 shows the final time step of the velocity profiles at the same line probes from the steady state simulation. Again, the solution on the medium fine mesh appears to coincide well with that of the finest mesh for all velocity profiles. It is noted that the top and bottom parts of the x velocity profile in figure 4.9a have become negative. Also the probed value of streamwise velocity in figure 4.8b has become negative during the simulation. Both of these regions correspond to a trough in wavy wall. A negative value of streamwise velocity, indicates that a recirculation zone has emerged in the troughs, caused by separation of the flow over the wall peaks. This is reasonable when the Reynolds number is increased. Based on the results from the two conducted mesh refinement studies, the medium fine mesh will be chosen for further analysis of the flow problem.



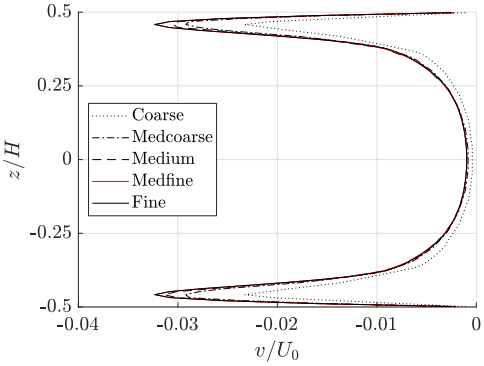
(a) x velocity



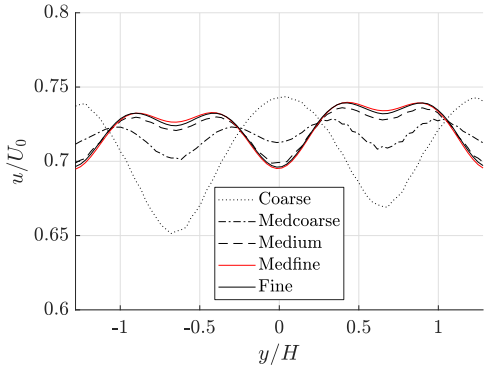
(b) y velocity



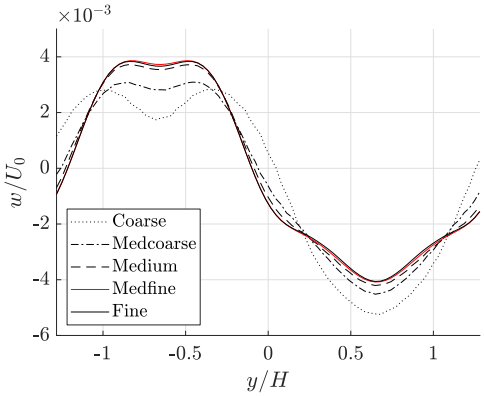
(c) x velocity



(d) y velocity



(e) x velocity



(f) z velocity

Figure 4.9: Equal time step: Probed velocity profiles in different parts of the domain at final time step: (a) & (b) Vertical line in the z direction from wall to wall in between wave troughs, (c) & (d) Vertical line in the z direction from wall to wall at $x = y = 0$, i.e. in between saddle points, (e) & (f) Horizontal line in the y-direction at $x = 0$ and $z/H = 0.25$, i.e. half way in between the centre plane and the upper mean wall location.

Temporal convergence study for equal mesh and different time steps

A new transient study using the concluded converged mesh from the two previous studies is carried out with different time steps. This is done in order to determine a reasonable time step for the further simulations, that is fine enough to preserve temporal accuracy of the solution, without being so fine that computational resources are being wasted. The time steps used, and the corresponding average and maximum Courant numbers for the study, are found in table 4.2.

Table 4.2: Time steps dt^*, and the corresponding average and maximum Courant numbers C_{avg} and C_{max}, used for the time step variation study.

dt^*	C_{avg}	C_{max}
0.085	Not converged	Not converged
0.034	0.7	1.09
0.017	0.35	0.55
0.009	0.18	0.27
0.004	0.09	0.14

Figures 4.10a and 4.10b of the viscous wall force and pressure wall force, respectively, provides a convincing indication that the solution have reached temporal convergence for the time step of $dt^* = 0.034$, which is highlighted using a solid red line. A finer time step beyond this is not producing any higher resolved results, and are therefore considered inefficient use of computational resources. A similar observation is made looking at the mass flow rate development in figure 4.10c. The time step of $dt^* = 0.085$ is seen to produce results outside of the plot window, or with severe oscillations in the solutions, which is reasonable as the simulations has not been able to converge. This time step has simply been too large for the solver to handle, giving unacceptably high values for C. Even though the solution has not blown up to infinity as it often will, all accuracy is lost.

The velocity probe plots in figure 4.11 tells a similar story, where no change in the x velocity curves is observed when reducing the time step beyond $dt^* = 0.034$. Also here, the non-converged solution for the coarsest time step is visible in the plots.

Also for this study, the velocity profiles are probed in the last times step. Only plots from one of the line probes are included, as all of them show the exact same tendencies, which is that all curves are equal when the time step is $dt^* = 0.034$ smaller. All the remaining curves are included in appendix C.3.

Looking at table 4.2, the corresponding average and maximum Courant numbers for a time step of $dt^* = 0.034$ are C_{avg} and C_{max} is found. The average value of C for this time step is approximately 0.7, which is well below one. The maximum value of 1.09 is noted, however, it is considered acceptable. This is mostly based on the results from this temporal convergence study, showing that no noticeable difference is observed when lowering the time step further,

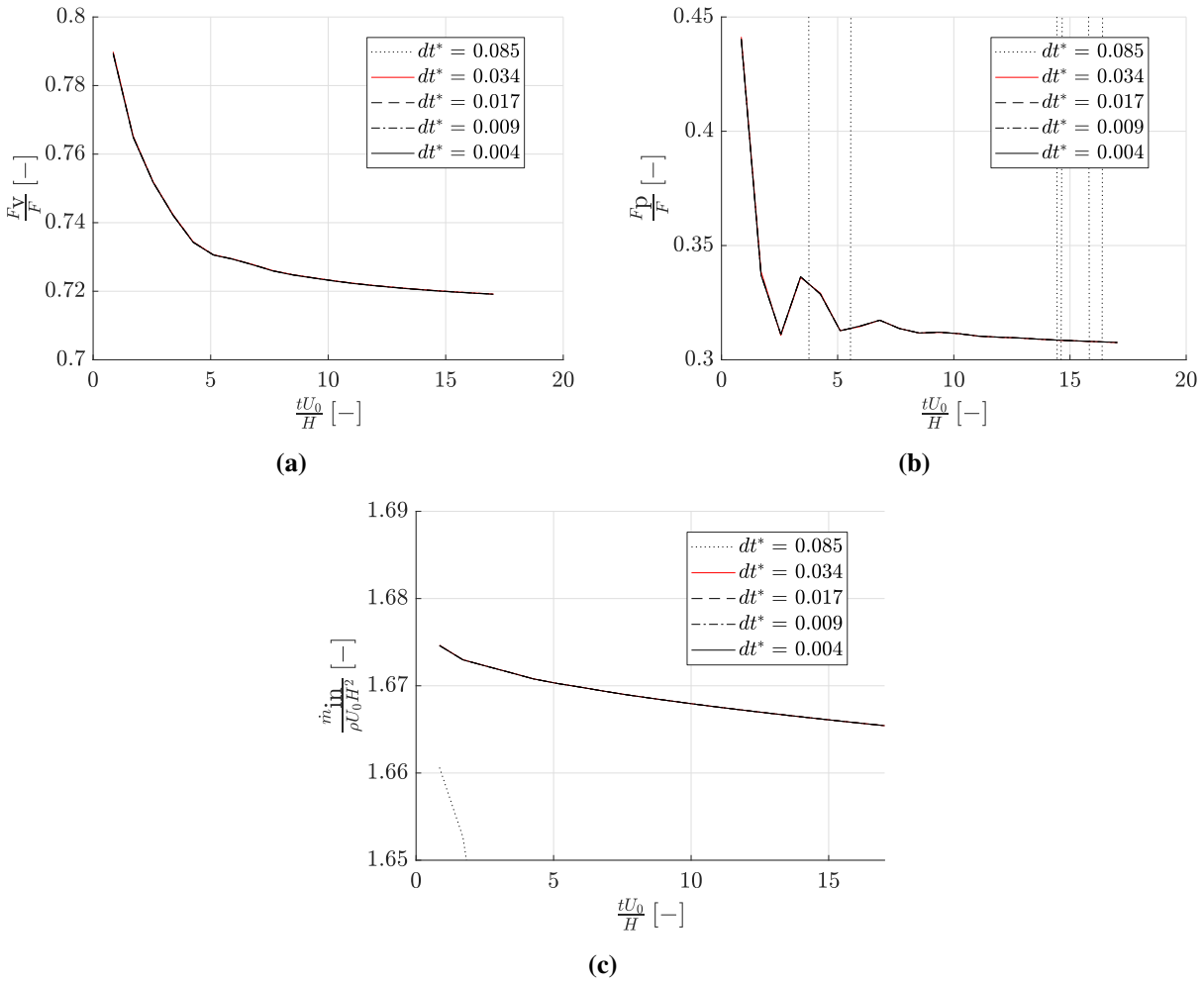


Figure 4.10: Equal mesh: Transient development of (a) Viscous wall force F_v and (b) Pressure wall force F_p for the different time steps given in table 4.2. Both F_v and F_p are made dimensionless with the total momentum forcing term F. (c) Shows the transient development of mass flow rate $\dot{m}_{in}$ into the domain, made dimensionless with $\rho U_0 H^2$.

achieving $C_{max} < 1$. In addition, the mesh cells where $C = C_{max}$ is located in the vicinity of the midplane at $z = 0$, half way between the wavy walls. In this thin region, the flow is almost uniform, changing relatively slow in all directions. It is therefore reasonable to assume little error is caused by a Courant number only slightly higher than one in this region. The effects of the wavy walls are more predominant in the regions away from the midplane, where $C < 1$.

The time step of $dt^* = 0.034$ is concluded as the time step balancing computational cost and temporal accuracy. This means that both the spatial and temporal discretization of the flow problem is determined, meaning the following analysis can be conducted with an increased confidence that the CFD solver is actually solving the flow problem correctly.

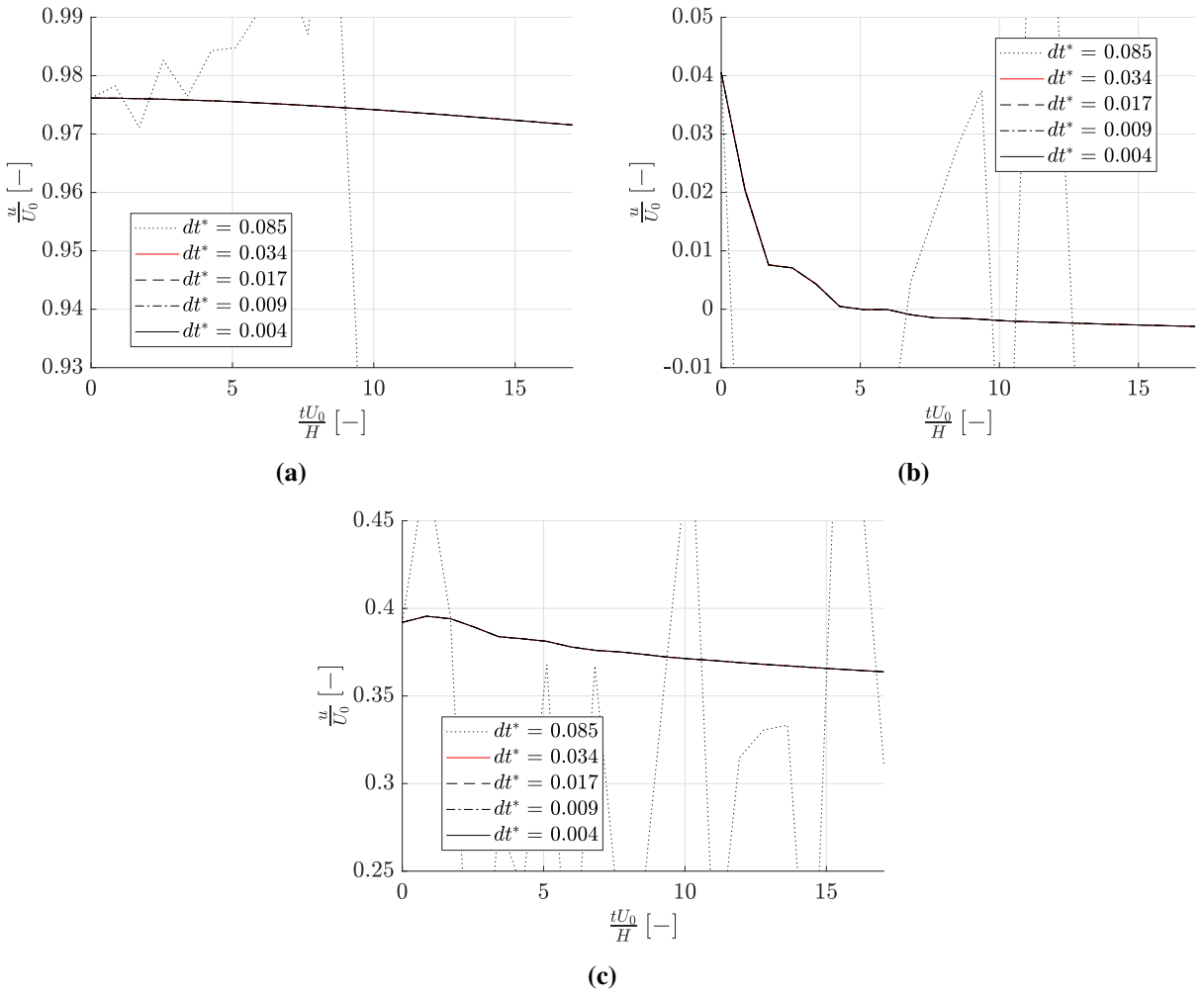
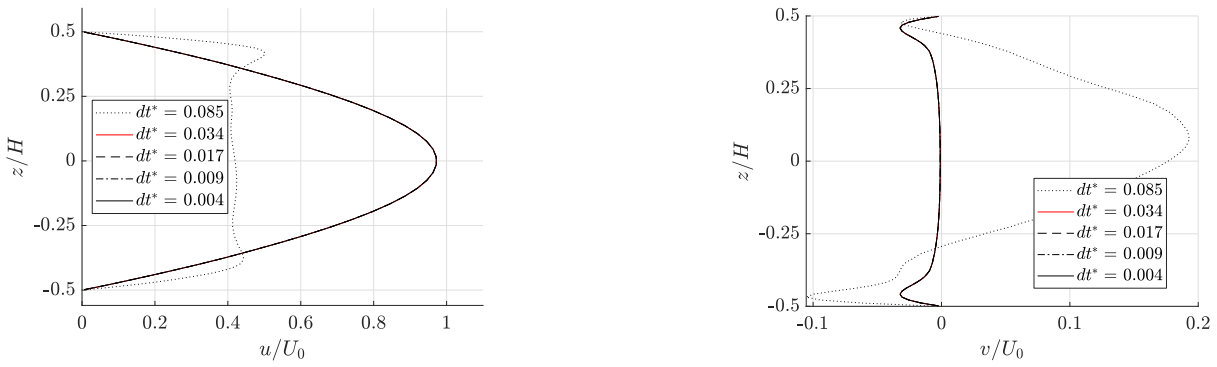


Figure 4.11: Equal mesh: Probes for streamwise velocity placed at different locations in the domain, at (a) $x = y = z = 0$, (b) $x/H = 0.2734$, $y/H = -0.6563$ and $z/H = 0.5$, which is located a wall amplitude a below the top wall in a trough, (c) $x/H = 0.2734$, $y/H = 0.6563$ and $z/H = 0.375$, which is located a wall amplitude a below the top wall on a peak



(a) x velocity

(b) y velocity

Figure 4.12: Equal mesh: Probed velocity profiles at final time step on a vertical line in the z direction from wall to wall at $x = y = 0$, i.e. in between saddle points

4.4.3 Viscous length scale and y-plus

While the flow is changing from case to case when the Reynolds number and domain is altered, the same dimensional mesh parameters is still used for all cases. It is therefore useful to define a new length scale to nondimensionalise the mesh cell sizes based on flow properties for the individual cases, and not the channel height H, which is kept constant for all cases. By doing this, it is possible to say something about how well the flow in the individual cases is resolved. This new length scale is defined as

$$l^+ = \frac{\nu}{u_*}, \tag{4.2}$$

and it is called the viscous length scale. It is used to define the dimensionless mesh cell size

$$\Delta h^+ = \frac{\Delta h}{l^+}. \tag{4.3}$$

The quantity l^+ is also used to define another value of interest, namely y^+, given by

$$y^+ = \frac{y_p}{l^+}. \tag{4.4}$$

Here, y_p is the normal distance from the wall to the cell centre of the cell closest to the wall, and it is important to note that it is not related to the y direction in the flow problem. The quantity y^+ is a nondimensional measure of how well the boundary layer is resolved near the wall, and a $y^+ < 1$ is generally required to accomplish this. OpenFOAM has a built in utility to calculate y^+, and it is called `yPlus`. This function provides data for average and maximum values of y^+. Using (2.2), (4.2) and (4.4), results from this output can also be used to determine the true friction Reynolds number in the flow, $Re_{\tau,t}$. To do this, the average value of y^+ is considered, and y_p is half of the known minimum cell height h_{min} in the inflation layer. As previously discussed, this number will be lower than the nominal friction Reynolds number Re_τ for a laminar flow.

4.5 Case setup

4.5.1 Building a new case

The conventional way of setting up a new case in OpenFOAM, is to copy a folder from a tutorial case provided in the installation files. The tutorial case may or may not be similar to the new case, the most important is to get a set of the necessary configuration files that can be altered to describe the new case.

4.5.2 Boundary and initial conditions

The boundary condition prescribed at the top and bottom wavy walls is a no-slip wall condition. No-slip is necessary to develop the boundary layer required for the mechanism driving the Langmuir Circulations. The no-slip boundary condition forces all the velocity components to $u = v = w = 0$ at the boundary. For pressure, a zero gradient boundary condition is set normal to the walls, i.e., $\partial p/\partial n = 0$. All the other boundary conditions on the domain surfaces are given as periodic, i.e. during the simulation, the streamwise-normal faces are considered the same, and similarly for the transverse-normal faces. For the streamwise-normal faces, this is implemented with the `cyclic` option in OpenFOAM, which is the standard periodic boundary condition. For the transverse-normal face, `cyclicAMI` is used. The AMI stands for arbitrary mesh interface, and using this option allows for a mismatch in the surface meshing between two faces. Instead of having a perfect coupling between the faces, the problem is solved by mapping the data from on face onto the other, using weighted interpolation. The mismatch between the faces on the mesh in this study is considered small, so this is accepted as a solution to the problem.

Using periodic boundary conditions allows for considering the domain as infinitely long in both streamwise and transverse direction. The alternative to this approach would be to consider a long and wide enough domain, prescribe an inlet velocity corresponding to the specified Reynolds number, and study the evolution of the circulation cells at different streamwise locations in the domain. Initial simulations indicate that the dynamics of the instability is slow, and that the flow could have to travel trough the periodic domain on the order of ~ 100 times before the circulation currents properly breaks down, depending on the flow parameters. Using inlet and outlet conditions for the flow is therefore not considered a viable option for solving the problem. When using periodic boundary conditions in a simulation, the fact that the periodicity may enforce unnatural behaviour of the flow must be recognised. In the project work (Brostrøm 2019), only one period of the wavy wall boundaries was utilised, as the flow was steady and the Reynolds number was low. In this study, an extra wave period of the boundary walls are used in both directions to account for the increased Reynolds numbers and transient effects, and to attempt to minimise potential forcing effects on flow structures larger than one period from the periodic boundaries.

The initial conditions used for studying the transient development is set as the velocity field from a steady simulation with a nominal $Re_\tau = 25$, where the Langmuir Circulations are shown to exist (Akselsen and Ellingsen 2019; Brostrøm 2019). This is done using a steady state solver. The initial condition for the steady state simulation is set to the expected mean velocity in the streamwise direction, and to zero for the other velocity components and pressure. This solution is transferred into a transient case as the initial condition, using a built in function called `mapFields`. This case is running the transient solver. The boundary conditions for

both the steady state and transient simulations, and the initial conditions for the steady state simulation are set in the files `p` and `U` for pressure and velocity respectively, located in the `/0` directory. A higher Re_τ for the transient simulation is set by altering the ν and F, and it is done in a way such that the maximum streamwise velocity U_0 is roughly the same in the steady and in the transient case. This is done such that sudden initial changes to the velocity field is kept to a minimum when the transient simulation begins. An alternative to doing this, which would be more physically correct, would be to ramp up the velocity profile or to ramp down the viscosity. The reason for not doing this is due to the especially slow change in dynamics of the system, and initial test simulations give indications of long simulation times for the transient effects to develop. Simulation times would be even longer if the flow first would have to gradually change to a new Reynolds number. Modification of ν is done in the `transportProperties` file. Implementation of F is somewhat more involved, and will be discussed later in the chapter.

4.5.3 Solver

The solver used to obtain the initial conditions for the transient simulations is called `simpleFoam`. It is a steady state solver for incompressible, turbulent flow, and uses the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) algorithm (OpenFOAMWiki 2019). An option is included to use the consistent version of the algorithm, namely SIMPLEC, which generally allows for higher under-relaxation factors than SIMPLE, and therefore faster convergence rates (ANSYS 2009b) (26.3.1). For the transient solver, `pimpleFoam` is used. This is also a solver for incompressible, turbulent flow, but for transient problems. The algorithm is a combination of the PISO (Pressure-Implicit with Splitting of Operators) and the SIMPLE method. PISO divides the operators into an implicit predictor and multiple explicit corrector steps. The scheme is not considered as iterative, and very few corrector steps are necessary to obtain desired accuracy (OpenFOAMWiki 2014). It is noted that even though both the steady state and the transient solvers are made to deal with turbulence models, no models are used by setting the `simulationType` option in `turbulenceProperties` to `laminar`.

The settings for the discretization schemes are set in the `fvSchemes` file, and all spatial schemes are set to second order accurate methods. For the time discretization, the second order backward differentiation formula (BDF2) is used. Second order accurate schemes are chosen to increase the accuracy of the solutions for a given time step and cell size, as first order schemes can be highly dissipative in the presence of strong gradients (Kukudzhanov and Zhurov 2013). The convergence criterion for the residuals of the solvers is set in the `fvSolution` file. Residuals for final values of p and U for every time step are set to 10^{-6}. Linear solvers for p and U is also set in this file. For pressure, a geometric algebraic multigrid (GAMG) method is used, and for velocity, the symmetric Gauss Seidel method is chosen.

