Generated Sound Waves in Corrugated Pipes

Master's thesis in applied mathematics

by

Arve Lillejord

June 1, 2012

Department of Mathematics
University of Bergen
Norway

Prologue

When I set out to do my master's thesis I had not had any experience in computational fluid dynamics except for theory on the Navier-Stokes equations. This meant that a large part of my master's would be dedicated to learning the fundamentals of the discretisation of the Navier-Stokes equations, turbulence modelling and mesh generation. In addition to that, OpenFOAM is notorious for having a steep learning curve and a little sparse documentation. However, by taking part in the OpenFOAM community and studying the source code I was able to learn a lot.

As I progressed in learning OpenFOAM, my work flow began to be more structured. I would first generate the geometry, then a great deal of effort was put into meshing this geometry. This was in the beginning done with the third party software Salomè [1]. Then I would set up the case parameters to the best of my knowledge, such as boundary conditions and solver settings. As I learnt more about meshing and OpenFOAM I could gradually improve these parameters. From mesh to a complete simulation is a rather lengthy process, which made this a big part of my work. In computational fluid dynamics, experience with the tools you are working with is very valuable in order to get the most out of any simulation. Unfortunately, time was restricted as I had a deadline for my master thesis. When solving a large eddy simulation problem, the mesh requirements makes the required computing time massive. Luckily, at the University of Bergen we have one of the most powerful computing resources in Norway which I was given permission to use. This has given me hands on experience with HPC (high performance computing).

I would like to extend my sincere gratitude to my supervisors Guttorm Alendal and Gunnar Furnes for their guidance, support and their enthusiasm towards this project, which in turn motivated my work and wanting to continue to pursue this project even after finishing my master's degree in applied mathematics. I would also like to thank my fellow students who has provided me with countless breaks from the long hours spent at the university. Finally I would like to thank my family for making my education possible and a heartfelt thank to my girlfriend Lisa who has been there all the way.

Abstract

This paper studies sound waves generated in corrugated pipes by using computational fluid dynamics. The author did not have any prior experience in computational fluid dynamics and therefore a substantial part was dedicated to theory in computational fluid dynamics. On this background, this paper aimed to develop the necessary boundary conditions and solver settings in order to simulate the problem using OpenFOAM. The main result consists of two cases, where the first case studied the flow over a single corrugation and the second case a study of the flow through a corrugated pipe. It was found that the first case was able to predict frequencies matching previous work. The second case was not able to produce the features connected to singing in corrugated pipes. This is largely attributed to how the inlet boundary condition was set.

Contents

1 Introduction

Today the easiest and most practical way of transporting fluids is by using pipes. Pipes come in different sizes and shapes depending on their intended usage. For flexible pipes the most common design is a corrugated pipe, which is both strong and highly flexible. Unfortunately these pipes are prone to "singing". When air with sufficient speed flows through a corrugated pipe a clear acoustic tone is heard. The turbulence produced by the corrugation leads to vortex shedding/shear instabilities, which in turn excites the acoustic natural frequency of the pipe. This results in a standing wave. For small pipes this may not be a problem as they are normally not used for transporting air/gas, however in sea installations (oil and gas production) the use of corrugated pipes is very common in flexible risers. When gas is transported in these risers they produce sound waves. These sound waves are a problem that oil companies want to reduce as they are a nuisance on the platforms.

In order to solve this problem it is important to have a good understanding of the mechanics behind this event and being able to predict the singing. This is where computational fluid dynamics (CFD) comes in as a quick and cost effective way of studying the capability a corrugated pipe has of singing. This paper will use the open source computational fluid dynamics code *OpenFOAM* to study the phenomenon of sound generated by corrugated pipes.