4.6 Source term

4.6.1 Implementation of source term

The source term for the streamwise momentum equation F is implemented as an implicit linear source term on the form

$$F(u) = S_u + S_p u, \tag{4.5}$$

where S_u is the explicit part and S_p is the linearised part of the source term. A file named `fvOptions` is added to the `/constant` directory where this linear source term is implemented. Inside the files, the built in function `SemiImplicitSource` is added. In this study, the source term should be constant over the domain, and independently of velocity u. Therefore, the linear term vanishes, i.e., $S_p = 0$.

4.6.2 Verification of source term

A simplified test case is included in order to verify that the source term F is implemented as intended in the CFD software using `SemiImplicitSource`. To do this, a simplified version of the problem is modelled in OpenFOAM, by removing the undulation of the boundary walls. Doing this transforms the problem to a plane Poiseuille flow, where the equations in section 3.3 are exact for a laminar flow. This flow is independent of the transverse coordinate, and can therefore be simplified to two dimensions. Figure 4.13 shows the analytical expression for $U(z)$

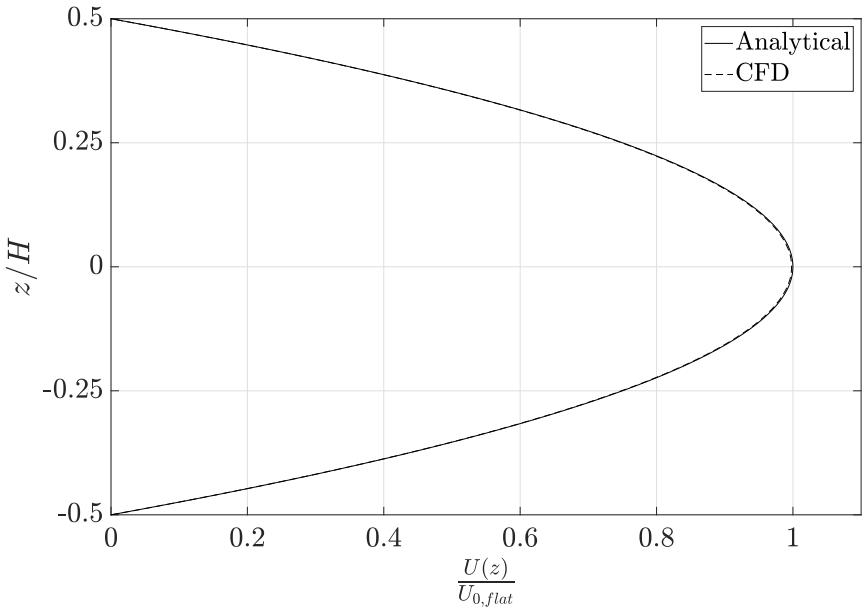


Figure 4.13: Comparison between the analytical solution and the CFD result for Poiseuille flow between flat plates. Both curves are scaled by the analytical centre-plane velocity

given by (3.14), compared to the velocity profile from the numerical simulation. Both results

use the viscosity and momentum source term from (3.19) and (3.20), respectively. The axes are made dimensionless with the analytical centre-plane velocity $U_{0,\text{flat}}$ and the channel height H. As seen from the figure, the velocity profiles coincide on the same curve, indicating that the source term is implemented as intended.

4.7 High performance computing

4.7.1 Running simulations on Vilje

Other than for small test cases, the simulations are carried out on the Vilje high performance computing (HPC) cluster. Vilje is a SGI Altix ICE X system procured by NTNU together with the Norwegian Meteorological Institute (MET) and UNINETT Sigma. It has been used for numerical weather prediction in operational forecasting by MET as well as for research in a broad range of topics at NTNU and other Norwegian universities, colleges and research institutes (NTNU-HPC-Group n.d.). Access to the cluster is provided by Sigma2, who manages the national e-infrastructure for large-scale data- and computational science in Norway (Sigma2 n.d.). Having access to a cluster enables for simulations with a higher number of cells, as the computational domain can be divided between processors and the solution algorithms can be parallelised. In OpenFOAM, the partitioning is done using the `decomposePar` function. How the partitioning should be performed, is specified in the `decomposeParDict` file. In this study, the mesh cells is uniformly distributed in both the x and y direction, so the choice is made to split the domain into uniform parts in these directions. For cases where the mesh is not so easily divided into uniform sections, more advanced methods for the decomposition is available.

Connecting to Vilje is done via the Linux Terminal, using the Secure Shell (`SSH`) protocol, and file transfer to and from Vilje is done using the Secure File Transfer Protocol (`SFTP`). Running jobs on Vilje is done using bash scripts, which is a file containing the sequence of commands that the computer has to execute for the simulations to run. This includes loading the modules needed to run the simulations, e.g. OpenFOAM itself. In this file, the number of processors and maximum run time on Vilje is also specified, amongst some other information about the job and user.

4.7.2 Weak scaling study

When doing parallel computing, one must determine an appropriate number of processor cores to be used for the computations. In addition to the limitation of available cores, the weak scaling of the problem must be considered. The concept of weak scaling explains how the speed-up of the computations for a fixed problem size increases as more processors are added

for the calculations. Weak scaling is described by Amdahl's law (Amdahl 1967), written as

$$S = \frac{1}{(1 - f_p) + \dfrac{f_p}{n_c}}, \tag{4.6}$$

where S is the theoretical speed-up, f_p is the fraction of the computation that can be run in parallel and n_c is the number of cores used. As can be seen from (4.6), the speed-up S is limited by $1/(1 - f_p)$ as the number of cores tends to infinity. Here, $(1 - f_p)$ is the fraction of the computation that is limited by serial calculations. For a CFD problem, weak scaling is generally good, as the majority of the computational time goes towards solving systems of linear equations within each time step. These are calculations that benefit from parallelisation. Implicit time integration schemes leads to increased serial computational fraction, but this is only when going from one time step to the next.

A weak scaling study of the problem at hand is done, where simulations of 500 time steps are performed on the full fluid domain with the converged mesh and time step from the mesh refinement study. Different number of processor cores are used for the different cases. A representative fraction of the time steps are written to file, as this proportion will influence the total speed-up. This is due to the fact that a high or low number of time steps written to file would significantly affect the reconstruction times, creating results biased towards reducing or increasing the number of cores, respectively. The execution times for each simulation are recorded. The number of cores used in the test ranges from 1 to 128, where each increment is a power of two. Results from the weak scaling study is presented in figure 4.14, and it shows the observed speed-up compared to Amdahl's law for different parallelisable fractions f_p. A linear curve is seen for $f_p = 1$, which is the maximal theoretical speed-up for the idealised case where all parts of the computation are parallelisable.

Two different curves are included in the figure in addition to the Amdahl theoretical curves, both plotted in red. The dashed red line represents the speed-up of the simulation when only the solver execution times in OpenFOAM are considered, while the solid red line is for the execution times where the whole job is included. This means that the decomposition and reconstruction of the domain before and after the simulation, which are serial task for the computer, contributes to the speed-up. Both cases are included as they can both provide valuable insight. It can be seen from the figure that both the solver and the total speed-up exhibit the same behaviour for a smaller number of cores, up to about 16. This is sensible, because the program does not have to spend much time decomposing and reconstructing the domain, making the solver time the dominant contribution to the speed-up. For a higher number of cores, the two curves begin to drift apart, as the time taken for decomposition and reconstruction of a large number of subdomains is becoming significant compared to the time used by the solver.

It is of course the total observed speed-up curve that needs to be considered when determin-

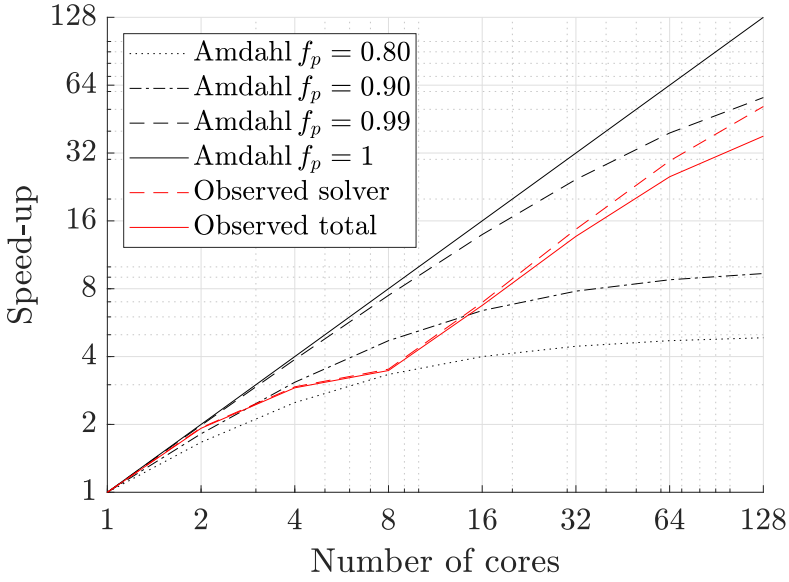


Figure 4.14: Results from weak scaling study, showing observed speed-up compared to Amdahl's law for different f_p

ing the number of processor cores for the simulations, as this is the actual speed-up of program. It is however interesting to study the curve for the solver speed-up as well. Up to eight cores, both observed curves are showing signs of flatting out. A further increase in cores produces close to linear speed-up of the program. This could be because the solver parallelisation in OpenFOAM has not reached its ideal efficiency when there are still a high number of mesh cells in each subdomain. The mesh used contains approximately 1.2 million cells, meaning 150000 per core when using eight cores. Following the solver speed-up curve for higher number of cores, one can see close to linear behaviour up to $n_c = 64$, where the slope starts to decrease. The fact that the curve shows close to linear behaviour, gives indication of a very good weak scaling of the solver in this region. At this point, each core deals with approximately 20000 mesh cells, and the solver is beginning to reach its maximum capacity for efficient decomposition of the domain. For this amount of cells, the actual calculation times are becoming so small that the overhead time, i.e. the time used for communication between the cores, is becoming significant. A further decrease in mesh cells per core would eventually lead to a negative slope of the curve due to this effect, which would be a highly wasteful use of resources. It is important to note that there could be several other factors that influence the speed-up in a real computer, that can also contribute to such deviations.

A choice is made to use $n_c = 64$. There are several reasons for stopping at this number of cores. Further increase in the number of cores requested, would also increase the queuing time to run jobs on the cluster. In addition, the observed speed-up curve is beginning to flatten out. This means that even though the total simulation time goes down by adding more cores, the central processing unit hour (CPU-hour) cost will become higher and higher. The CPU-hours

is a measurement of the total run-time on all the processor cores used in a program combined. Since the users of Vilje are given a certain number CPU-hours to use, this also has to be taken into account.

4.8 Post processing

Two different types of plots are used to visualise the rotational structures in the flow and how they evolve in time. Streamline plots show the qualitative behaviour of the circulation cells, and streamwise vorticity contours gives quantitative values to the rotation of the flow. By studying these plots, an indication of how both the shape and the strength of the rotational cells vary with time is given. Both types of plots are defined by streamwise averaged values, such that the net rotational current through the domain can be studied. This in turn means that the plots can be considered the average cross-sectional (y, z)-plane, i.e. normal to the bulk flow. The averaged v and w velocity fields in the streamwise normal plane is used to compute the streamlines. The velocity and vorticity data from OpenFOAM are exported as `.csv` files in ParaView, a standard, open source post-processing software used along with OpenFOAM. The data is then imported into MATLAB, and interpolated into a 3D Cartesian grid for further processing and plotting. The script for doing the post processing is included in Appendix D. Nodes in the Cartesian grid that fall outside of the fluid domain, i.e. that are "inside" the wall, are made to NaN (not a number), such that averages containing these are excluded from the plots. Vorticity is made dimensionless with the time scale from (3.4), as the unit of vorticity is s^{-1}. The streamline plots are generated using the `streamslice` function in MATLAB. For these plots, the whole width and height of the domain is included, and the outlines of the sinusoidal walls are indicated by dotted lines. For the vorticity plots however, the regions of the domain near the walls are left out of the plots. This is done due to large values of vorticity near the walls compared to that of the circulations, because of strong velocity gradients near the wavy walls. If those regions were to be included, small relative magnitude of vorticity in the circulation cells would reduce the plots ability to properly visualise the rotational dynamics.

4.9 Validation of CFD code

Validation of the CFD model is a crucial step towards acquiring reliable results from a simulation. The concept of validation is to determine if the computational simulation agrees with physical reality. It examines the science in the models through comparison to experimental results (Slater 2008a). Where no experimental data exist, comparison with highly resolved numerical data is also used for validation purposes. For the present study, where the transient development of the Langmuir Circulations are in focus, no published experimental or numeri-

cal data exists at this time. Despite this, steps will be taken in order to validate the CFD code as best as possible. Two simulation cases will be evaluated, where the first is simulating the laminar boundary layer development over a flat plate, and comparing the results with the Blasius solution for a flat plate, as discussed in section 3.2. The Blasius solution data is taken from (Lal and Paul M 2014). Even though this test case does not represent all parts of the flow in this study, it is still a good way of confirming that the code is in fact able to model a boundary layer correctly, which is considered an important element for the mechanism driving the circulation cells. Having a virtually exact solution to the problem is invaluable when trying quantify the code's performance. The second case will be to compare the steady state solution in OpenFOAM for a lower nominal friction Reynolds number of $Re_\tau = 25$, to the results produced by the LBM code in (Akselsen and Ellingsen 2019). Doing this will provide valuable information about the CFD code's ability to capture the flow mechanics behind the Langmuir Circulations. The development of a laminar boundary layer over a flat plate is independent of the transverse

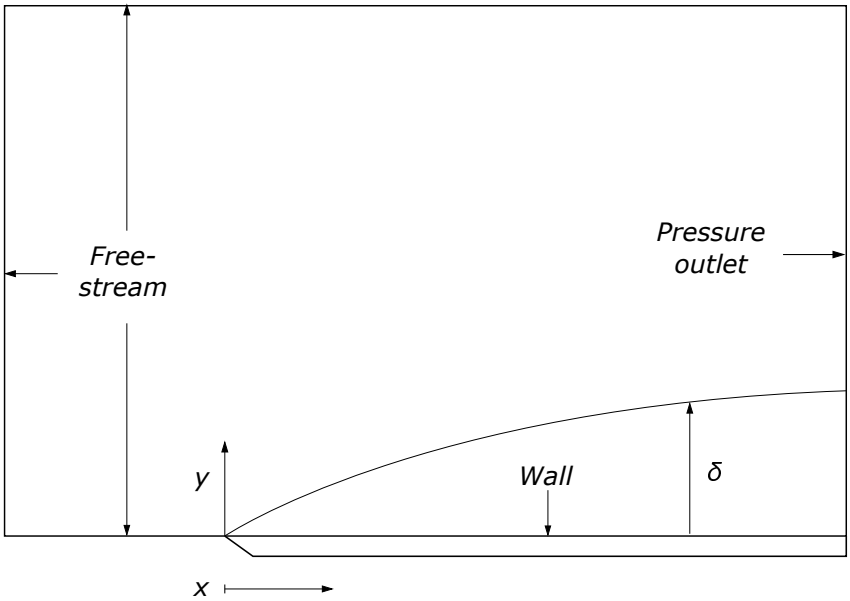


Figure 4.15: Boundary types used for the test case

direction, and therefore a two dimensional model will be made to simulate this flow. Figure 4.15 illustrates how the boundaries in the flat plate test case are defined. For the top, left and bottom boundary up to the leading edge of the plate, the freestream velocity U is prescribed. A Neumann condition is set for pressure, i.e. $\partial P/\partial n = 0$, where n is the normal direction to the boundary. At the wall, a no slip boundary condition is used, setting velocity in both directions to $u = v = 0$. Also here, zero normal gradient is set for pressure. Finally, the right boundary of the domain is specified as a pressure outlet, where $P = 0$ and $\partial u/\partial n = \partial v/\partial n = 0$.

Figure 4.16 shows the mesh used for the flat plate test case, which is a fully structured mesh containing only rectangular cells. The mesh is refined near the wall to capture the velocity gradient in the boundary layer. Additionally, the mesh is refined in the streamwise direction

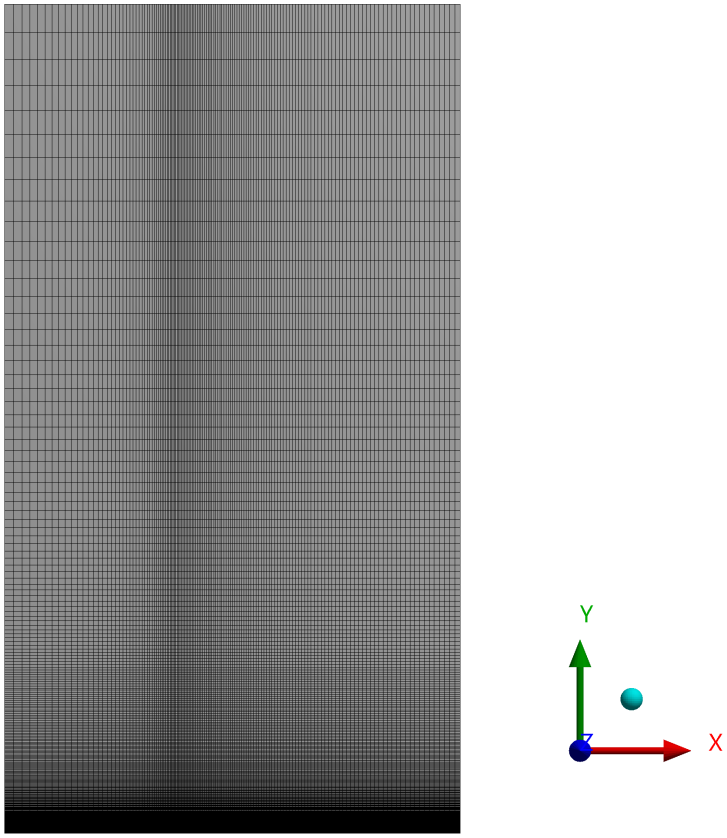


Figure 4.16: Mesh used for the test case

at the leading edge of the plate due to the strong initial gradients where the boundary layer development begins. The domain height is made such that the the minimum value of $\eta \approx 600$ at $x = L$, meaning the boundary layer takes up less than one percent of the domain height, since the boundary layer height is at about $\eta = 5$. This is done to minimise far field effects from the top boundary.

A Reynolds number based on the flat plate length L, freestream velocity U and fluid kinematic viscosity ν, is chosen to be $Re_L = 48679$, i.e. well below the critical Reynolds number $Re_{L,c} \approx 5 \cdot 10^5$. The simulation is conducted using both the steady state and the transient solver. For both cases, the initial condition is the freestream velocity U in the whole domain. Results for the transient simulations are extracted when a steady state solution is reached. Line probes of streamwise velocity u are extracted from the CFD results at four arbitrary streamwise locations x, excluding $x = 0$ where the Blasius solution is invalid. The normal direction axis y is scaled to the similarity variable η, using (3.9) for every x considered. Streamwise velocity profiles of the Blasius solution and from the CFD data with the transient solver are normalised with the freestream velocity U and plotted together in figure 4.17. The steady state solution is identical that of the transient solver..

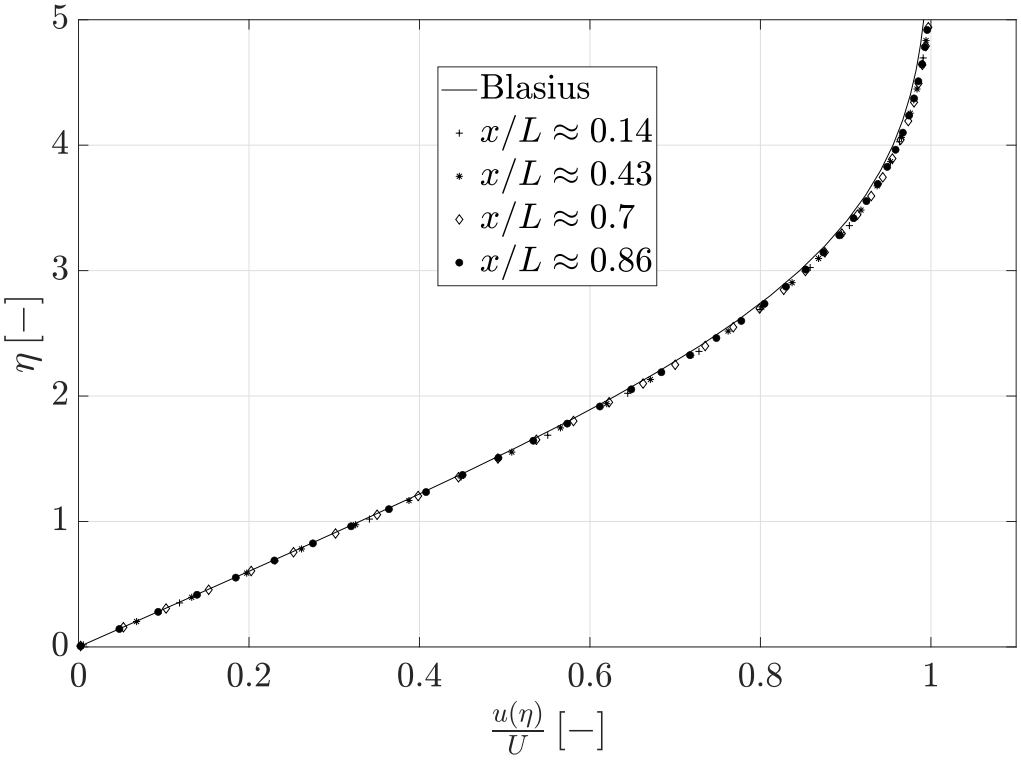
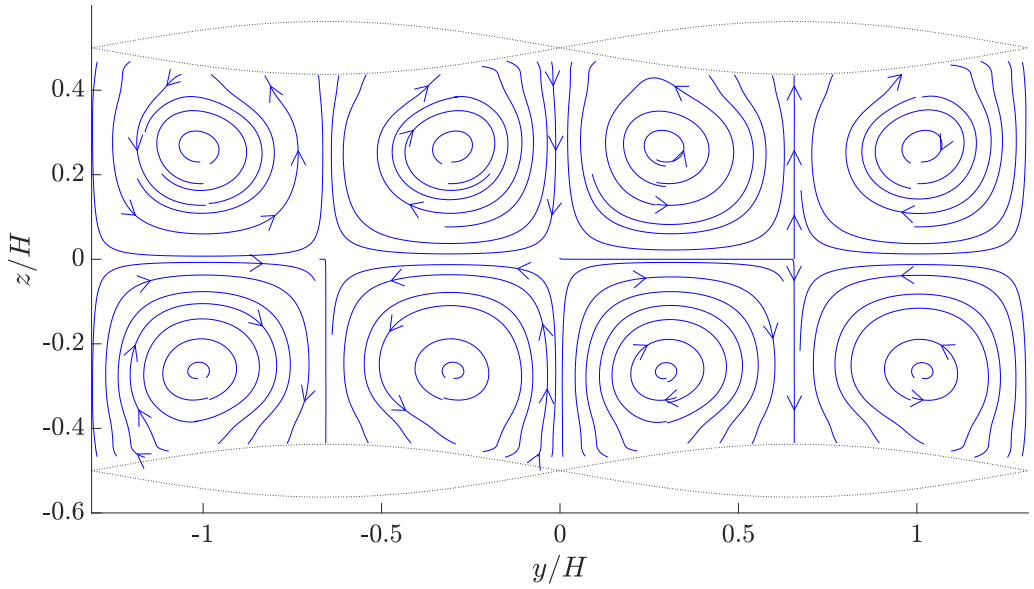


Figure 4.17: Blasius solution of a laminar boundary layer over a flat plate compared to CFD results

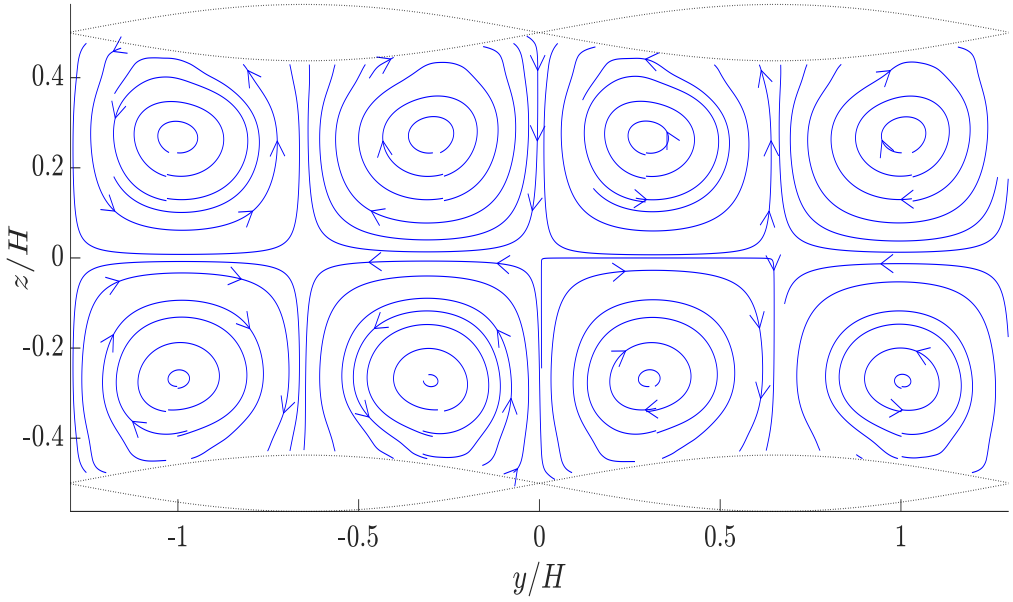
As expected, the boundary layer retains its shape across the whole plate, it is just scaled in the wall normal direction. This is the assumption made when introducing the similarity variable η. Some small deviations from the Blasius solution are observed for higher η, but this can be explained by the fact that the freestream velocity increases by approximately 0.5% above the flat plate, due to mass conservation. If this new apparent freestream velocity is accounted for in the definition of η and when scaling u, the graphs all collapse into one. To reduce this error further, the domain would have to be even taller for the boundary layers effect on U to diminish. These results indicate that the code is able to accurately model the formation of a laminar boundary layer developing over a flat plate, providing confidence that the code will be able to handle the boundary layer formation in other types of flows.

For the second part of the validation study, the steady state results from OpenFOAM are now compared to those from the LBM code. The exact same domain and flow parameters are used for both simulations, which are identical to those in the steady part of the mesh refinement study. The finest mesh considered in the mesh refinement study is used to produce the results in OpenFOAM. Figure 4.18 shows the streamlines for the two different simulations. It shows good qualitative comparability of the results, where the circulation structures are clearly visible and exhibits the same behaviour. It is noted that some of the streamlines in both plots are not closed, which is expected for a periodic domain for a steady state solution. This could be caused by too few data points in the streamwise direction to calculate a true average over the domain

length. It is possible that numerical error in the `streamslice` function in MATLAB could also contribute to this discrepancy.



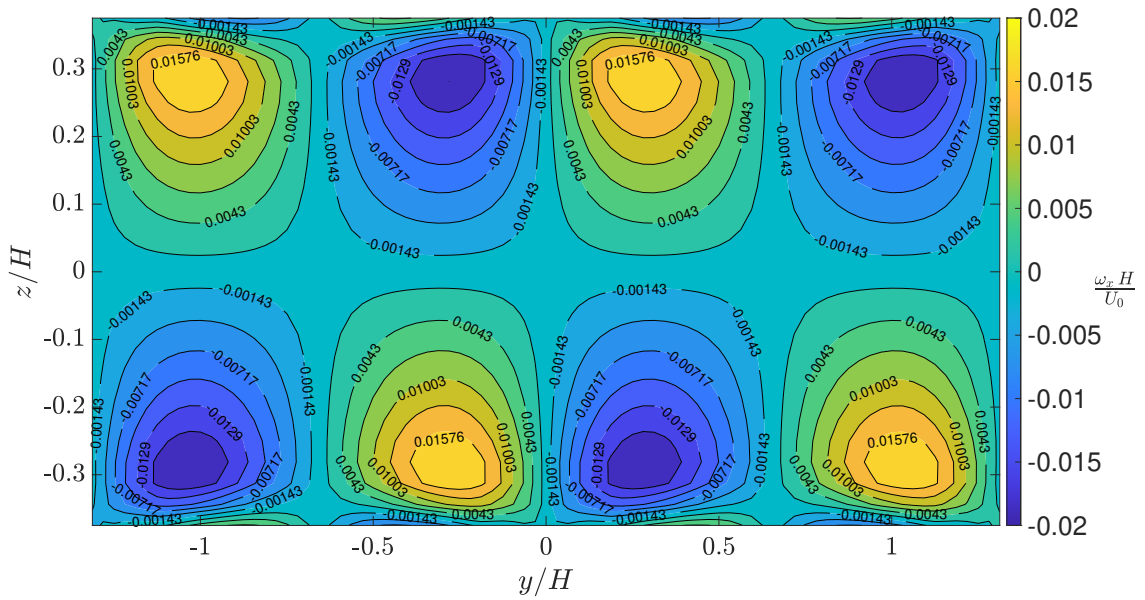
(a)



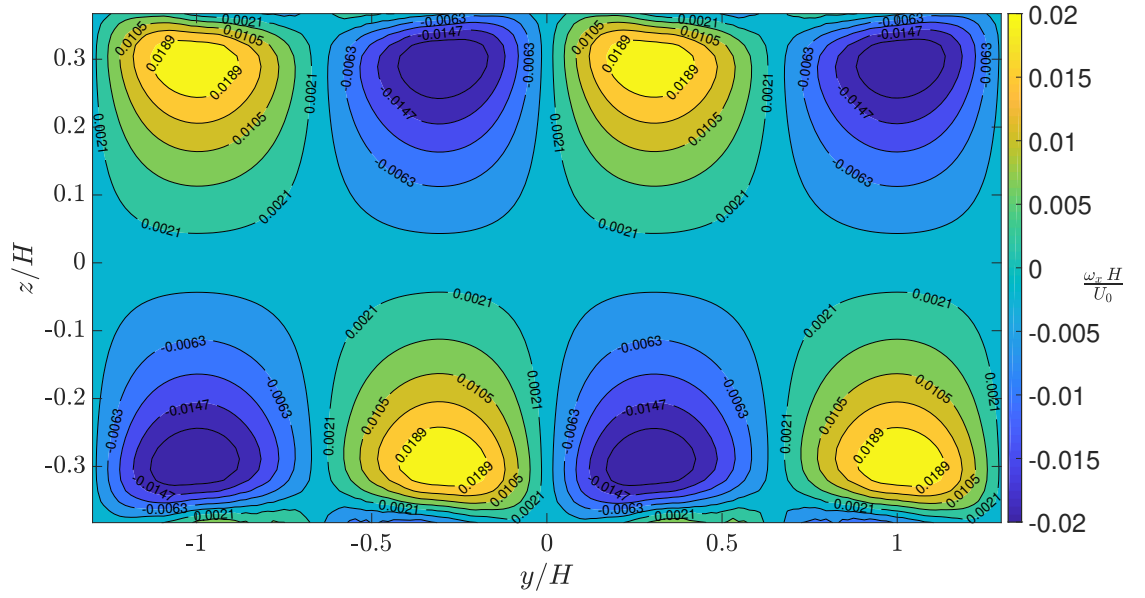
(b)

Figure 4.18: Base case: $Re_\tau = 25$, $\theta = \pi/8$, $a = 0.0625H$ and $|\mathbf{k}^*| = 2\,\pi$. (a) Streamline results from OpenFOAM, (b) Streamline results from LBM code

Figure 4.19 shows the streamwise-averaged x vorticity results for the two simulations. The circulation cells can be seen as increases in the streamwise vorticity, where the sign indicates the rotational direction. This visualisation enables for some quantitative data to be extracted from the figure. The peak nondimensionalised streamwise vorticity ω^* defined as $\omega H/U_0$ in the OpenFOAM simulation is $\omega^* \approx 0.0186$, and $\omega^* \approx 0.02$ for the LBM simulation. This represents a difference of approximately 7%, which is considerably lower than for the same comparison in the project work (Brostrøm 2019), where results from ANSYS Fluent were compared to the LBM results. An error of 7% is considered acceptable, and gives confidence that the CFD code in OpenFOAM is able to accurately simulate the formation of Langmuir Circulations in the steady state case.



(a)



(b)

Figure 4.19: Base case: $Re_\tau = 25$, $\theta = \pi/8$, $a = 0.0625H$ and $|\mathbf{k}^*| = 2\,\pi$. (a) Streamwise vorticity results from OpenFOAM, (b) Streamwise vorticity results from the LBM code

4.10 Parameter study

To answer the research questions stated in the introduction chapter, a parameter study is conducted. A base case is first defined and a simulation is done. This case is chosen in a way such that it is expected to produce unstable results for the Langmuir Circulations, based on observations from (Akselsen and Ellingsen 2019; Brostrøm 2019). The result form this simulation will

form the starting point for the parameter study. From here, two nondimensional parameters will be altered, and the goal will be to study the effects on the stability of the circulation currents by altering these parameters. The parameter study has been limited to look at the effect of varying the nominal friction Reynolds number Re_τ and the maximum streamwise wave steepness of the wall α_x. Meanwhile, the rest of the parameters of the domain will be kept constant, where the wave amplitude $a = 0.0625H$ and the transverse maximum wave steepness $\alpha_y = 0.15$. Since $\alpha_x = k_x a$ is altered, while $\alpha_y = k_y a$ and a remain constant, it means that θ, the angle between the wave numbers k_x and k_y, will also be varied as a consequence. It is noted that the domain lengths L_x and L_y in the x and y directions respectively, are rounded to integer numbers, which is done in order to ensure better numerical control over the locations of the periodic boundary conditions. This is useful when creating the geometry and meshing the fluid domain. The wave numbers are altered to preserve an integer number of wall wave lengths in both directions, however this change is only approximately 1%, and considered negligible compared to the increments of α_x in the parameter study. All the cases to be studied is listed in table 4.3, where the case names follows the structure

$$\text{Case } \underbrace{\text{XX}}_{Re_\tau}\text{\_}\overbrace{\text{XX}}^{\alpha_x}.$$

For all cases, a steady state simulation for a lower nominal friction Reynolds number of 25 on the same domain and mesh is used as an initial condition. This is done so that the Langmuir Circulations are present in the initial condition of the transient simulation. From there, the transient development is studied. The nominal friction Reynolds number is altered by adjusting the kinematic viscosity ν with (3.19), and subsequently the momentum source term F with (3.20), as discussed in section 4.5.2. In order to change the wave steepness α_x, the base case domain is simply stretched or compressed in the x direction, with a factor yielding the correct α_x. This task can readily be performed using SpaceClaim, where the geometry is modelled.

Table 4.3: Variation of nominal friction Reynolds number Re_τ and angle θ in the parameter study

Case name	Re_τ	α_x
Base case	45	0.36
Case 35_0.36	35	0.36
Case 40_0.36	40	0.36
Case 50_0.36	50	0.36
Case 45_0.26	45	0.26
Case 45_0.46	45	0.46

All cases in the parameter study are simulated for 25000 time steps. This represents a final

time of $t^*_{\text{max}} \approx 850$ for all simulations. It is noted that U_0 varies to some degree throughout the simulations, depending on the case, making it difficult to define an accurate time scale. A representative U_0 from after the initial readjustment to the new viscosity is chosen as the constant value, as it is the same to within 3% for all cases. Having the same time scale makes it easier to compare between the cases. The run-time of the simulations corresponds to above 150 flow through times for the longest domain in the parameter study, for Case $45\_0.26$, which is expected to be long enough for the instabilities of the flow structures to grow forth.

Chapter 5

Results

This chapter will present the CFD results of the transient development of the Langmuir Circulations. The base case simulation will first be studied in depth, where the objective is to observe in detail how the circulation patterns break down over time. Following this, the results from the other cases, defined in 4.10, will be presented. The effects on the flow stability from varying the parameters will be investigated.

5.1 Observed numerical characteristic

Before the results are presented, some characteristic proprieties of the mesh and the flow for the individual cases are given, in order to quantify how the different flows are resolved, and also to provide some addition metrics to describe the flows. These characteristic proprieties are presented in table 5.1. The table includes the spatial average and spatial maximum y-plus y^+_{avg} and y^+_{max}, maximum cell size in viscous units Δh^+, the true friction Reynolds number $Re_{\tau,t}$, compared to nominal friction Reynolds number Re_τ, and observed maximum Re_0 based on U_0 and H for each case. In addition, Spatial average and spatial maximum Courant numbers C_{avg} and C_{max} are also included, to say something about the temporal discretization of the cases. All y-plus values and Courant numbers are taken as the maximum of all time steps, to provide the worst case values. This is also done for Re_0, where the maximum streamwise velocity is used in (2.2) for the estimate. For Δh^+ and $Re_{\tau,t}$, the maximum of the quantity y^+_{avg} in time is used in (2.2), (4.2), (4.3) and (4.4) to try to estimate meaningful values for the individual cases.

Studying the table, it is determined that the mesh is well resolved for all cases, with y^+_{max} well within 1 for all cases, and y^+_{avg} of course even lower. The statement that $Re_{\tau,t}$ should be smaller than Re_τ holds for all but two cases, and the hope is that the following results will shine some light on why this is. The Reynolds number Re_0 has observed values well below the critical Reynolds number $Re_c = 11544$ for a plane Poiseuille flow. Looking at the Courant numbers, it is concluded that C_{avg} is well below one for all cases, of close to constant value of

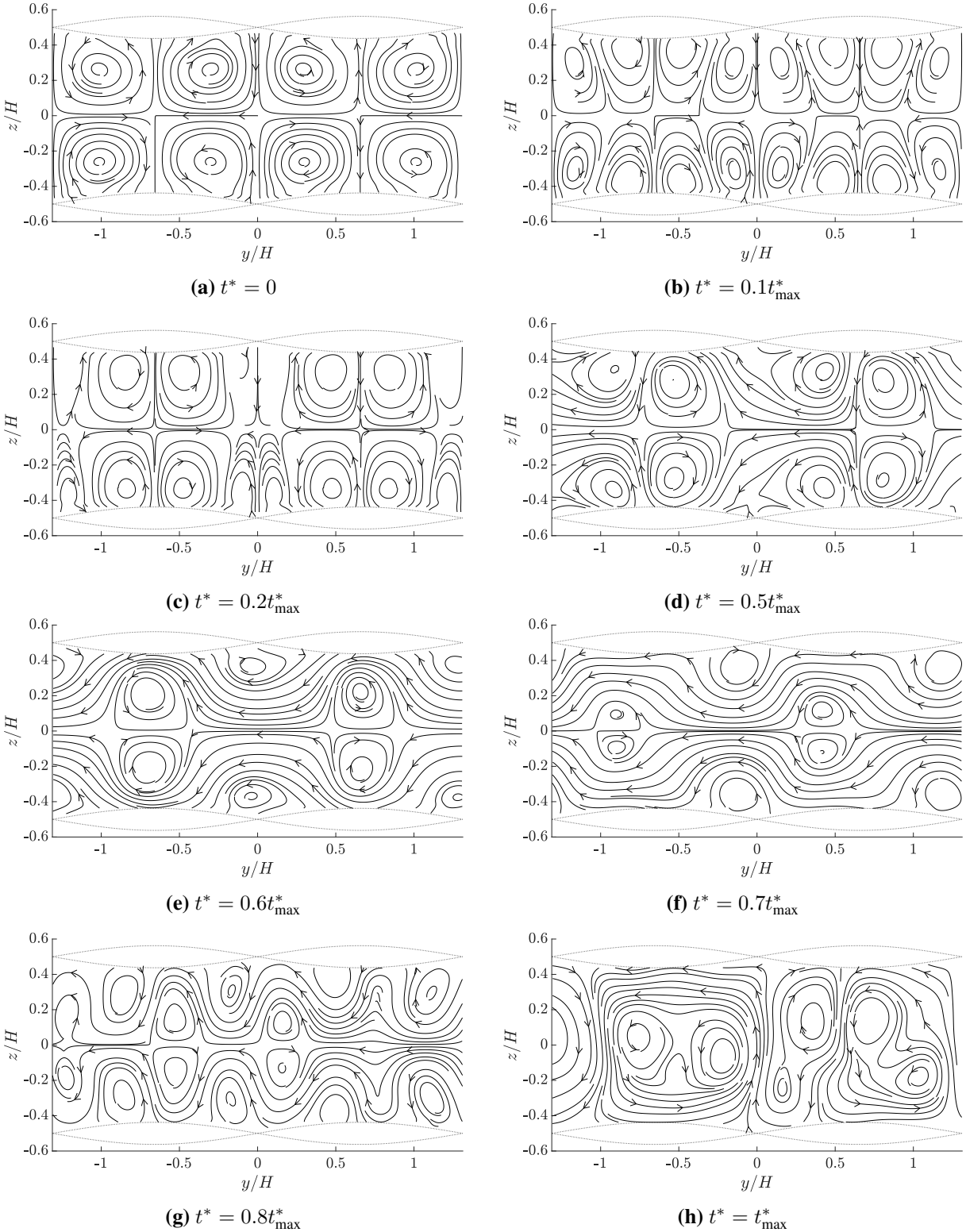
0.74. For the values of C_{max} however, some of the values are becoming somewhat too large, and should ideally not be much larger than one. As an implicit time scheme is used, this does not cause any stability issues, but accuracy could be reduced. It is noted that these values are the largest of the local values across all time steps for each simulation, meaning the absolute worst case.

Table 5.1: Observed values for highest spatial average and spatial maximum y-plus y^+_{avg} and y^+_{max}, and estimates for maximum cell size in viscous units Δh^+, true friction Reynolds number $Re_{\tau,t}$, compared to nominal friction Reynolds number Re_τ, and observed maximum Re_0 based on U_0 and H for each case. Spatial average and spatial maximum Courant numbers C_{avg} and C_{max} are also included.

Case name	y^+_{avg}	y^+_{max}	Δh^+	$Re_{\tau,t}$	Re_τ	Re_0	C_{avg}	C_{max}
Base case	0.18	0.49	2.45	39	45	1958	0.74	1.30
Case 35_0.36	0.13	0.23	1.74	28	35	1170	0.74	1.13
Case 40_0.36	0.15	0.26	1.99	32	40	1501	0.74	1.13
Case 50_0.36	0.31	0.61	4.17	67	50	2416	0.74	1.38
Case 45_0.26	0.18	0.30	2.44	39	45	1946	0.75	1.14
Case 45_0.46	0.23	0.52	3.02	48	45	1921	0.74	1.22

5.2 Transient development of the base case simulation

Figure 5.1 shows the streamline plots for the base case simulation. The initial condition is given for $t^* = 0$, which depicts the regular and steady circulation patterns from the steady state simulation of $Re_\tau = 25$. Moving forward in time, to $0.1t^*_{\text{max}}$, a new pattern has begun to emerge. The original vortex formations have become smaller and migrated towards the saddle points of the walls. Meanwhile, new pairs of counter rotating cells have been created at the peaks at troughs of the domain. This tendency continues, and at $0.2t^*_{\text{max}}$, the original cells have almost completely vanished while the new set grows larger. At $0.5t^*_{\text{max}}$, the initial cells are completely gone, and the newly formed cells have taken over the domain. in addition, the streamlines have begun to move in between cell pairs. When time reaches $0.6t^*_{\text{max}}$, a new seemingly ordered structure has appeared. For the lower half of the domain, the cells who has a counter clockwise have moved to the saddle points, while the clockwise rotating cells have moved above the peaks and troughs. The opposite is true for the upper half, retaining a symmetrical pattern about the midplane. The streamlines indicate a net flux in the negative transverse y direction, however this is below 0.5% of the streamwise bulk flow. At $0.7t^*_{\text{max}}$, a similar pattern is visible, only with smaller vortices above the peaks and troughs. At $0.8t^*_{\text{max}}$, new vortices with higher wavenumbers have spawned, and at $t^* = t^*_{\text{max}}$, the regular pattern is gone and the vortex structures have become irregular.