The singing corrugated pipe phenomenon has been studied for some time. Most studies have been experimental in nature [17, 4, 19, 8] and some have utilized computational fluid dynamics [15]. There is an agreement on the source of the singing which is found to be vortex shedding caused by the corrugations and in turn excites a standing wave inside the pipe [9]. The standing wave is similar to that of a straight pipe which is open in both ends [4]. The frequency of the vortex shedding and the velocity is connected by the dimensionless Strouhal number $St = f \cdot L/u$. The Strouhal number should be constant for any velocity and is found to lie between 0.4-0.6 [8, 19]. The onset of turbulence is important for the generation of the vortices occurring inside the corrugations, however, there has been debate as to which Reynolds number is sufficient in order to obtain a turbulent flow. Crawford [4] found that the classical limit of $Re \approx 2000$ was sufficient, this has been protested by Cadwell [3], which found that the necessary Reynolds number for turbulence could be lower and suggested a different choice of characteristic length scale for calculating the Reynolds number. It is logical to assume that the shape of the corrugation will impact the pipes ability to produce sound. From experiments it was found that rounding the edges of the corrugation up stream could increase the amplitude of the sound produced by a magnitude of 3-4 [12, 19]. This paper will mainly study the behaviour of flow over corrugations and will use a simple model of the corrugation.

2 Fluid Dynamics

2.1 The Continuum Hypothesis

At the base of fluid mechanics, or continuum mechanics, lies the continuum hypothesis. The motivation comes from daily experiences with fluids. We observe fluids as a medium filling a container continuously, which is what the continuum hypothesis describes. The continuum hypothesis of continuum mechanics simply asserts that the macroscopic variables of the medium are distributed continuously throughout the medium [5]. In reality this is not true as molecules are discontinuous, or discrete. However, we assume that the continuum hypothesis is valid as long as the Knudsen number Kn is large enough. The Knudsen number is the ratio of the mean free path of a molecule (average distance between collisions) and the length scale of the problem we wish to study. So if $Kn = \lambda/L << 1$ we assume that the continuum hypothesis is valid.

2.2 Flow Field Description

In fluid mechanics we can describe the flow field in two different ways. There is Lagrangian and Eulerian description. Lagrangian describes the path of an individual particle starting at time t_0 at the point $\mathbf{a} = \mathbf{a}(a_1, a_2, a_3)$. As time progresses the particle traces a curve in $\mathbb{R}^3$ which can be described as

$$\mathbf{x} = \mathbf{r}(\mathbf{a}, t). \tag{2.1}$$

The velocity of the particle is then given by

$$\mathbf{v}(\mathbf{a}, t) = \lim_{\Delta t \to 0} \frac{\Delta \mathbf{r}}{\Delta t} = \frac{\partial \mathbf{r}}{\partial t}(\mathbf{a}, t). \tag{2.2}$$

The Eulerian description describes a flow field $f = f(\mathbf{x}, t)$ in a geometric volume at a position $\mathbf{x}$. In contrast to Lagrangian description where you follow a particle, in Eulerian description we look at a geometric volume in space where particles flow through. We write the velocity as

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{v}(\mathbf{x}(t), t). \tag{2.3}$$

For fluids undertaking large deformations, such as turbulence, Eulerian description is more useful [5].

2.3 Material Derivative

Since Eulerian is the description of choice the following will derive the material derivative in Eulerian coordinates. By definition acceleration is the second derivative of the position vector, or the first derivative of the velocity. By differentiating (2.3) with respect to time the acceleration for a particle is obtained. Care must be taken as $\mathbf{x}$ is a function of time, so by the multivariate chain rule the acceleration is given by

$$\mathbf{a}(\mathbf{x}, t) = \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} \tag{2.4}$$

$$= \frac{\partial \mathbf{v}}{\partial t}\frac{\partial t}{\partial t} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial t} \tag{2.5}$$

$$= \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}. \tag{2.6}$$

The term $\nabla \mathbf{v}$ is the covariant derivative of a vector or more generally the tensor derivative. This also holds for scalar fields where the tensor derivative becomes the gradient. One defines the general material derivative with capital letters

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla \tag{2.7}$$

The material derivative consists of two parts, namely the local acceleration and the convective acceleration. Let ϕ be a generic variable, then the local acceleration is the rate of change of ϕ at a fixed point in space, where as the convective acceleration is the change in ϕ caused by the convective motion from one point in space to another where the value of ϕ is different.