(a) $t^* = 0$ **(b)** $t^* = 0.1t^*_{\text{max}}$

(c) $t^* = 0.2t^*_{\text{max}}$ **(d)** $t^* = 0.5t^*_{\text{max}}$

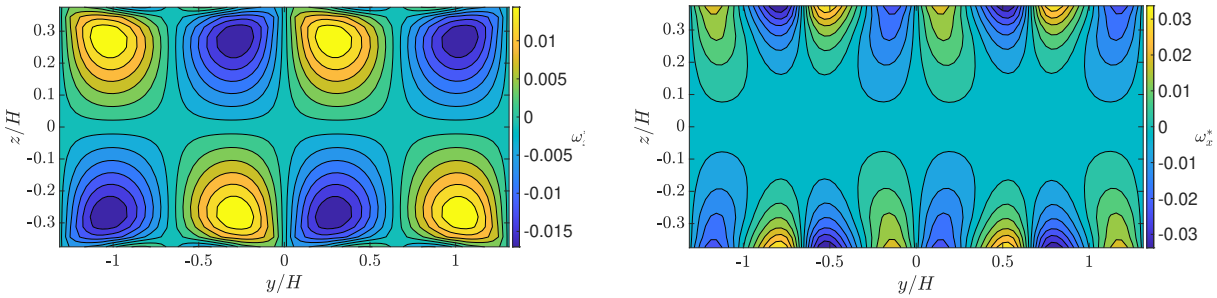
(e) $t^* = 0.6t^*_{\text{max}}$ **(f)** $t^* = 0.7t^*_{\text{max}}$

(g) $t^* = 0.8t^*_{\text{max}}$ **(h)** $t^* = t^*_{\text{max}}$

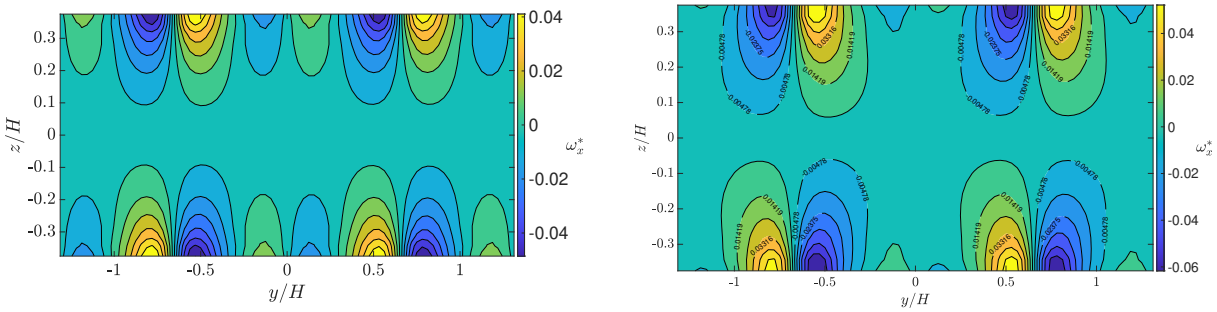
Figure 5.1: Base case: Streamwise-averaged normal velocity streamlines for different times, where $Re_\tau = 45$, $\alpha_x = 0.36$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/8$. Time increases from left to right and top to bottom.

Figure 5.2 shows the streamwise-averaged x vorticity contours for the same time instances as for the streamlines. While the streamlines give a better image of the qualitative behaviour of the circulation cells, the vorticity contours are able tell something about the strength of the currents. It is noted that the range of the colour bar to the right of the plots is allowed to vary in time, to better show variations in vorticity within each time instance. This is a compromise as the colour intensity looses its meaning when comparing across time steps. The increasing range of the colour bar as time moves forward, indicates stronger circulatory motion in the flow. At $0.8t^*_{\text{max}}$, the regular and antisymmetric pattern has disappeared. It is noted that the vorticity peaks are an order of magnitude larger than the for the regular structures in the initial state.

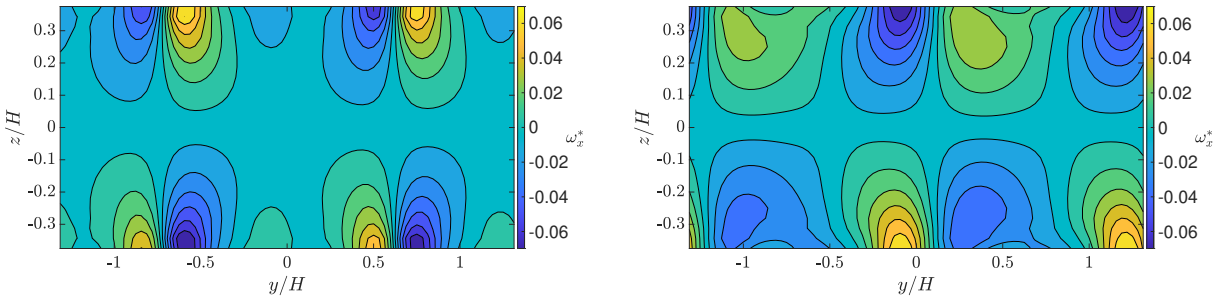
The final plot in both figure 5.1 and figure 5.2 exhibit irregular and seemingly random behaviour. It is suspected that the flow has in fact become turbulent, or are are in the transition of becoming turbulent due to instabilities caused by the undulating walls. To investigate this further, an x normal plane at $x = 0$ is extracted from ParaView, coloured with contours of the velocity magnitude $|\mathbf{u}^*|$. Four different time instances are studied, and the result is presented in figure 5.3. The plots have no axis labels, however the axes are identical to that of 5.1. The plane is rectangular as a cut through the walls at $x = 0$ produces straight lines, as $\sin(x = 0) = 0$. At $t^* = 0$, the initial condition is shown. It depicts a close to y independent velocity magnitude field, with only small differences between the y locations lining up with saddle points or with peaks and troughs of the wall boundary. For $t^* = 0.3t^*_{\text{max}}$, the velocity magnitude is shown to increase in the path of the saddle points, and decrease in the line of peaks and troughs. This is consistent with the observations of high momentum paths and low momentum paths in line of saddle points, and peaks and troughs in wall geometry by (Chan et al. 2018). This study was for pipes with equivalent wavy wall geometry, and at much higher Reynolds numbers. At $t^* = 0.8t^*_{\text{max}}$, signs of unstable structures in the flow is becoming visible, and at $t^* = 0.9t^*_{\text{max}}$, the velocity magnitude show erratic behaviour. It is noted that the values of maximum value of $|\mathbf{u}^*|$ has decreased significantly, and wider velocity profile is suggested by the short transition in colour from dark blue near the wall to grey. A flatter profile is indicated by small colour gradients away from the walls. All this is highly indicative of turbulent flow, which was suspected from the streamline and vorticity results.



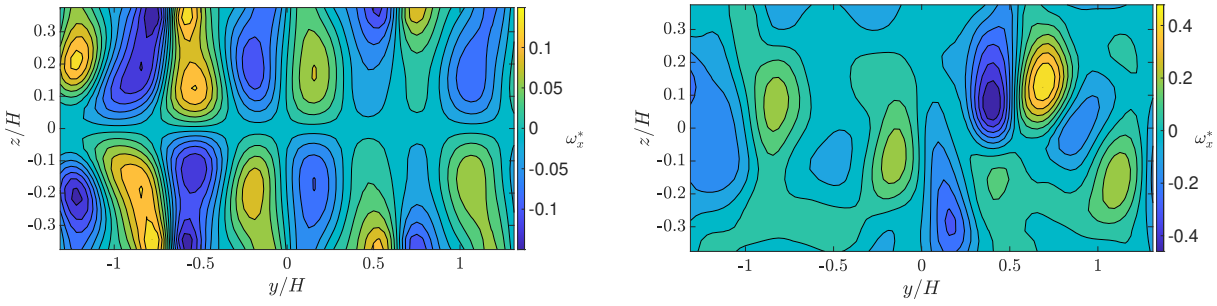
(a) $t^* = 0$ **(b)** $t^* = 0.1t^*_{\max}$



(c) $t^* = 0.2t^*_{\max}$ **(d)** $t^* = 0.4t^*_{\max}$

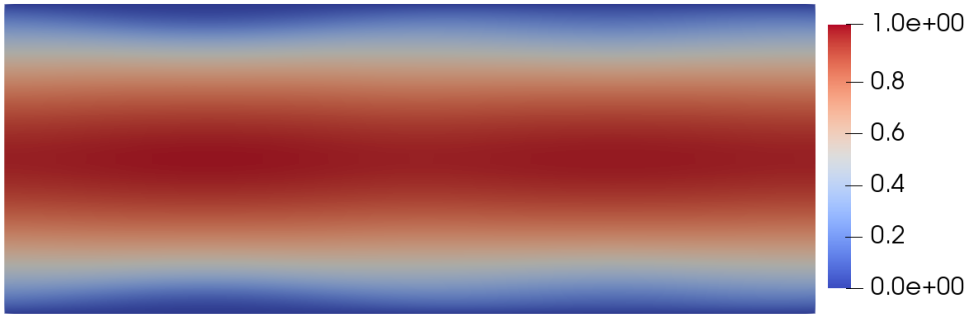


(e) $t^* = 0.5t^*_{\max}$ **(f)** $t^* = 0.7t^*_{\max}$

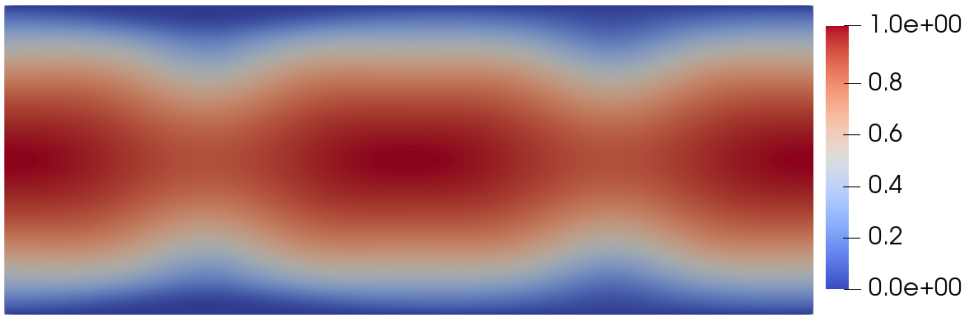


(g) $t^* = 0.8t^*_{\max}$ **(h)** $t^* = t^*_{\max}$

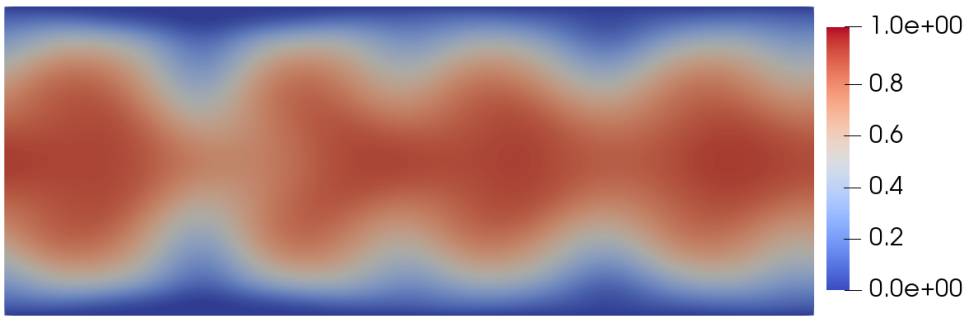
Figure 5.2: Base case: Streamwise-averaged x vorticity contours for different times, where $Re_\tau = 45$, $\alpha_x = 0.36$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/8$. Time increases from left to right and top to bottom.



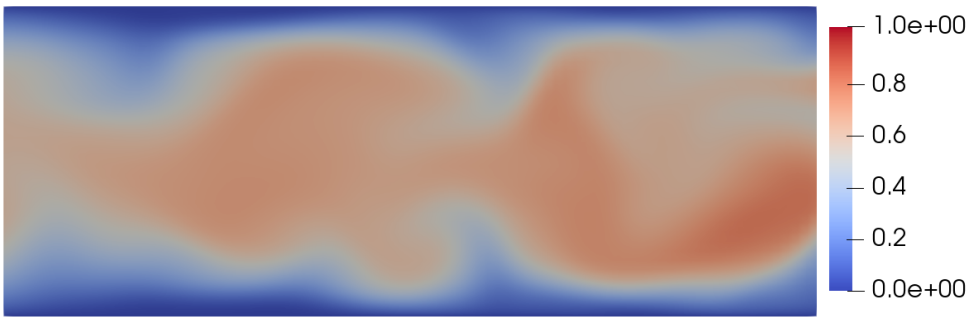
(a) $t^* = 0$



(b) $t^* = 0.3t^*_{\text{max}}$



(c) $t^* = 0.8t^*_{\text{max}}$



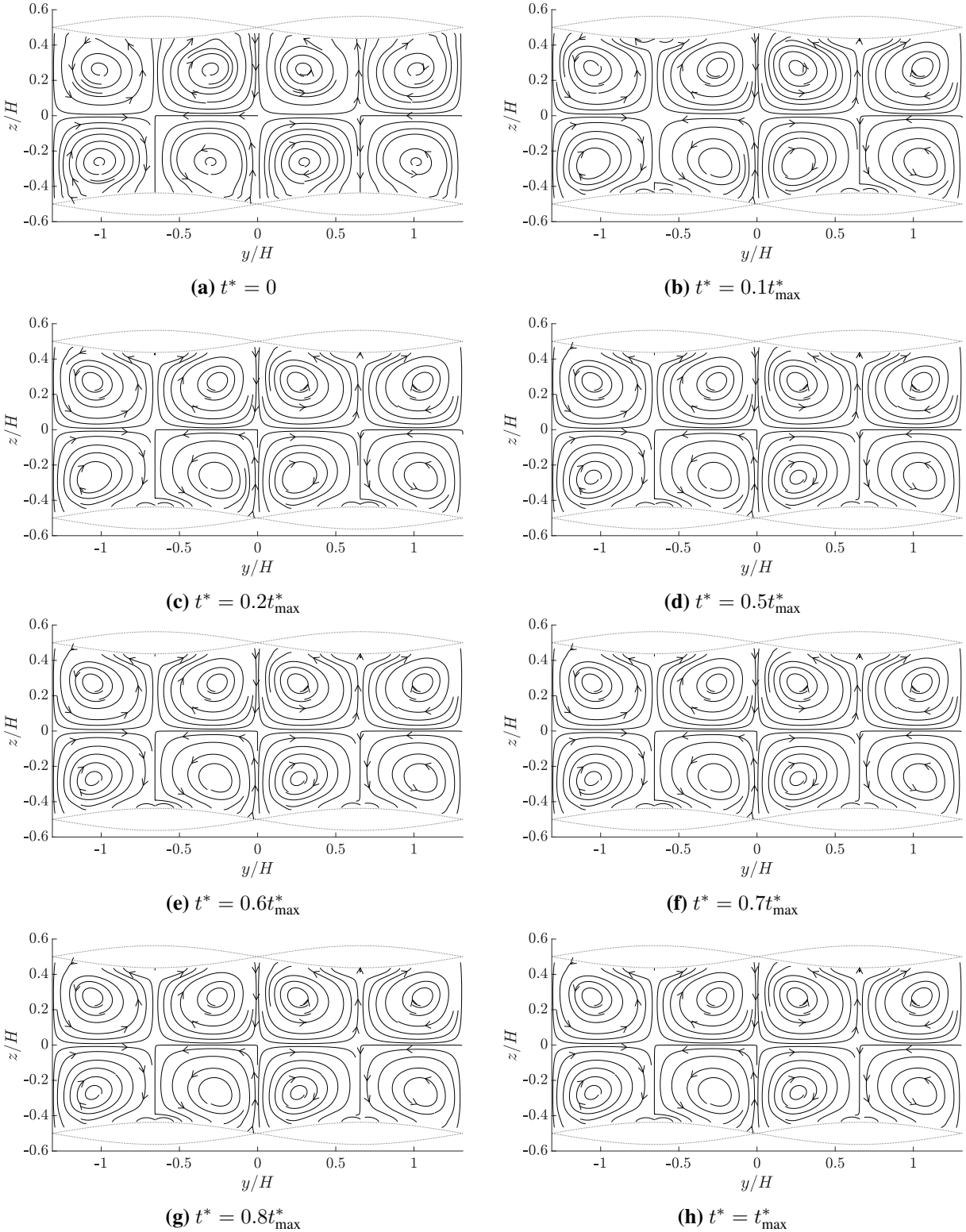
(d) $t^* = 0.9t^*_{\text{max}}$

Figure 5.3: Base case: Contour plots of velocity magnitude $|\mathbf{u}^*|$ in x-normal plane at $x = 0$ at different instances in time.

5.3 Effects of varying the friction Reynolds number

The variation of the nominal friction Reynolds number is first conducted, starting with the lowest Reynolds number, namely Case $35\_0.36$. In the project work leading up to the present work, the highest Re_τ to produce stable and regular circulation cells, was 30, which is why a slightly higher value was chosen as the minimal Reynolds number in this study. It is noted that the initial conditions for $t^* = 0$ is identical for all cases where Re_τ is varied, since they are all continuations of the same steady state simulation. They are included in all figures as a reference for the time developed plots. Figure 5.4 show the time development of the streamlines throughout the simulation. At $t^* = 0.1t^*_{\text{max}}$, some very slight tendencies of new streamlines have emerged in-line with the peaks and troughs of the wall, where the counter rotating vortex pairs appeared for the base case. However, from this point in time, the circulation cells retain their shape for the rest of the simulation. Looking at 5.5, it can be seen that the vorticity magnitudes have increased slightly, and remain the same from $0.1t^*_{\text{max}}$ until the end of the simulation.

Figure 5.6 shows the streamlines for Case $40\_0.36$. Similarly to Case $35\_0.36$, new pairs of circulation cells have emerged at the peaks and troughs of the wall at $t^* = 0.1t^*_{\text{max}}$. In this case they are slightly more prominent, and the original circulation currents show signs of yielding for the newly formed ones, migrating towards the saddle points, as was also what happened in the base case results. From this point in time, no visible changes in the shape of the circulation cells are visible. The vorticity in figure 5.7 indicates the same behaviour. Some slight increase in vorticity magnitude can be seen from the rescaling of the colour bar throughout the simulation, but as for Case $35\_0.36$, this case has also appeared to reach a steady state solution.



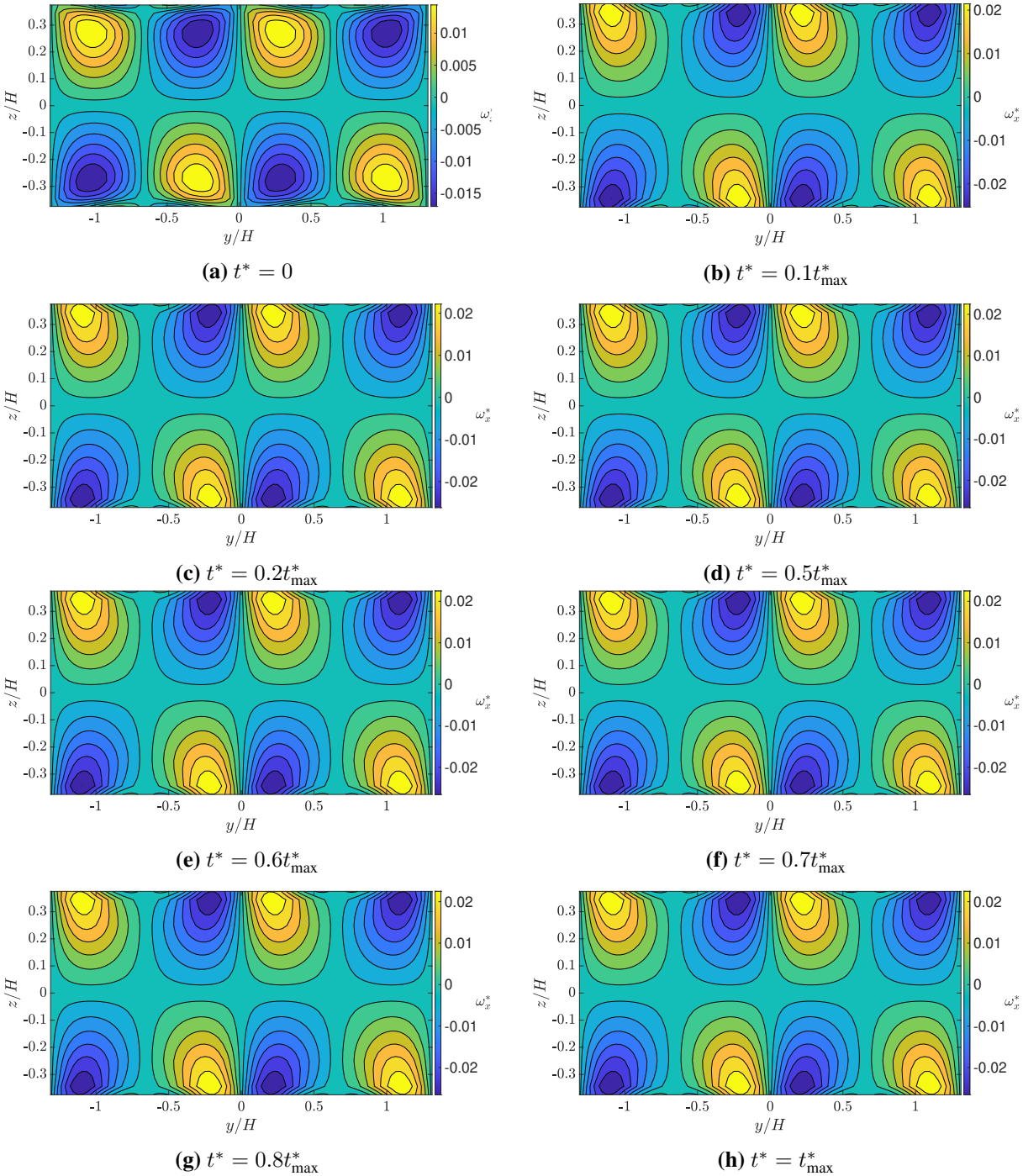
(a) $t^* = 0$ **(b)** $t^* = 0.1t^*_{\text{max}}$

(c) $t^* = 0.2t^*_{\text{max}}$ **(d)** $t^* = 0.5t^*_{\text{max}}$

(e) $t^* = 0.6t^*_{\text{max}}$ **(f)** $t^* = 0.7t^*_{\text{max}}$

(g) $t^* = 0.8t^*_{\text{max}}$ **(h)** $t^* = t^*_{\text{max}}$

Figure 5.4: Case 35_0.36: Streamwise-averaged normal velocity streamlines for different times, where $Re_\tau = 35$, $\alpha_x = 0.36$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/8$. Time increases from left to right and top to bottom.



(a) $t^* = 0$

(b) $t^* = 0.1t^*_{\text{max}}$

(c) $t^* = 0.2t^*_{\text{max}}$

(d) $t^* = 0.5t^*_{\text{max}}$

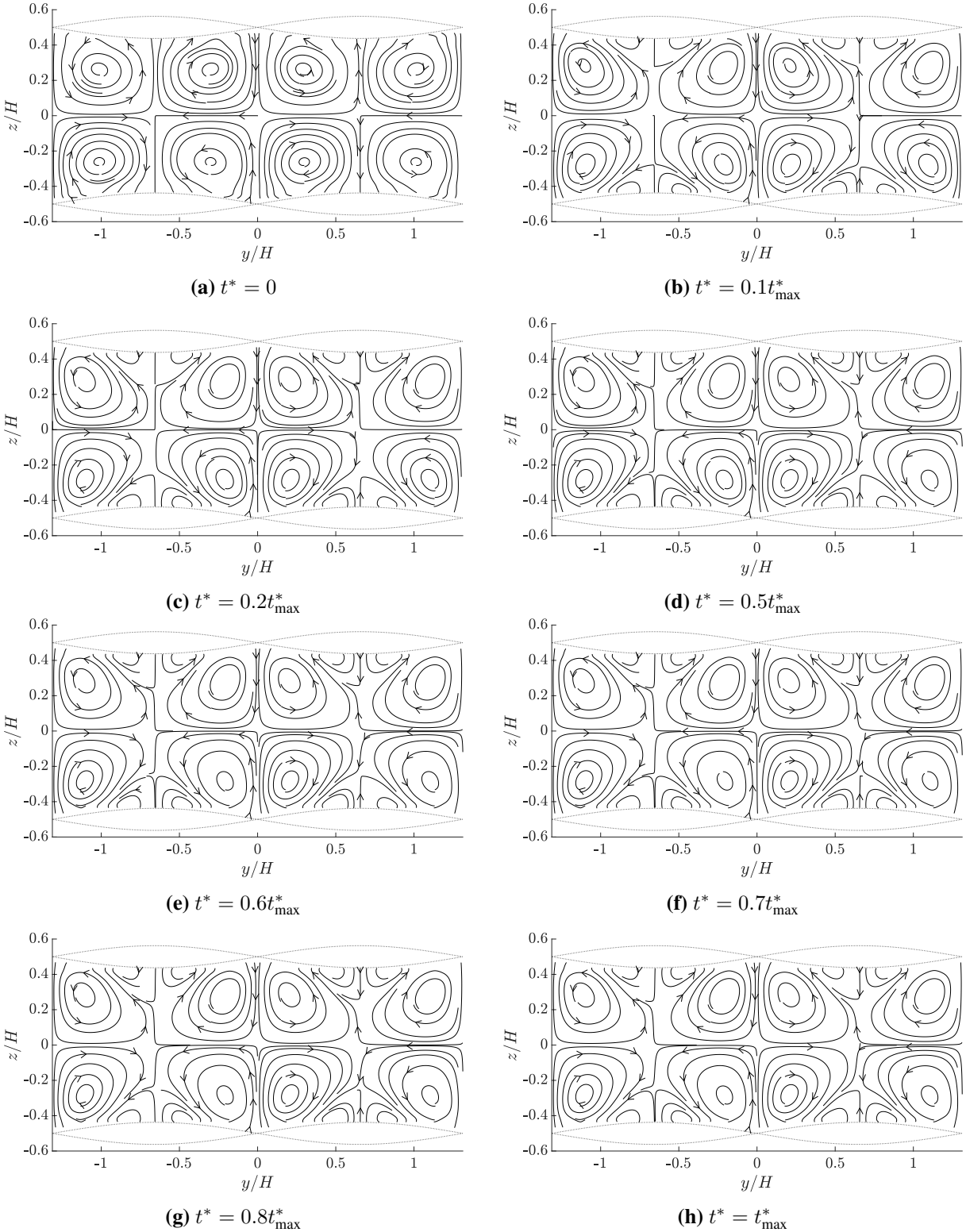
(e) $t^* = 0.6t^*_{\text{max}}$

(f) $t^* = 0.7t^*_{\text{max}}$

(g) $t^* = 0.8t^*_{\text{max}}$

(h) $t^* = t^*_{\text{max}}$

Figure 5.5: Case 35_0.36: Streamwise-averaged x vorticity contours for different times, where $Re_\tau = 35$, $\alpha_x = 0.36$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/8$. Time increases from left to right and top to bottom.



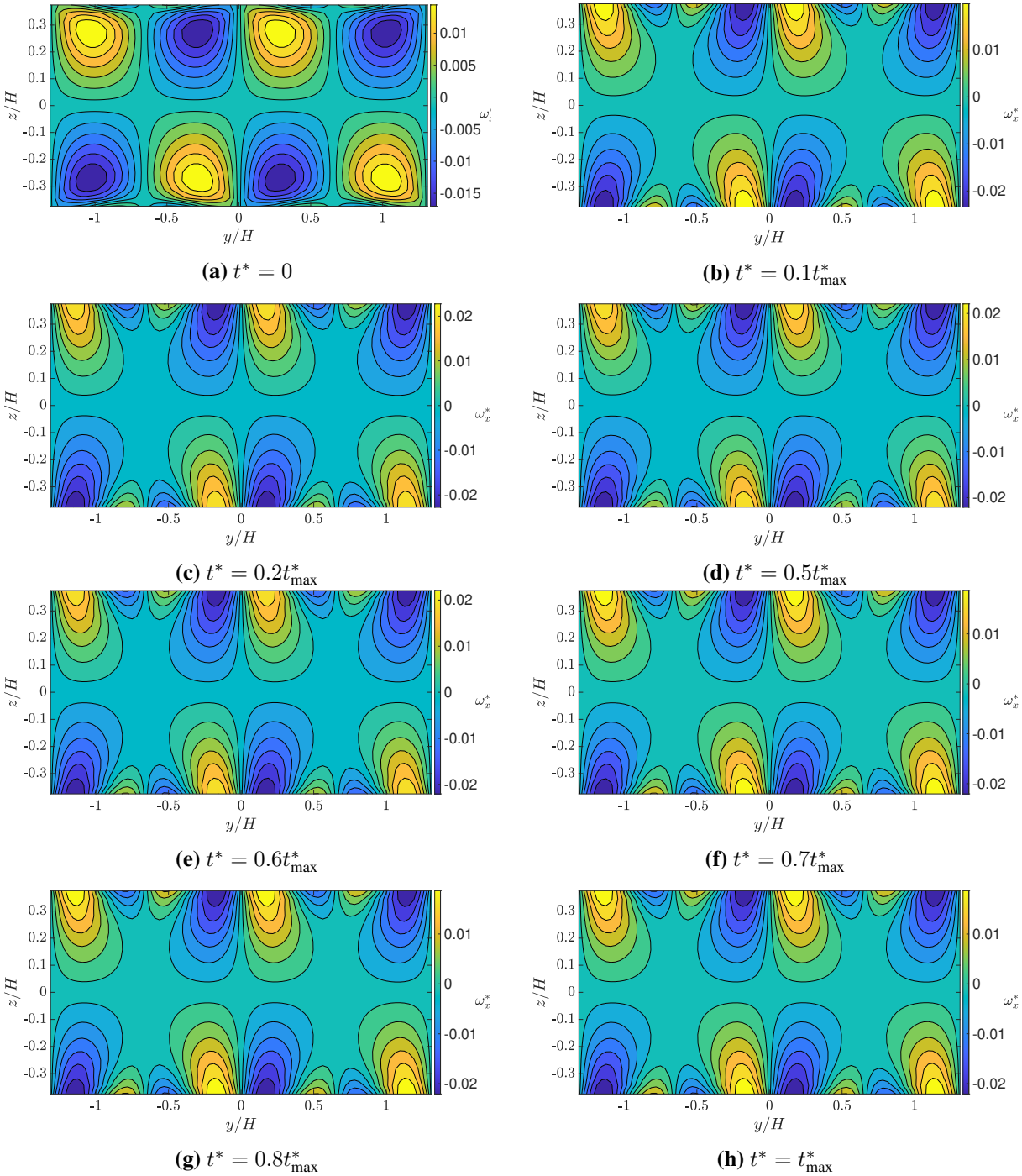
(a) $t^* = 0$ **(b)** $t^* = 0.1t^*_{\text{max}}$

(c) $t^* = 0.2t^*_{\text{max}}$ **(d)** $t^* = 0.5t^*_{\text{max}}$

(e) $t^* = 0.6t^*_{\text{max}}$ **(f)** $t^* = 0.7t^*_{\text{max}}$

(g) $t^* = 0.8t^*_{\text{max}}$ **(h)** $t^* = t^*_{\text{max}}$

Figure 5.6: Case 40_0.36: Streamwise-averaged normal velocity streamlines for different times, where $Re_\tau = 35$, $\alpha_x = 0.36$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/8$. Time increases from left to right and top to bottom.



(a) $t^* = 0$ **(b)** $t^* = 0.1t^*_{\text{max}}$

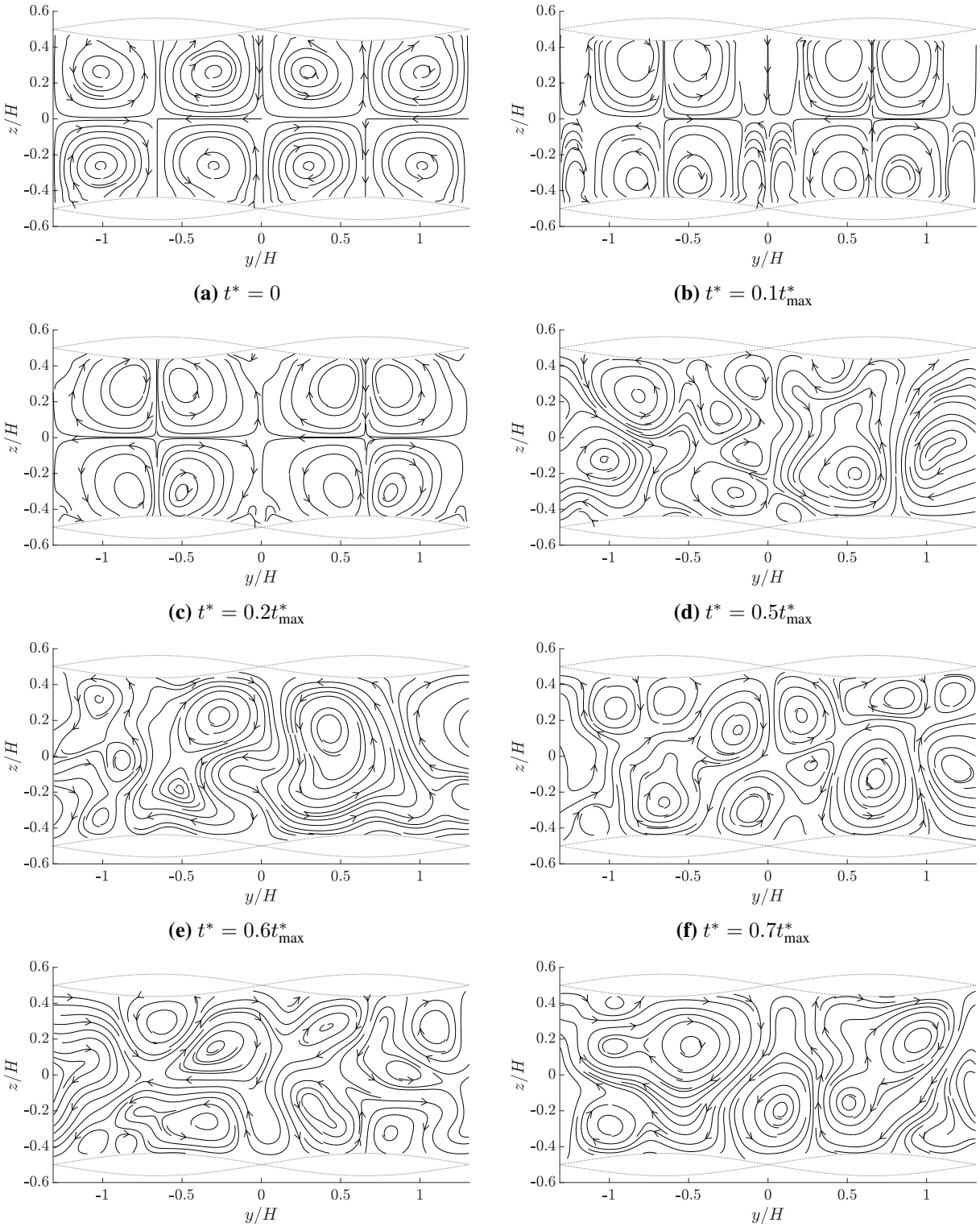
(c) $t^* = 0.2t^*_{\text{max}}$ **(d)** $t^* = 0.5t^*_{\text{max}}$

(e) $t^* = 0.6t^*_{\text{max}}$ **(f)** $t^* = 0.7t^*_{\text{max}}$

(g) $t^* = 0.8t^*_{\text{max}}$ **(h)** $t^* = t^*_{\text{max}}$

Figure 5.7: Case 40_0.36: Streamwise-averaged x vorticity contours for different times, where $Re_\tau = 40$, $\alpha_x = 0.36$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/8$. Time increases from left to right and top to bottom.

The highest nominal friction Reynolds number investigated in this study, is for $Re_\tau = 50$, and is the only case with a Re_τ larger than for the base case. Figure 5.8 show the streamlines from the simulation. For $t^* = 0.1t^*_{\text{max}}$, the new pairs of cells have already become dominant, and the original cells have moved to the saddle points and are close to disappearing. This plot is very similar to that of $t^* = 0.2t^*_{\text{max}}$ from the base case results in figure 5.1, indicating that the break down process has been accelerated. At $t^* = 0.2t^*_{\text{max}}$, the original cells are no longer visible, and at $0.5t^*_{\text{max}}$, the structures have already become fully irregular. As for the base case results, this is indicating that the flow is transitioning to turbulent. Figure 5.9 shows the vorticity contours for the case. The results show matching behaviour as the streamlines in terms of where the regular circulation structures have broken down. As for the base case results in figure 5.2, the plots after the break down show a large increase in vorticity amplitudes.



(a) $t^* = 0$

(b) $t^* = 0.1t^*_{\text{max}}$

(c) $t^* = 0.2t^*_{\text{max}}$

(d) $t^* = 0.5t^*_{\text{max}}$

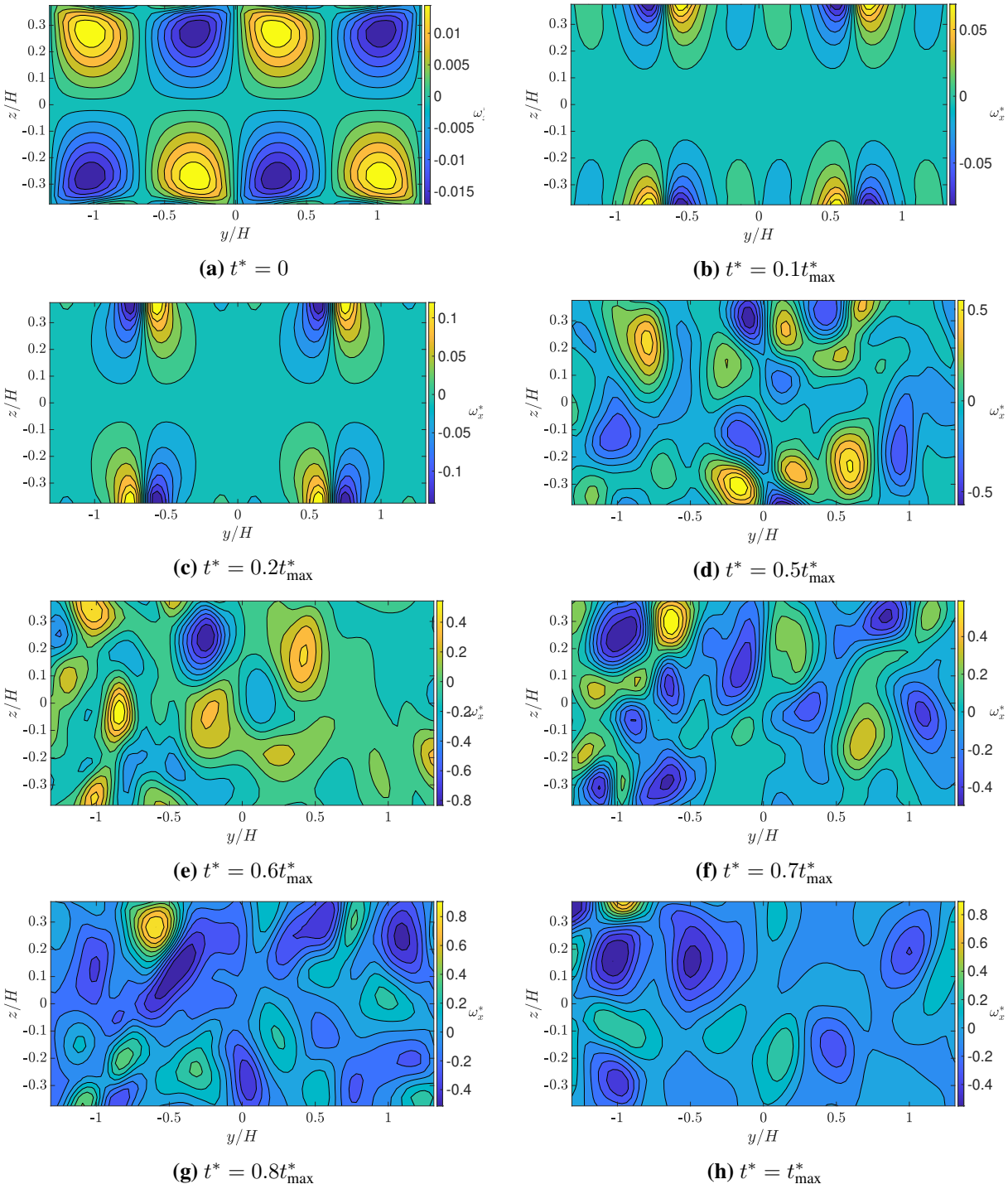
(e) $t^* = 0.6t^*_{\text{max}}$

(f) $t^* = 0.7t^*_{\text{max}}$

(g) $t^* = 0.8t^*_{\text{max}}$

(h) $t^* = t^*_{\text{max}}$

Figure 5.8: Case 50_0.36: Streamwise-averaged normal velocity streamlines for different times, where $Re_\tau = 35$, $\alpha_x = 0.36$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/8$. Time increases from left to right and top to bottom.



(a) $t^* = 0$

(b) $t^* = 0.1t^*_{\text{max}}$

(c) $t^* = 0.2t^*_{\text{max}}$

(d) $t^* = 0.5t^*_{\text{max}}$

(e) $t^* = 0.6t^*_{\text{max}}$

(f) $t^* = 0.7t^*_{\text{max}}$

(g) $t^* = 0.8t^*_{\text{max}}$

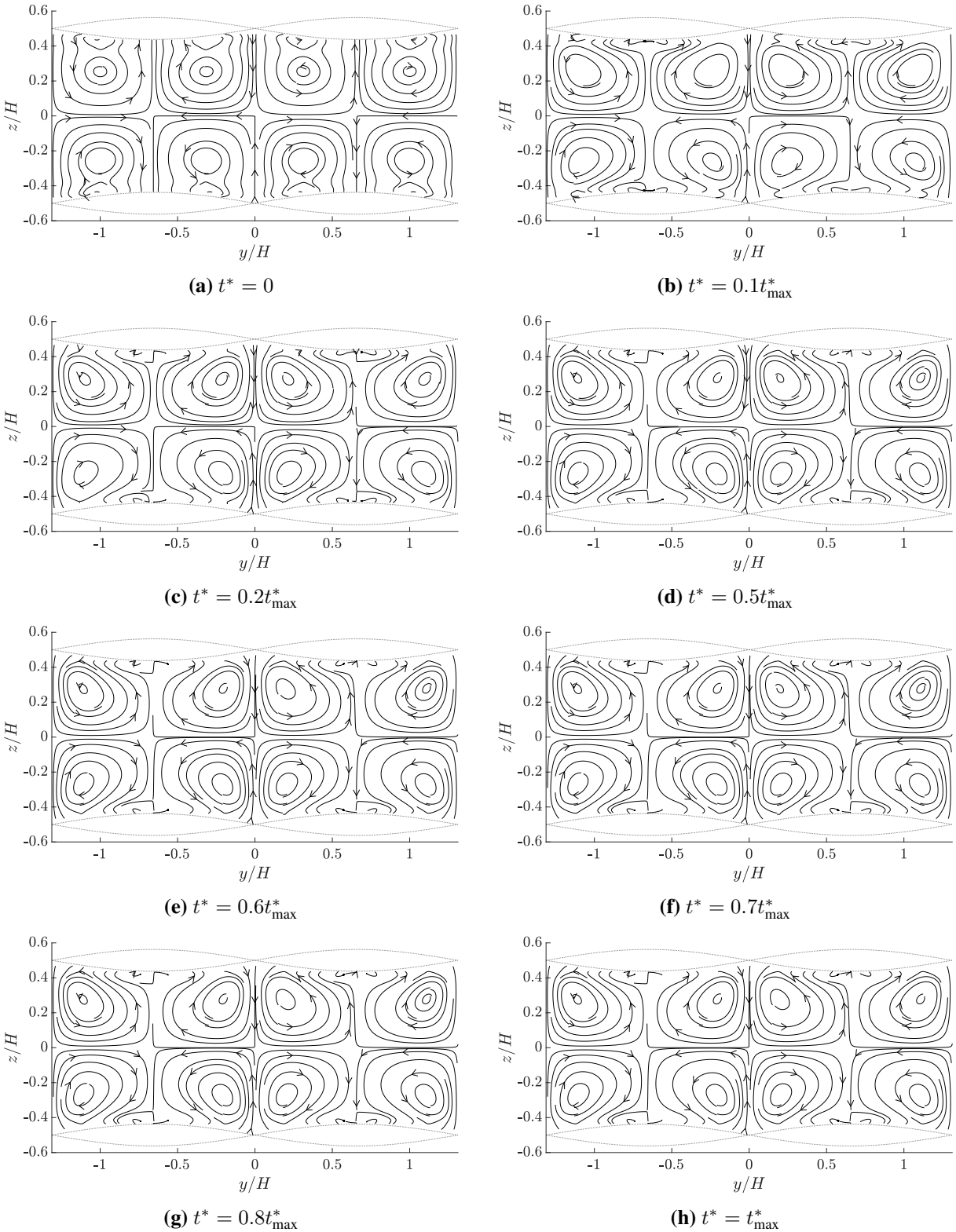
(h) $t^* = t^*_{\text{max}}$

Figure 5.9: Case 50_0.36: Streamwise-averaged x vorticity contours for different times, where $Re_\tau = 50$, $\alpha_x = 0.36$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/8$. Time increases from left to right and top to bottom.