2.4 Mass Conservation

Motivated by the principle of mass conservation, the change of mass inside a domain Ω must be equal to the rate of mass flowing into the domain through the boundary. The total amount of mass inside Ω at any given time t is

$$\int_\Omega \mathrm{d}m = \int_\Omega \frac{\mathrm{d}m}{\mathrm{d}V}\mathrm{d}V = \int_\Omega \rho \mathrm{d}\mathbf{x}. \tag{2.8}$$

By differentiating with respect to time, the change of mass per time inside the domain is

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \rho \mathrm{d}\mathbf{x} = \int_\Omega \frac{\partial \rho}{\partial t}\mathrm{d}\mathbf{x}. \tag{2.9}$$

To calculate the mass flow into the domain, consider a small surface element $\mathrm{d}\sigma$ on $\partial\Omega$ shown in figure 2.1. After a small time interval $\mathrm{d}t$ this surface element has moved a distance $\mathbf{v}\mathrm{d}t$ and traced out a cylinder shown in the figure 2.1. Assuming that $d\sigma$ and $\mathrm{d}t$ are sufficiently small such that all particles in the cylinder has equal and constant velocity as well as equal density, then all particles that pass through $\mathrm{d}\sigma$ has to be inside the cylinder.

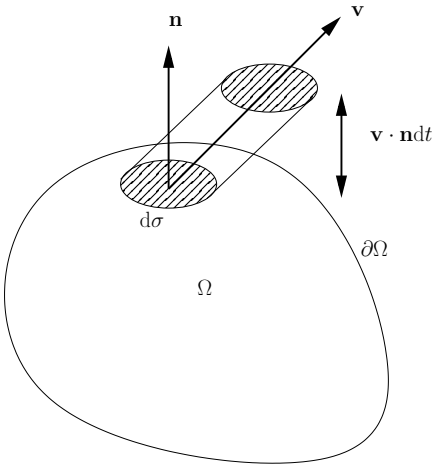


Figure 2.1: Flow through surface $\mathrm{d}\sigma$.

The volume of this cylinder is given by

$$\mathrm{d}V = \mathbf{v}\cdot\mathbf{n}\,\mathrm{d}t\,\mathrm{d}\sigma. \tag{2.10}$$

Which in turn gives the mass flow through $\mathrm{d}\sigma$

$$\mathrm{d}m = \rho \mathrm{d}V = \rho\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}t\,\mathrm{d}\sigma. \tag{2.11}$$

Here $\mathbf{n}$ is the outward unit normal to the surface. The expression $\rho\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}\sigma$ is the mass through $\mathrm{d}\sigma$ per time, which when integrating over the whole surface gives the rate of mass flow through $\partial\Omega$

$$-\int_{\partial\Omega} \rho\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}\sigma. \tag{2.12}$$

The minus sign in front of the integral is to define it so that mass flowing into the domain increase the mass. By equating the change of mass per time inside the domain to the mass flow into the domain an equation for mass conservation is obtained:

$$\int_\Omega \frac{\partial \rho}{\partial t} \mathrm{d}\mathbf{x} = -\int_{\partial\Omega} \rho \mathbf{v} \cdot \mathbf{n} \, \mathrm{d}\sigma. \tag{2.13}$$

Going from a surface integral on the right hand side to a volume integral by Gauss's theorem we obtain the continuity equation in global form

$$\int_\Omega \left(\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} \right) = 0. \tag{2.14}$$

Since this is valid for any arbitrary domain we can write the continuity equation on local form

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = 0. \tag{2.15}$$

2.5 Momentum Conservation

An equation of motion can be derived through the use of Newton's law of motion applied to an infinitesimal fluid element. Consider a small fluid element, then Newton's law requires that the sum of forces applied to the element is equal to the mass times acceleration of the fluid element. The forces on the fluid element is a result of the stress acting on the fluid element. The stress will be denoted as a tensor called the stress tensor τ.

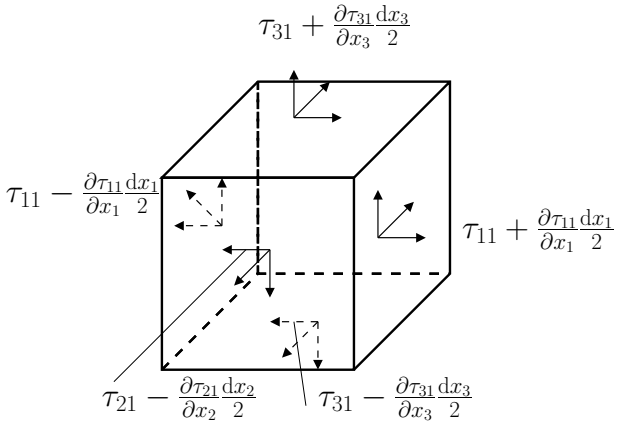


Figure 2.2: Stresses on a fluid element. For clarity, not all sides are shown.