5.4 Effects of varying the wall wave steepness

In this section, the cases with variations in the wave steepness α_x are considered, and the first to be investigated is Case $45\_0.26$, which corresponds to lower wave steepness of the wall than the base case. Unlike the previous cases, the geometry has had to be modified. This also means that the steady state solution will be different than the previous cases, even though they have the same nominal friction Reynolds number of 25. Because of this, it is first verified that stable Langmuir Circulations are present in the initial condition of the transient simulation. Looking at the streamlines in figure 5.10, it is concluded that regular circulation cells are present in the initial condition at $t^* = 0$. It is noted that smaller flow structures are present between the large scale circulation cells and the walls. At $t^* = 0.1t^*_{\text{max}}$, the large cells have moved towards the saddle points in the walls, however they have not yielded for any clear emerging cells, and are still the dominant structures in the streamline plot. Moving forward in time, essentially no more changes are visible in the streamline plots. Some small structures are visible on the peaks and troughs of the wall from the second plot, but they have not grown as large as they did for Case $40\_0.36$, in figure 5.6. Figure 5.11 of the vorticity indicates that the small structures in the initial condition has the opposite rotation of the larger scale cells. Some small variations in the colour bar range as time increases, indicates that a actual steady state is not reached, however, the flow seems to have reached a stable solution, with no indications of evolving any further qualitatively.

The final case in the parameter study is Case $45\_0.46$, where $\alpha_x = 0.46$. This corresponds to a shorter domain and tighter waves in the streamwise direction in the wall. The initial condition of $t^* = 0$ in the streamline plots in figure 5.12 show that the circulation cells are present and regular in the beginning for this case as well. At $t^* = 0.1t^*_{\text{max}}$, new pairs of cells with the opposite rotation as the original cells have already started to dominate the domain. The original Langmuir cells have decreased in size and moved towards the saddle points. At $t^* = 0.2t^*_{\text{max}}$, the new pairs of oppositely rotating circulation cells continues to grow, while the original cells have almost completely vanished. Both the plots from $0.1t^*_{\text{max}}$ and $0.2t^*_{\text{max}}$ show closely resembling qualitative features as the corresponding time instances for Case $50\_0.36$, in figure 5.8. For time $t^* = 0.5t^*_{\text{max}}$ however, the pattern takes another shape. The smaller cells towards the saddle points seem to reappear, forcing the new cells closer together on the peaks and troughs. The pattern remains almost completely regular and symmetric. For $t^* = 0.6t^*_{\text{max}}$, highly uniform structures are visible, resembling the initial state, only the rotational direction is opposite. Moving forward in time, to $t^* = 0.7t^*_{\text{max}}$, the pattern breaks down, and irregular behaviour of the streamline becomes visible. Figure 5.13 shows the corresponding vorticity results for the case. The same behaviour as in the streamline plots are visible in these plots as well. The state at $t^* = 0.5t^*_{\text{max}}$ is also shown to be highly regular in the vorticity plot. For $t^* = 0.6t^*_{\text{max}}$ however, shows that the vortices does not look as regular and smooth as they appeared in figure 5.12.



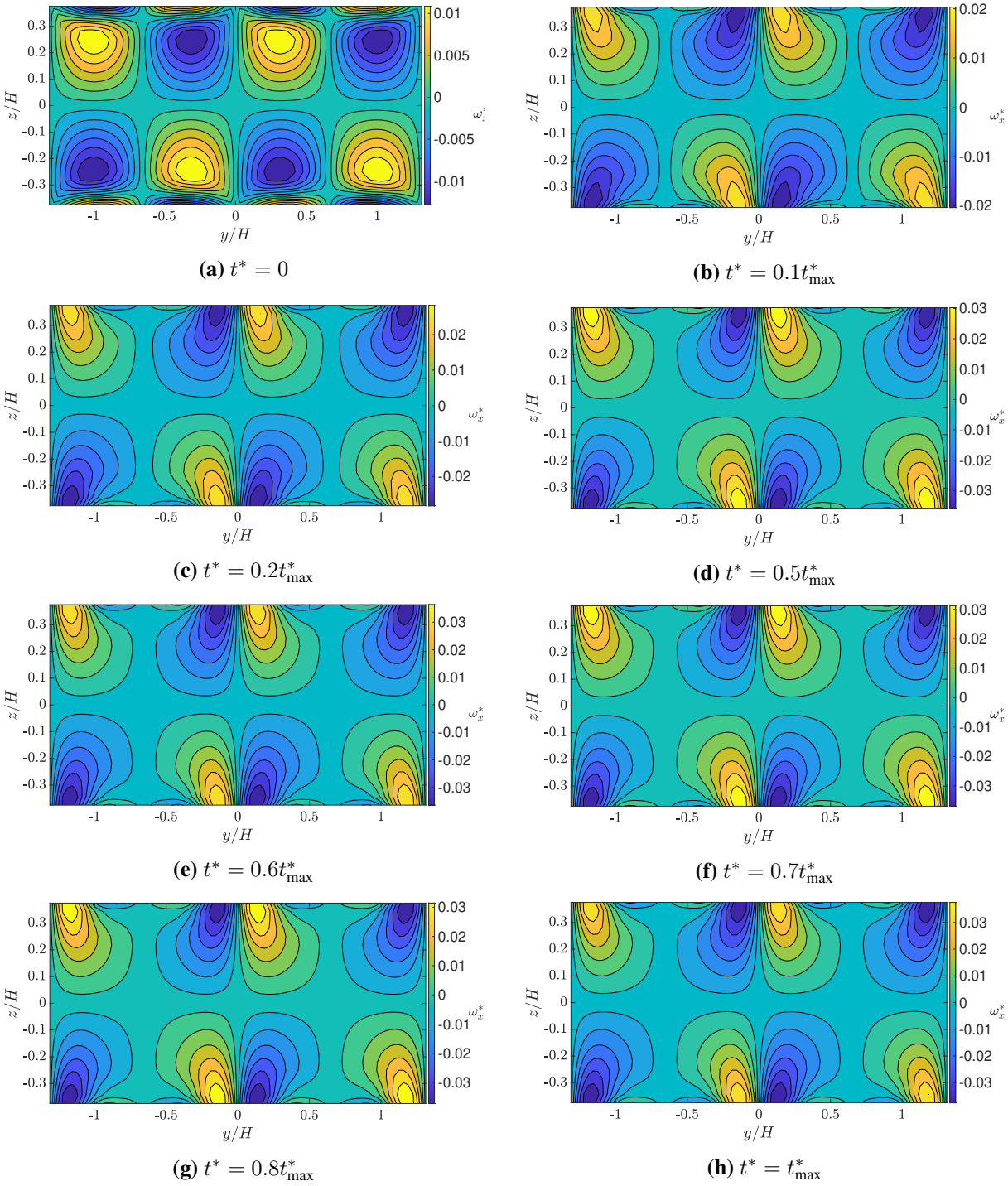
(a) $t^* = 0$ **(b)** $t^* = 0.1t^*_{\text{max}}$

(c) $t^* = 0.2t^*_{\text{max}}$ **(d)** $t^* = 0.5t^*_{\text{max}}$

(e) $t^* = 0.6t^*_{\text{max}}$ **(f)** $t^* = 0.7t^*_{\text{max}}$

(g) $t^* = 0.8t^*_{\text{max}}$ **(h)** $t^* = t^*_{\text{max}}$

Figure 5.10: Case 45_0.26: Streamwise-averaged normal velocity streamlines for different times, where $Re_\tau = 45$, $\alpha_x = 0.26$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/6$. Time increases from left to right and top to bottom.



(a) $t^* = 0$ **(b)** $t^* = 0.1t^*_{\text{max}}$

(c) $t^* = 0.2t^*_{\text{max}}$ **(d)** $t^* = 0.5t^*_{\text{max}}$

(e) $t^* = 0.6t^*_{\text{max}}$ **(f)** $t^* = 0.7t^*_{\text{max}}$

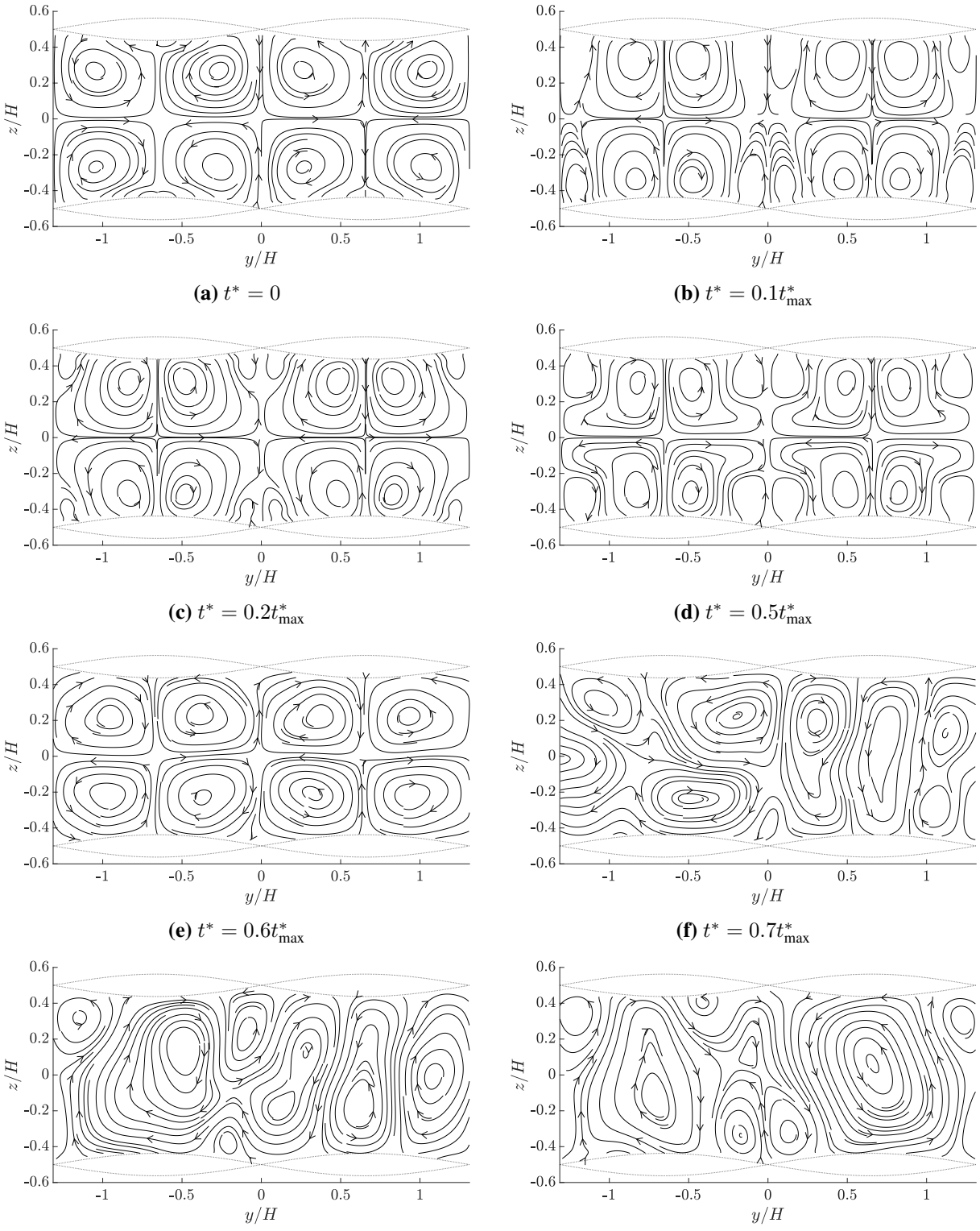
(g) $t^* = 0.8t^*_{\text{max}}$ **(h)** $t^* = t^*_{\text{max}}$

Figure 5.11: Case 45_0.26: Streamwise-averaged x vorticity contours for different times, where $Re_\tau = 45$, $\alpha_x = 0.26$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/6$. Time increases from left to right and top to bottom.

The vorticity magnitude is also large here compared to that in the beginning. At $t^* = 0.7t^*_{\text{max}}$ and $t^* = 0.8t^*_{\text{max}}$, the plots show irregular vortex structures with large amplitudes, consistent with the other cases in which the flow has turned unstable. Something strange now happens at $t^* = t^*_{\text{max}}$. The vortex structures are seemingly forming regular, almost alternating pattern.

This is consistent with the large scale structures in the last streamline plot in figure 5.12. In addition to the shape of the structures, it is also noted that the vorticity magnitude has significantly dropped from its maximum plotted values at around $t^* = 0.6t^*_{\text{max}}$, and in the same order of magnitude as in the initial condition.

It is suspected that the flow has been in the transition to turbulence in the course of the simulation, but is becoming laminar again towards the end. A similar strategy, used when further investigating the base case simulation for turbulence, is used for this case as well, looking at time instances of velocity magnitude contours in an arbitrary plane. The same plane as for the base case is chosen, and the results are presented in figure 5.14, and the axes is again exactly the same as in all the streamline plots. The first plot, at $t^* = 0.7t^*_{\text{max}}$, shows indications of a turbulent flow, with the same characteristic signs as for the base case velocity magnitude plots in figure 5.3. As time moves on, the velocity profile becomes a darker and red, and with a thinner centre. This is indications that the flow is becoming more laminar, as the centre-plane velocity is increasing and the gradients at the wall is decreasing, resulting in a velocity profile closer to a parabola again, instead of a square-like profile, typical of turbulent flow. It is noted that flow is not completely laminar, as the the maximum velocity still have a deficiency compared to the initial fully laminar solution, and it has not reached a steady solution.



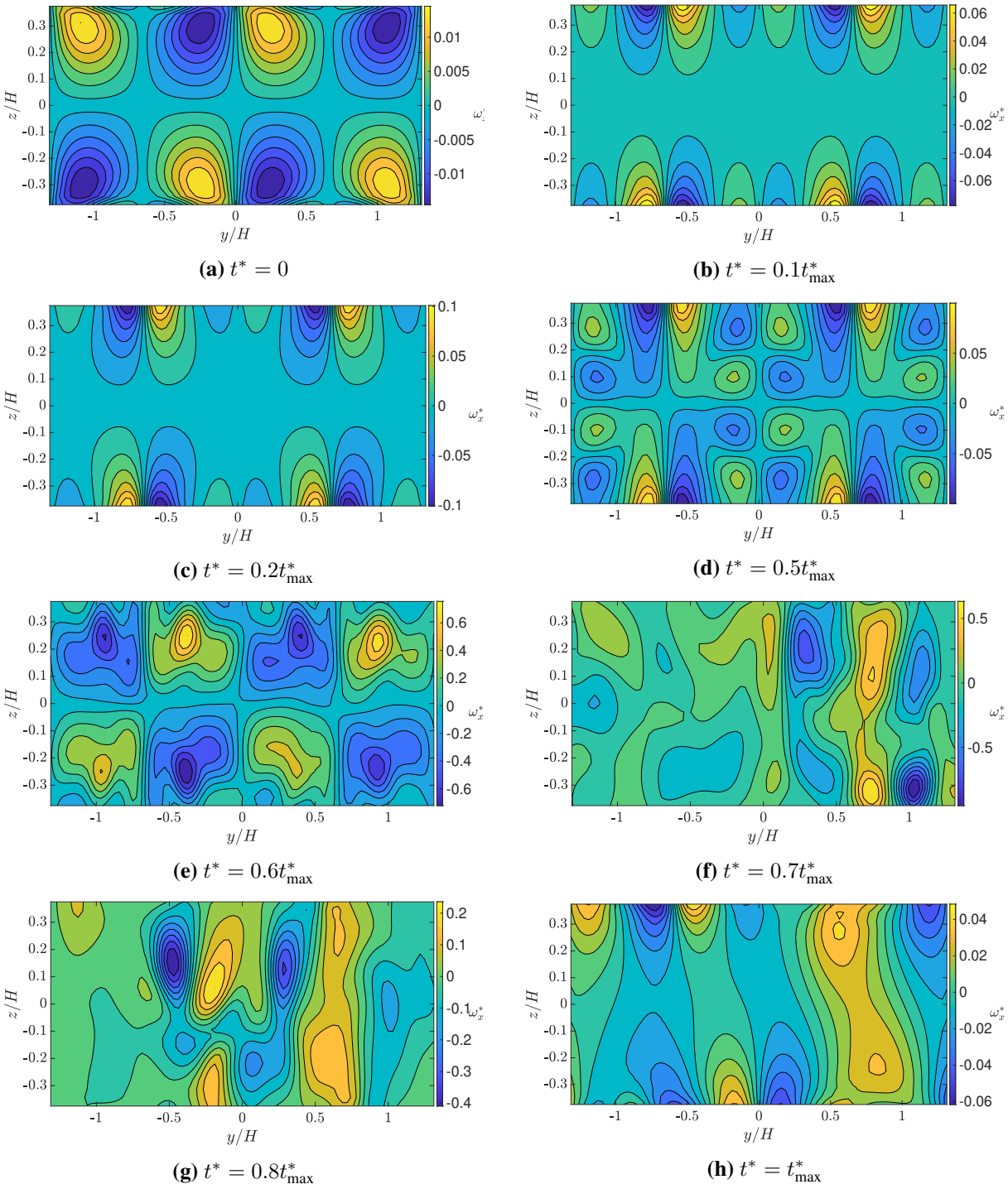
(a) $t^* = 0$ **(b)** $t^* = 0.1t^*_{\text{max}}$

(c) $t^* = 0.2t^*_{\text{max}}$ **(d)** $t^* = 0.5t^*_{\text{max}}$

(e) $t^* = 0.6t^*_{\text{max}}$ **(f)** $t^* = 0.7t^*_{\text{max}}$

(g) $t^* = 0.8t^*_{\text{max}}$ **(h)** $t^* = t^*_{\text{max}}$

Figure 5.12: Case 45_0.46: Streamwise-averaged normal velocity streamlines for different times, where $Re_\tau = 45$, $\alpha_x = 0.46$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/10$. Time increases from left to right and top to bottom.



(a) $t^* = 0$

(b) $t^* = 0.1t^*_{\text{max}}$

(c) $t^* = 0.2t^*_{\text{max}}$

(d) $t^* = 0.5t^*_{\text{max}}$

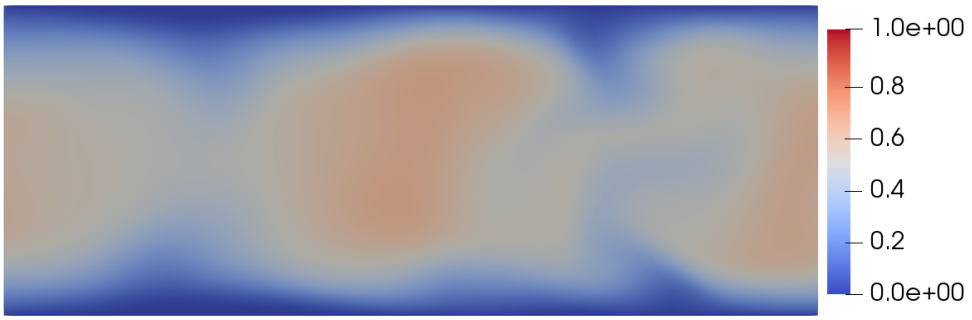
(e) $t^* = 0.6t^*_{\text{max}}$

(f) $t^* = 0.7t^*_{\text{max}}$

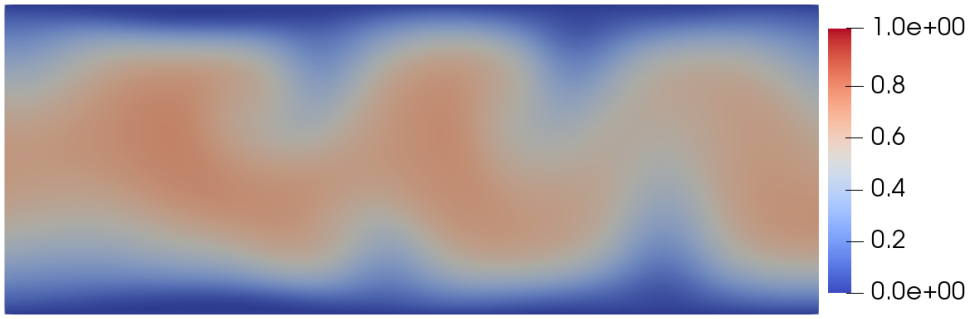
(g) $t^* = 0.8t^*_{\text{max}}$

(h) $t^* = t^*_{\text{max}}$

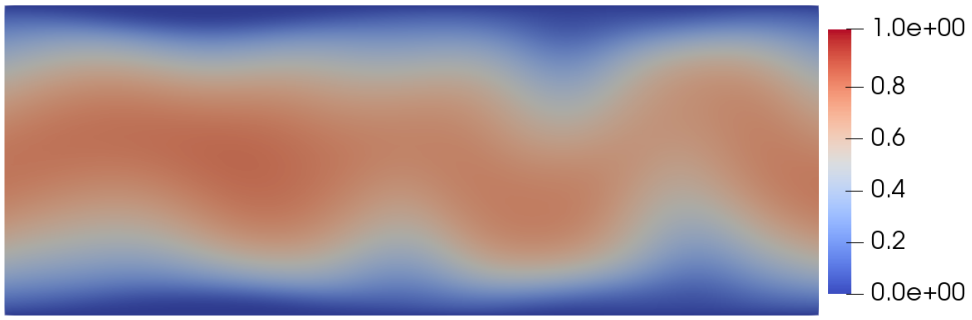
Figure 5.13: Case 45_0.46: Streamwise-averaged x vorticity contours for different times, where $Re_\tau = 45$, $\alpha_x = 0.46$, $\alpha_y = 0.15$, $a = 0.0625H$ and $\theta \approx \pi/10$. Time increases from left to right and top to bottom.



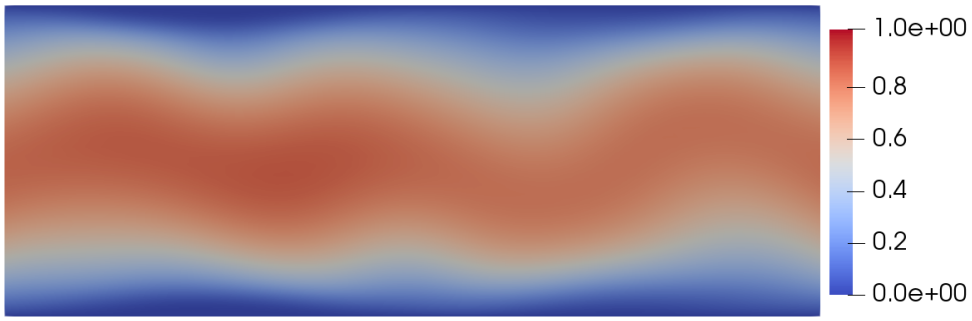
(a) $t^* = 0.7t^*_{\text{max}}$



(b) $t^* = 0.8t^*_{\text{max}}$



(c) $t^* = 0.9t^*_{\text{max}}$



(d) $t^* = t^*_{\text{max}}$

Figure 5.14: Case 45_0.46: Contour plots of velocity magnitude $|\mathbf{u}^*|$ in x-normal plane at $x = 0$ at different instances in time.

Chapter 6

Discussion

In this chapter, the results will be explained and the transient development for the different cases will be compared, all with the research questions stated in the introduction in mind.

The results from the base case simulation of $Re_\tau = 45$ and $\alpha_x = 0.36$ show that the flow is not able to sustain steady circulation structures, and that unstable flow behaviour becomes dominant at some point towards the end of the simulation. From figure 5.1, it is seen that the transition from stable Langmuir cells to unstable flow structures seems to follow a relatively ordered and regular time line, making it possible to describe the qualitative development leading up to the instability. First, the formation of oppositely rotating circulation cells emerges in line with the peaks and troughs of the wavy wall boundary. These cells continue to grow, as the original cells decreases in size, while simultaneously migrating towards the saddle points on the domain boundary. It is suspected that this behaviour is related to the high momentum paths in line of the saddle points, visualised in the velocity magnitude contours in figure 5.3b. Figure 5.1d, at $t^* = 0.5t^*_{\text{max}}$, or approximately 425, shows the first plot where the original circulation cells have completely vanished, giving a sense of the time scale of the cell break down. At this point, the flow in the centre of the channel has travelled approximately 120 times through the domain. When the original cell structures have vanished completely, the new pair of rotating cells are visibly unstable, and they begin to merge, indicated by the streamlines connecting in between the cells. From here, small transverse currents are observed, which might be responsible for advecting the circulation cells in the negative y direction, aligning them with the peaks and troughs, and with the saddle points. Signs of instability in the flow is observed, as more vortex structures with higher wavenumbers and vorticity are spawned inside the domain. Shortly after, transition to turbulence is suspected, since the regular structures break completely down, and characteristic signs of turbulence is observed in the velocity magnitude contours. This is in contrary to what was assumed in the introduction, where it was stated that turbulence was not suspected in the flow.

When looking at cases from the parameter study where friction Reynolds number is varied,

it is clear that the change of this parameter has had a substantial effect on the stability of the Langmuir Circulations. For a sufficiently small nominal friction Reynolds number of $Re_\tau = 35$ in Case $35\_0.36$, the circulation cells remain stable, with only small deviations from the steady state solution of $Re_\tau = 25$ in the initial condition. It is reasonable that a lower Reynolds number flow is more likely to remain stable, as the increased viscosity dampens out growth of potential instabilities. Increasing the nominal friction Reynolds number to 40, the circulation cells show signs of yielding, however, the Reynolds number is not high enough for the other set of oppositely rotating structures to expand into the bulk of the domain. The flow stabilises at this new equilibrium point. For the highest Re_τ in the parameter study of 50, the simulation results show very similar initial behaviour as for the base case, up until the new set of unstable rotating cells have taken over the domain. From here, the regular structures are observed to break down in transition to turbulence significantly sooner than it did for the base case. Higher vorticity magnitudes indicates that the unstable circulation structures are stronger for the higher the higher Reynolds number. With the same argument as for the stability of the lowest Reynolds number case, it is reasonable that the decrease in viscosity works destabilising on the flow, accelerating the break down of the cells.

Also the wave steepness of the wall boundary is observed to influence the transient development of the circulation cells significantly. The lower wave steepness of $\alpha = 0.26$ in Case $45\_0.26$ shows to have a stabilising effect on the Langmuir cells. Similar rotational structures as in Case $40\_0.36$ are visible throughout the simulation, with a close to stable solution where the cells are slightly shifted towards the saddle points in the walls. The stabilising effect of the low wave steepness is explained by a smoother alternating change in vertical velocity of the flow near the wall, due to less frequent forcing from the wavy structures. For Case $45\_0.46$, the opposite is observed. Here, the transition to unstable structures are accelerated by the increased α_x of 0.46. The same reasoning is used to explain this behaviour, i.e. that the higher frequency of change in vertical velocity is becoming too violent for stability to be maintained in the flow. As for the other cases where the circulation cells are observed to break down, a similar qualitative development is indicated by the results, where unstable, oppositely rotating cell pairs are spawned, filling the domain, before the structures break down in transition to turbulence. As indicated by figure 5.14 from the results, the solution seems to more laminar towards the end of the simulation. This could indicate that the solution is right in the transitional regime between laminar and turbulent, and that the unstable structures combined with the high wave steepness of the wall could slow down enough for the viscous effects to become dominant again. A longer simulation of this case would be needed to investigate the further development, to discover whether the flow would reach a stable solution in the end, or perhaps alternate between a regular and an irregular solution right at the transition point of the flow.

Table 5.1 from the results chapter shows that the observed true value of friction Reynolds number $Re_{\tau,t}$ was higher than the nominal friction Reynolds number for two of the cases,

namely Case $50\_0.36$ and Case $45\_0.46$. The assumption that the opposite should be true, is based on the flow being laminar. A transition to turbulence increases the spatial velocity gradients near the walls, which would increase the friction velocity, and therefore also the observed friction Reynolds number. These two cases happens to be the cases where the irregular patterns and unstable structures appear the earliest during the simulations, indicating that the higher $Re_{\tau,t}$ is related to the transition to turbulence. One additional case showed signs of transitioning to turbulent, which was the base case simulation. Table 5.1 shows that for this case, $Re_{\tau,t} < Re_{\tau}$. This is assumed to be because this particular case showed signs of transitioning only for the final part of the simulation. Since the true friction Reynolds numbers are based on y^+_{avg}, it is suspected that the highest spatial average of y^+ in time has not been able to grow as much as in the two other unstable cases. This would explain why a lower $Re_{\tau,t}$ is observed.

As mentioned in the results, the maximum Courant numbers shown in 5.1 have become somewhat higher than a comfortable level. The mesh refinement study in section 4.4.2 indicated that a $C_{\text{max}} \approx 1.09$ was still acceptable, but since several of the cases produces flows that are no longer close to uniform and unidirectional near the mid plane, where C_{max} is found. Since the solver is implicit in time, no stability issues arises from this. However, if more time and computational resources were available, the simulations would have been run again with a smaller time step, to verify that the results from the simulations would still be the same in the time periods where the unstable structures are present.

Chapter 7

Conclusion

The objective of this master's thesis has been to use CFD software to simulate and study the transient behaviour of boundary layer flows with sinusoidally modulated wall boundaries. For the right flow conditions, considering both the flow properties and the wavy wall geometry, the work of (Akselsen and Ellingsen 2019) has shown that distinct circulation pairs of counter rotating circulation cells are formed in the flow. The rotational cells are dynamically equivalent to Langmuir Circulations, which can be observed below the water surface on windy days. These circulation structures can become unstable and break down if the flow conditions are altered sufficiently. It is this transition to unstable flow that has been the focus of interest for the present work. A CFD model has been created in OpenFOAM 7, an open source `C++` toolbox for fluid flow simulations. The geometry and mesh are created in ANSYS, and imported into OpenFOAM. Several steps have been taken to verify the developed model, and to validate the CFD code by performing a test case simulation, the laminar boundary layer development over a flat plate, which has a known solution. The simulation results are also compared with existing results where they exist. High performance computer resources on Vilje, provided by Uninett Sigma2, have been utilised to carry out the computations. A study has been conducted for choosing an optimal number of CPU cores, where both computational time and the cost of CPU-hours have been considered.

Three research questions were stated in the introduction chapter and the goal of the analysis has been to provide answers for all of them. The questions are stated as

- In what way does the circulation cells evolve to become unstable?
- What is the time scale for the circulation cells to break down?
- How does different flow and domain parameters affect the instability of the cells?

To answer the first two questions, a base case is defined with nominal friction Reynolds number $Re_\tau = 45$ and a wall wave steepness α_x in the streamwise direction is set to 0.36.

An initial condition from a steady state simulation of $Re_\tau = 25$, where the Langmuir Circulations are known to be present and stable. Transient simulations have shown that the circulation structures break down because new pairs of rotational cells of opposite rotation emerge in line with the peaks and troughs of the wall boundaries, as shown from the streamlines in figure 5.1. These new cell becomes dominant, and the original cells disappear from the domain. The new structures however are not stable, and merges together as can be seen in the figure. Higher wave number vortices are spawned in the domain, breaking down the regular structures in the flow. Because of the irregular flow structures combined with the flattening out of the velocity profile, it is concluded that the flow is in fact transitioning to becoming turbulent.

The Langmuir circulation cells breaks down continuously from the beginning of the simulation, and at $t^* \approx 425$, no more signs of the the original cells are visible, shown in figure 5.1d. This corresponds to approximately 120 times that the flow in the centre has travelled through the domain.

To answer the final question, a parameter study is conducted, where the Re_τ and α_x are varied, and the simulation results are compared. The results show that a lower friction Reynolds number of $Re_\tau = 35$ acts stabilising on the flow, eliminating the break down completely. Increasing the Reynolds number leads to more instability and a more rapid break down of the Langmuir cells, as can be seen in figure 5.8 for the highest Re_τ of 50. Variation of α_x is also show to have an impact on the stability of the circulations. A lower wave steepness of $\alpha_x = 0.26$ has a stabilising effect, as less violent alternating vertical forcing is inflicted on the flow while it moves over the wavy boundaries. The opposite is true for the the higher wave steepness of $\alpha_x = 0.46$, where the increased frequency of the alternating motion of the fluid moving peaks and troughs, causes the circulation structures to break down and become unstable faster.

There are several possibilities for further research on the topic of Langmuir Circulations in flows over undulating walls. Time averaged data of the turbulent flow could be studied, to investigate whether there still exists a mean circulation in the flow. It would also be interesting to look at the variation of other dimensionless parameters in the flow problem, such as the amplitude a/H, and how it affects the Langmuir cells and their stability. By adding the energy equation to the model, the effects on the temperature mixing caused by the circulation structures could studied, which could be interesting for industrial purposes. Experimental data on the problem would also be highly beneficial as a means of validating the CFD results, however, this would probably be difficult, due to the long channel length needed for the circulations to form.

Bibliography

Akselsen, A. and Ellingsen, S. (2019). "Topography-driven Langmuir circulation in boundary layers". In: *Under consideration for: Journal of Fluid Mechanics*.

Ali, Z., Tucker, P. G., and Shahpar, S. (Apr. 2017). "Optimal mesh topology generation for CFD". In: *Computer Methods in Applied Mechanics and Engineering* 317, pp. 431–457.

Amdahl, G. (1967). "Validity of the single processor approach to achieving large scale computing capabilities". In: *Conference Proceedings* 30, pp. 483–485.

ANSYS (2009a). *ANSYS FLUENT Theory Guide*. URL: `https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/main_pre.htm`.

— (2009b). *ANSYS FLUENT User Guide*. URL: `https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/main_pre.htm`.

Brostrøm, A. (2019). "Numerical simulation of circulation in boundary layers over sinusoidally modulated walls". In: *Project thesis*.

CFD-Direct (n.d.). *About OpenFOAM*. URL: `https://cfd.direct/openfoam/about/`.

Chan, D. et al. (2018). "Secondary motion in turbulent pipe flow with three-dimensional roughness". In: *Journal of Fluid Mechanics* 854, pp. 5–33.

Chen, S. and Doolen, G. (1998). "Lattice Boltzmann Method for Fluid Flows". In: *Annual Review of Fluid Mechanics* 30, pp. 329–364.

Craik, A. D. D. (1970). "A wave-interaction model for the generation of windrows". In: *Journal of Fluid Mechanics* 41, pp. 801–821.

— (1977). "The generation of Langmuir circulations by an instability mechanism". In: *Journal of Fluid Mechanics* 81.2, pp. 209–223.

Craik, A. D. D. and Leibovich, S. (1976). "A rational model for Langmuir circulations". In: *Journal of Fluid Mechanics* 63, pp. 401–426.

Faller, A. (1964). "The angle of windrows in the ocean". In: *Tellus* 16.3, pp. 363–370.

Faller, A. and Caponi, E. (1978). "Laboratory Studies of Wind-Driven Langmuir Circulations". In: *Journal of Geophysical Research* 83.C7, pp. 3617–3633.

Gong, W., Taylor, P. A., and Dörnbrack, A. (1996). "Turbulent boundary-layer flow over fixed aerodynamically rough two-dimensional sinusoidal waves". In: *Journal of Fluid Mechanics* 312, pp. 1–37.

Hirt, C. W. and Nichols, B. D. (1981). "Volume of fluid (VOF) method for the dynamics of free boundaries". In: *J. Comput. Phys.; (United States)* 39:1.

Kukudzhanov, V. N. and Zhurov, A. (2013). *Numerical Continuum Mechanics*. De Gruyter, p. 254. URL: https://www.degruyter.com/view/title/122898.

Kukulka, T. et al. (2009). "Significance of Langmuir circulation in upper ocean mixing: Comparison of observations and simulations". In: *Geophysical Research Letters* 36, pp. 119–123.

Lal, S. A. and Paul M, N. (Oct. 2014). "An Accurate Taylors Series Solution with High Radius of Convergence for the Blasius Function and Parameters of Asymptotic Variation". In: *Journal of Applied Fluid Mechanics* 7, pp. 557–564.

Langmuir, I. (1938). "Surface motion of water induced by wind". In: *American Association for the Advancement of Science* 87.2250, pp. 119–123.

Leibovich, S. (1977). "Convective instability of stably stratified water in the ocean". In: *Journal of Fluid Mechanics* 82.3, pp. 561–581.

— (1983). "The Form and Dynamics of Langmuir Circulations". In: *Annual Review of Fluid Mechanics* 15, pp. 391–427.

Luettich, R. (2018). *Coastal Hazards Related to Storm Surge*. MDPI AG. ISBN: 9783038427117. URL: https://books.google.no/books?id=B%5C_pQDwAAQBAJ.

Müller, B. (Oct. 2018). *Introduction to Computational Fluid Dynamics - Lecture notes for the course Computational Heat and Fluid Flow*.

NTNU-HPC-Group (n.d.). *About Vilje*. URL: https://www.hpc.ntnu.no/ntnu-hpc-group/vilje/about-vilje.

OpenFOAMWiki (2014). *PISO*. URL: https://openfoamwiki.net/index.php/PISO.

— (2019). *SimpleFoam*. URL: https://openfoamwiki.net/index.php/SimpleFoam.

Orszag, S. (1971). "Accurate solution of the Orr-Sommerfeld stability equation". In: *Journal of Fluid Mechanics* 50, pp. 659–703.

Phillips, W. R. C., Wu, Z., and Lumley, J. L. (1996). "On the formation of longitudinal vortices in a turbulent boundary layer over wavy terrain". In: *Journal of Fluid Mechanics* 326, pp. 321–341.

Rathakrishnan, E. (2019). *Applied Gas Dynamics*. Wiley. URL: https://books.google.no/books?id=esaKDwAAQBAJ.

Scott, J. et al. (1969). "On the mechanism of Langmuir Circulations and their role in epilimnion mixing". In: *Limnology and Oceanography* 14.4, pp. 493–503.

Sigma2 (n.d.). *About Sigma2*. URL: https://www.sigma2.no/about-sigma2.

Slater, J. W. (2008a). *Overview of CFD Verification and Validation*. URL: https://www.grc.nasa.gov/WWW/wind/valid/tutorial/overview.html.

— (2008b). *Uncertainty and Error in CFD Simulations*. URL: https://www.grc.nasa.gov/WWW/wind/valid/tutorial/errors.html.

Thompson, J., Soni, B., and Weatherill, N. (1998). *Handbook of Grid Generation*. CRC Press. URL: https://books.google.no/books?id=ImaDT6ijKq4C.

Thurnherr, A. (2002). *Wind Rows*. URL: https://www.ldeo.columbia.edu/~ant/Langmuir.html.

Appendix

A OpenFOAM configuration files

A.1 - p

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            cyclic;
    }

    outlet
    {
        type            cyclic;
    }

    top
    {
        type            zeroGradient;
    }

    bottom
    {
        type            zeroGradient;
    }

    left
    {
        type            cyclicAMI;
    }

    right
    {
        type            cyclicAMI;
    }
}

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

A.2 - U

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0.1925 0 0);

boundaryField
{
    inlet
    {
        type            cyclic;
    }

    outlet
    {
        type            cyclic;
    }

    top
    {
        type            noSlip;
    }

    bottom
    {
        type            noSlip;
    }

    left
    {
        type            cyclicAMI;
    }

    right
    {
        type            cyclicAMI;
    }
}

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

A.3 - boundary

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

6
(
    inlet
    {
        type            cyclic;
        inGroups        List<word> 1(cyclic);
        nFaces          11340;
        startFace       2720223;
        matchTolerance  0.0001;
        transform       unknown;
        neighbourPatch  outlet;
    }
    outlet
    {
        type            cyclic;
        inGroups        List<word> 1(cyclic);
        nFaces          11340;
        startFace       2731563;
        matchTolerance  0.0001;
        transform       unknown;
        neighbourPatch  inlet;
    }
    right
    {
        type            cyclicAMI;
        inGroups        List<word> 1(cyclicAMI);
        nFaces          3645;
        startFace       2742903;
        matchTolerance  0.0001;
        transform       translational;
        neighbourPatch  left;
        separationVector (0 252 0);
        method          faceAreaWeightAMI;
    }
    left
    {
        type            cyclicAMI;
        inGroups        List<word> 1(cyclicAMI);
        nFaces          3645;
        startFace       2746548;
        matchTolerance  0.0001;
        transform       translational;
        neighbourPatch  right;
        separationVector (0 -252 0);
        method          faceAreaWeightAMI;
    }
    top
    {
        type            wall;
        inGroups        List<word> 1(wall);
        nFaces          20412;
        startFace       2750193;
```

A.3 - boundary

```
    }
    bottom
    {
        type            wall;
        inGroups        List<word> 1(wall);
        nFaces          20412;
        startFace       2770605;
    }
)

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

A.4 - transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 0.00456;

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

A.5 - `turbulenceProperties`

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType laminar;

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

A.6 - `fvSchemes` for steady state solver

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwind grad(U);
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}


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

A.7 - `fvSchemes` for transient solver

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
      default         backward;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;

    div(phi,U)      Gauss linear;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;

}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}


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

A.8 - `fvSolution` for steady state solver

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.001;
        smoother        GaussSeidel;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0.001;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
    consistent      yes;

    residualControl
    {
        p               1e-6;
        U               1e-6;
    }
    pRefCell    0;
    pRefValue   0;
}

relaxationFactors
{
    equations
    {
        U               0.9;
        ".*"            0.9;
    }
}


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

A.9 - `fvSolution` for transient solver

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       0;
        relTol          0.01;
        smoother        GaussSeidel;
        cacheAgglomeration no;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "pcorr.*"
    {
        $p
        tolerance       0.02;
        relTol          0;
    }
    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0.1;
    }

    UFinal
    {
        $U;
        tolerance       1e-06;
        relTol          0;
    }
}

PIMPLE
{
    correctPhi          yes;
    nOuterCorrectors    50;
    nCorrectors         1;
    nNonOrthogonalCorrectors 0;
    pRefCell    0;
    pRefValue   0;

    outerCorrectorResidualControl
    {
        "(U|p)" { relTol 0.001; tolerance 1e-6; }
```

A.9 - `fvSolution` for transient solver

```
    }
}

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


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

A.10 - `controlDict` for steady state solver

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         5000;

deltaT          1;

writeControl    timeStep;

writeInterval   1000;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

functions
{
        force{
                type        forces;
                libs        ("libforces.so");
                rho         rhoInf;
                rhoInf      1.0;    // Fluid density
                patches     (bottom top);

                CofR        (0 0 0);
                pitchAxis   (0 1 0);
        }
}


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

A.11 - `controlDict` for transient solver

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pimpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         100000;

deltaT          4;

writeControl    timeStep;

writeInterval   500;

purgeWrite      0;

writeFormat     binary;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  no;

maxCo           0.5;


functions
{
        #includeFunc residuals
        force{
                type        forces;
                libs        ("libforces.so");
                writeControl    timeStep;
                writeInterval   250;
                rho         rhoInf;
                rhoInf      1.0;    // Fluid density
                patches     (bottom top);

                CofR        (0 0 0);
        }

}
```

A.12 - `fvOptions`

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

momentumSource
{
    type      vectorSemiImplicitSource;

    active          true;
    selectionMode   all;

    volumeMode      specific;
    injectionRateSuSp
    {
        U    ((1.0288e-05 0 0) 0);
    }
}

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

A.13 - decomposeParDict

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      decomposeParDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 64;

method          hierarchical;

hierarchicalCoeffs
{
    n               (16 4 1);
    delta           0.001;
    order           xyz;
}


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

A.14 - `probes` - probe definitions

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
-------------------------------------------------------------------------------
Description
    Writes out values of fields from cells nearest to specified locations.

\*---------------------------------------------------------------------------*/

#includeEtc "caseDicts/postProcessing/probes/probes.cfg"

writeInterval 100;

interpolationScheme     cellPoint;

fields (p U);
probeLocations
(
    (0 0 0)
    (8.75 -21 16)
    (8.75 21 12)
);

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

A.15 - `singleGraph` - line probe definitions (1 of 5)

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
-------------------------------------------------------------------------------
Description
    Writes graph data for specified fields along a line, specified by start
    and end points.

\*---------------------------------------------------------------------------*/

start   (8.75 -21 -17.9);
end     (8.75 -21 17.9);
fields  (U);

#includeEtc "caseDicts/postProcessing/graphs/sampleDict.cfg"

setFormat csv;

setConfig
{
    type lineUniform;
    axis xyz;
    nPoints 100;
}

#includeEtc "caseDicts/postProcessing/graphs/graph.cfg"

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

B Bash scripts

B.1 - Script for running jobs on Vilje

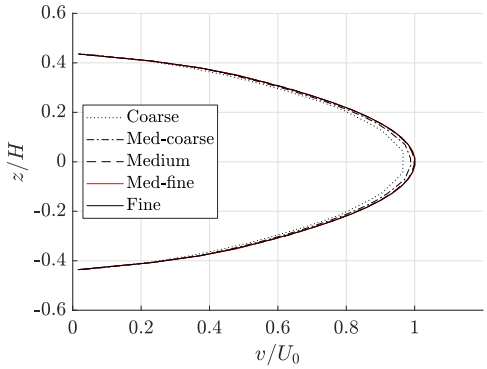
```
#!/bin/bash
#####################################################
#
#  OpenFOAM job script
#
#####################################################
#
#PBS -N re_45
#PBS -A <user-id>
#PBS -l select=4:ncpus=32:mpiprocs=16
#PBS -l walltime=10:30:00

# Load modules needed for job
module purge
module load gcc/6.2.0
module load mpt/2.14
module load openfoam/7

# Go to working directory
cd /work/andbro/OpenFOAM/andbro-7/run/result_files/re_45

# Run OpenFOAM in the working directory
mapFields ../steady_re_files/ -sourceTime latestTime -consistent
decomposePar >log.decomposePar
mpiexec_mpt pimpleFoam -parallel 2>&1 | tee log.solver
reconstructPar -newTimes >log.reconstructPar
rm -r processor*
```

B.1 - Script for extracting data for mesh refinement study

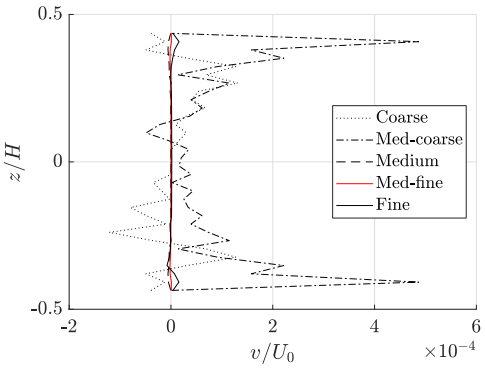
```
#!/bin/bash

# Bash script for automating data extraction for mesh refinement study

#Remove potential existing files
rm -r postProcessing/probes
rm -r postProcessing/force/0/*F.dat
rm -r postProcessing/yPlus
rm -r postProcessing/flowRatePatch\(name\=inlet\)
rm -r postProcessing/flowRatePatch\(name\=outlet\)
rm -r postProcessing/singleGraph*

#Probe data
postProcess -func probes
cat postProcessing/probes/0/U | cut -d '#' -f1 | cut -d '(' -f2 | cut -d ')' -f1 >
postProcessing/probes/0/uprobe1
cat postProcessing/probes/0/U | cut -d '#' -f1 | cut -d '(' -f3 | cut -d ')' -f1 >
postProcessing/probes/0/uprobe2
cat postProcessing/probes/0/U | cut -d '#' -f1 | cut -d '(' -f4 | cut -d ')' -f1 >
postProcessing/probes/0/uprobe3
cat postProcessing/probes/0/p | cut -d '#' -f1 > postProcessing/probes/0/pprobe

#Get probe locations
cat postProcessing/probes/0/p | grep '(' | cut -d '(' -f2 | cut -d ')' -f1 >
postProcessing/probes/0/probe_pos.dat
echo "Probe data complete"

#Force data
cat postProcessing/force/0/forces.dat | cut -d '(' -f4 | cut -d ' ' -f1 >
postProcessing/force/0/viscous_F.dat
cat postProcessing/force/0/forces.dat | cut -d '(' -f3 | cut -d ' ' -f1 >
postProcessing/force/0/pressure_F.dat
echo "Force data complete"

#yPlus data
pimpleFoam -postProcess -func yPlus
mv postProcessing/yPlus/0/yPlus.dat postProcessing/yPlus/0/yPlus_0.dat
echo "yPlus data complete"

#Flowrate data
postProcess -func "flowRatePatch(name=outlet)"
postProcess -func "flowRatePatch(name=inlet)"
echo "Flowrate data complete"

#Line probe data
postProcess -func singleGraph1
postProcess -func singleGraph2
postProcess -func singleGraph3
postProcess -func singleGraph4
postProcess -func singleGraph5
echo "Line probe data complete"
```

C Additional velocity profiles from mesh refinement study

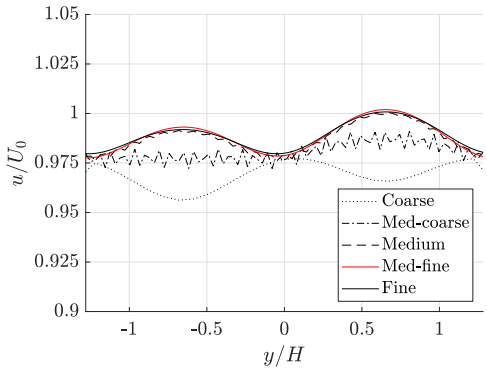
C.1 - Steady state mesh refinement study



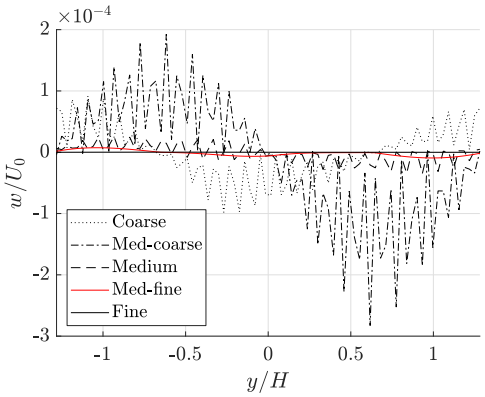
(a) x velocity



(b) y velocity



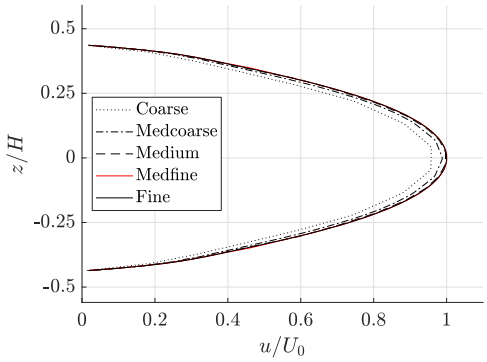
(c) x velocity



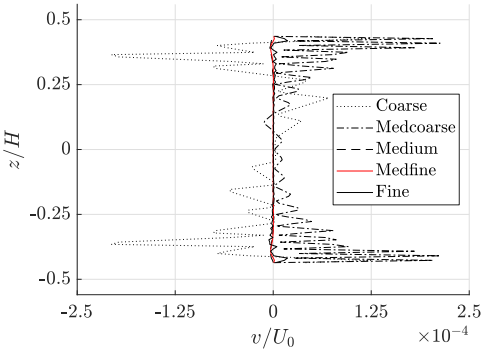
(d) y velocity

Steady state: Probed velocity profiles in different parts of the domain at final time step: (a) & (b) Vertical line in the z direction from wall to wall in between wave peaks, (c) & (d) Horizontal line in the y-direction at $x = z = 0$, i.e. in the centre plane of the channel. Accepted mesh in solid red.

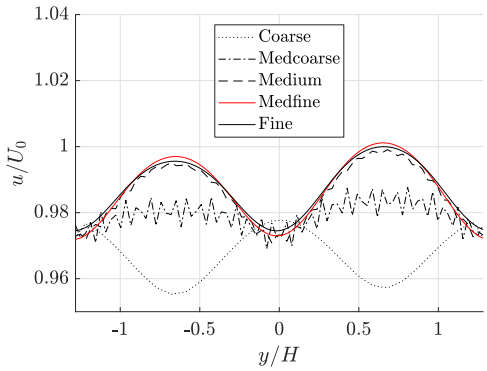
C.2 - Transient mesh study for different meshes and equal time step



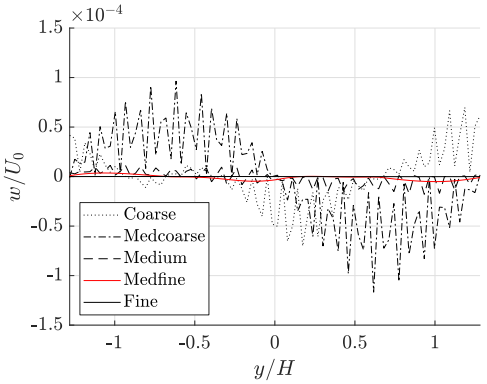
(a) x velocity



(b) y velocity



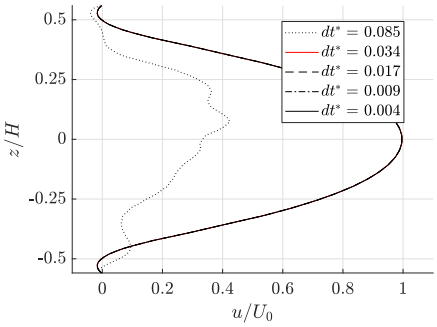
(c) x velocity



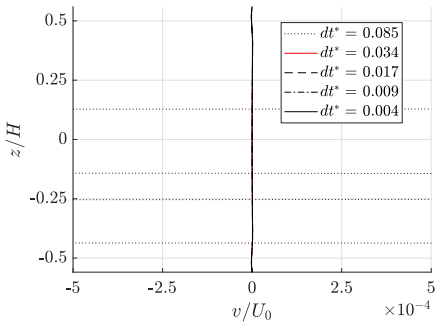
(d) z velocity

Equal time step: Probed velocity profiles in different parts of the domain at final time step: (a) & (b) Vertical line in the z direction from wall to wall in between wave peaks, (c) & (d) Horizontal line in the y-direction at $x = z = 0$, i.e. in the centre plane of the channel. Accepted mesh in solid red.

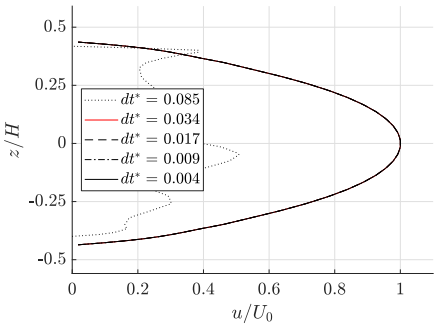
C.3 - Temporal convergence study for equal mesh and different time steps



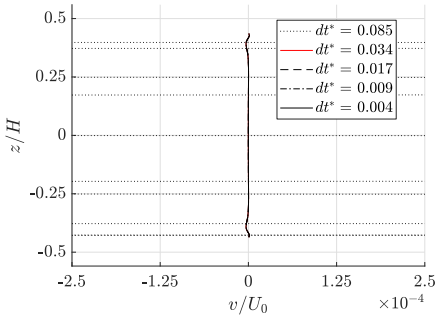
(a) x velocity



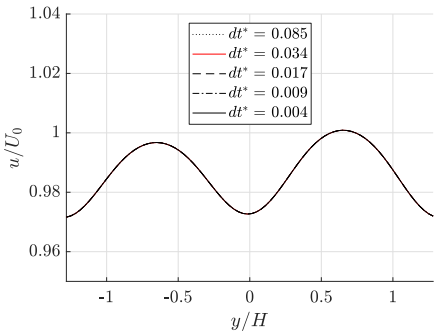
(b) y velocity



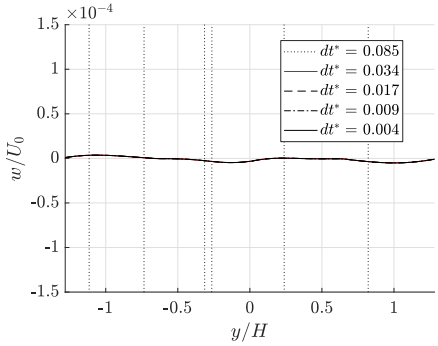
(c) x velocity



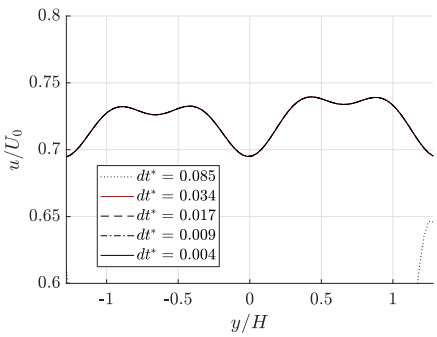
(d) y velocity



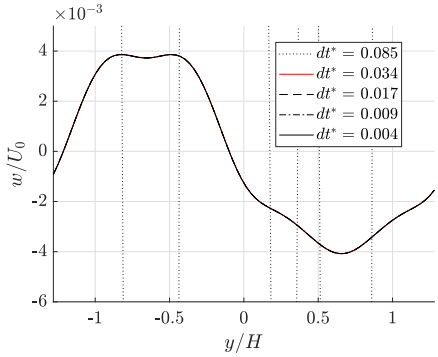
(e) x velocity



(f) z velocity



(g) x velocity



(h) z velocity

Equal mesh: Probed velocity profiles in different parts of the domain at final time step: (a) & (b) Vertical line in the z direction from wall to wall in between wave troughs, (c) & (d) Vertical line in the z direction from wall to wall in between wave peaks, (e) & (f) Horizontal line in the y-direction at $x = z = 0$, i.e. in the centre plane of the channel, (g) & (h) Horizontal line in the y-direction at $x = 0$ and $z/H = 0.25$, i.e. half way in between the centre plane and upper the mean wall location. Accepted time step in solid red.

D MATLAB scripts

D.1 - Script for postprocessing of the raw CFD data to a uniform grid

```
% Clean command window and workspace
clc;
clear;

% Plot settings
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');
set(0,'defaultAxesFontSize',16)

% This is set to true for the first time the script is run, to read
 new data
% from the csv-files
do_read_new = false;

% toggle plotting
do_plot = false;

if do_read_new
    % Run through all the different cases
    for caseNum = 1:6
        % Give the user feedback on progression
        fprintf('case number: %d\n\n',caseNum)

        % Get the case name to load, and the prefix for writing files
        switch caseNum
            case 1
                caseName = 're_files/re_45';
                saveName = 're_45';
                lx = 105;
            case 2
                caseName = 're_files/re_35';
                saveName = 're_35';
                lx = 105;
            case 3
                caseName = 're_files/re_40';
                saveName = 're_40';
                lx = 105;
            case 4
                caseName = 're_files/re_50';
                saveName = 're_50';
                lx = 105;
            case 5
                caseName = 'alpha_files/alpha_0.26';
                saveName = 'alpha_0.26';
                lx = 144;
            case 6
                caseName = 'alpha_files/alpha_0.46';
                saveName = 'alpha_0.46';
                lx = 81;
        end
```

D.1 - Script for postprocessing of the raw CFD data to a uniform grid

```
        % Define domain
        nx_periods = 3;
        ly = 84;
        kx = nx_periods*2*pi/lx;
        ky = 2*pi/ly;
        H = 32;
        x0 = -(lx)/2:1:(lx)/2;
        y0 = (-ly/2):1:(ly/2);
        z0 = -18:1:18;

        % Node indices
        ni1 = length(x0);
        nj1 = length(y0);
        nk1 = length(z0);

        % Construct a Cartesian grid in the domain
        [X_3d,Y_3d,Z_3d] = meshgrid(x0,y0,z0);
        [Y_2d,Z_2d] = meshgrid(y0,z0);

        % Define which files to be read, where this number indicates
 the
        % index of the files written from ParaView
        loadNumbers = [0,5,10,25,30,35,40,50];

        % Allocate matricies for averaged velocities and maximum
 velocity
        clear v_mean w_mean omega_mean;
        [v_mean,w_mean,omega_mean] =
 deal(zeros(nk1,nj1,length(loadNumbers)));
        u_max_vel = zeros(1,length(loadNumbers));

        % Loop trough all time instances
        cind = 1;
        for m = loadNumbers
            % Give the user feedback on progression
            fprintf('Reading %1.1d of %1.1d files\n
\n',cind,length(loadNumbers));
            fprintf("Reading raw data from CSV-files...\n");

            % Get filenumber to read
            filenumber = m;
            filenumber_str = num2str(filenumber);

            % Read files
            %CSV file structure:
            %P,     x-vel, y-vel, z-vel, x-vort,y-vort, z-vort, x-pos,
 y-pos, z-pos
            raw_data = csvread(['~/OpenFOAM/andbro-7/run/
result_files/',caseName,...
                '/datafiles0.',filenumber_str,'.csv'],1,0);

            % Get max velocity from the time instance
            u_max_vel(cind) = max(raw_data(:,2));
```

D.1 - Script for postprocessing of the raw CFD data to a uniform grid

```
            fprintf("COMPLETE: Reading raw data from CSV-files\n\n");

            % Interpolating data to uniform grid
            fprintf("Interpolating data to uniform grid...\n\n");

            % y-velocity
            fprintf("Interpolating v-velocity...\n");
            v_interpolation_func =
scatteredInterpolant(raw_data(:,8),raw_data(:,9),...
                raw_data(:,10),raw_data(:,3));
            v_interpolated = v_interpolation_func(X_3d,Y_3d,Z_3d);
            fprintf("COMPLETE: Interpolating v-velocity\n\n");

            % z-velocity
            fprintf("Interpolating w-velocity...\n\n");
            w_interpolation_func =
scatteredInterpolant(raw_data(:,8),raw_data(:,9),...
                raw_data(:,10),raw_data(:,4));
            w_interpolated = w_interpolation_func(X_3d,Y_3d,Z_3d);
            fprintf("COMPLETE: Interpolating w-velocity\n\n");

            % x-veorticity
            fprintf("Interpolating x-vorticity...\n\n");
            omega_interpolation_func =
scatteredInterpolant(raw_data(:,8),raw_data(:,9),...
                raw_data(:,10),raw_data(:,5));
            omega_interpolated =
omega_interpolation_func(X_3d,Y_3d,Z_3d);
            fprintf("COMPLETE: Interpolating data to uniform grid\n");

            % Remove nodes that fall outside of the domain
            for j = 1:length(y0)
                for i = 1:length(x0)
                    for k = 1:length(z0)
                        if(Z_3d(j,i,k) < -H/2 +
2*sin(kx*x0(i))*sin(ky*y0(j)) || Z_3d(j,i,k)...
                                > H/2 - 2*sin(kx*x0(i))*sin(ky*y0(j)))
                            v_interpolated(j,i,k) = nan;
                            w_interpolated(j,i,k) = nan;
                            omega_interpolated(j,i,k) = nan;
                        end
                    end
                end
            end


            % Compute streamwise-averaged values for all quantities
            v_mean(:,:,cind) =
squeeze(mean(v_interpolated(:,1:end-1,:),2))';
            w_mean(:,:,cind) =
squeeze(mean(w_interpolated(:,1:end-1,:),2))';
            omega_mean(:,:,cind) =
squeeze(mean(omega_interpolated(:,1:end-1,:),2))';
```

D.1 - Script for postprocessing of the raw CFD data to a uniform grid

```
            if cind == length(loadNumbers)
                clc;
                fprintf('COMPLETE: Loading of CSV-files\n\n')
            end
            cind = cind + 1;
        end %for m = loadNumbers

        % Save the workspace so that the interpolation only has to be
 done once
        save([saveName,'.mat'])
        clc;
    end %for caseNum
else
    % This replaces the entire previous section after the first run-
through
    % Load the specified case for plotting. Only one case is plottet
 at a
    % time for easier inspection of the figures
    caseToLoad = 're_45';
    load([caseToLoad,'.mat']);
end %if do_read_new


% Plotting
if do_plot
    close all
    for i = 1:length(loadNumbers)

        % Vorticity plot
        skip = 7;
        z_include = skip:nk1-(skip-1);
        figure('units','normalized','outerposition',[0 0 1 1])
        c_limits =
 round(linspace(min(min(squeeze(omega_mean(z_include,:,i))*H/
u_max_vel(i))),...
            max(max(squeeze(omega_mean(z_include,:,i))*H/
u_max_vel(i))),14),5);
        [c,h] = contourf(Y_2d(z_include,:)/H,Z_2d(z_include,:)/
H,squeeze(omega_mean(z_include,:,i))...
            *H/u_max_vel(i),c_limits);
        xlabel('$y/H$')
        ylabel('$z/H$')
        %caxis([-0.02 0.02])
        axis tight
        set(gca,'FontSize',40)
        colorb_s = colorbar;
        colorb_s.Label.Interpreter = 'latex';
        colorb_s.FontSize = 40;
        ylabel(colorb_s,'$\omega_x^*$','Rotation',0,'FontSize',40)

        % Save figures using export_fig
        export_fig(sprintf('../Figures/Results/%s/%s', caseToLoad,
[caseToLoad,'_','vorticity','_',num2str(loadNumbers(i))]),...
            '-transparent','-eps');
```

D.1 - Script for postprocessing of the raw CFD data to a uniform grid

```
        % Streamslice plot
        figure('units','normalized','outerposition',[0 0 1 1])
        a = streamslice(Y_2d/H,Z_2d/H,squeeze(v_mean(:,:,i))/
u_max_vel(i),squeeze(w_mean(:,:,i))/u_max_vel(i),1.2);
        set(a,'Color','k')
        hold on
        plot(y0/H,2/H*[-1,1,-1,1]'.*sin(2*pi/ly*y0) + [H/2,H/2,-H/2,-
H/2]'/H,'k:')
        xlabel('$y/H$')
        ylabel('$z/H$')
        axis tight
        axis([-ly/2/H,ly/2/H,-0.6001 0.6001])
        set(gca,'FontSize',40)

        % Save figures using export_fig
        export_fig(sprintf('../Figures/Results/%s/%s', caseToLoad,
[caseToLoad,'_','streamline','_',num2str(loadNumbers(i))]),...
            '-transparent','-eps');
    end % for i = 1:length(loadNum)
end %if do_plot
```

Published with MATLAB® R2019b

Andreas Brostrøm

Numerical simulation of circulation in boundary layers over sinusoidally modulated walls

NTNU
Norwegian University of
Science and Technology