From figure 2.2 it is apparent that the forces in the x-direction is given by

$$\begin{aligned}
&\left(\tau_{11} + \frac{\partial \tau_{11}}{\partial x_1}\frac{\delta x_1}{2} - \tau_{11} + \frac{\partial \tau_{11}}{\partial x_1}\frac{\delta x_1}{2} \right) \delta x_2 \delta x_3 \\
+ &\left(\tau_{21} + \frac{\partial \tau_{21}}{\partial x_2}\frac{\delta x_2}{2} - \tau_{21} + \frac{\partial \tau_{21}}{\partial x_2}\frac{\delta x_2}{2} \right) \delta x_1 \delta x_3 \\
+ &\left(\tau_{31} + \frac{\partial \tau_{31}}{\partial x_3}\frac{\delta x_3}{2} - \tau_{31} + \frac{\partial \tau_{31}}{\partial x_3}\frac{\delta x_3}{2} \right) \delta x_1 \delta x_2 \\
= &\left(\frac{\partial \tau_{11}}{\partial x_1} + \frac{\partial \tau_{21}}{\partial x_2} + \frac{\partial \tau_{31}}{\partial x_3} \right) \delta x_1 \delta x_2 \delta x_3.
\end{aligned}$$

The other directions are derived in the same manner giving the general result

$$\frac{\partial \tau_{ij}}{\partial x_j} \delta x_1 \delta x_2 \delta x_3. \tag{2.16}$$

Then by using Newton's law we get

$$\rho\delta x_1\delta x_2\delta x_3\frac{Du_i}{Dt} = \rho g_i\delta x_1\delta x_2\delta x_3 + \frac{\partial\tau_{ij}}{\partial x_j}\delta x_1\delta x_2\delta x_3, \tag{2.17}$$

where $\rho g_i\delta x_1\delta x_2\delta x_3$ is the body force and $\rho\delta x_1\delta x_2\delta x_3$ is the mass of the fluid. Dividing by the volume gives what is usually called Cauchy's equation of motion

$$\rho\frac{Du_i}{Dt} = \rho g_i + \frac{\partial\tau_{ij}}{\partial x_j}. \tag{2.18}$$

This equation governs the motion of the fluid.

2.5.1 Constitutive Equation

In order to obtain the complete Navier-Stokes equations we need to relate the stress and deformation in a fluid. This relation is called a constitutive equation. We will look at a linear relation between the stress and deformation for a newtonian fluid. The stress in a fluid can be split up into two main components, one stress component for when the fluid is at rest and one for the stress developed due to the motion of the fluid. For a fluid at rest the stress is composed of only normal components on a surface and does not depend on the orientation of the surface. Therefore the stress tensor would be isotropic and the only second order tensor which is isotropic is the Kronecker delta.

$$\delta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.19}$$

For a fluid at rest the only normal stress component would be the thermodynamic pressure such that $\tau_{ij} = -p\delta_{ij}$. For a fluid in motion the stress would be influenced by the viscosity which would cause the stress tensor to lose its isotropicy. Therefore we introduce a nonisotropic component σ_{ij} called the deviatoric stress tensor. The stress tensor can now be written as

$$\tau_{ij} = -p\delta_{ij} + \sigma_{ij}. \tag{2.20}$$

The negative sign is introduced as the normal components of the stress tensor are regarded as tension when positive and compression when negative. For a general newtonian fluid it can be shown [10] that the deviatoric stress tensor becomes

$$\sigma_{ij} = 2\mu e_{ij} + \lambda e_{mm}\delta_{ij}, \tag{2.21}$$

where

$$e_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{2.22}$$

and $e_{mm} = \nabla\cdot\mathbf{u}$ is the volumetric strain rate and μ and λ are scalars. The complete stress tensor is now

$$\tau_{ij} = -p\delta_{ij} + 2\mu e_{ij} + \lambda e_{mm}\delta_{ij}. \tag{2.23}$$

When the fluid is incompressible the stress tensor reduces to $\tau_{ij} = -p\delta_{ij} + 2\mu e_{ij}$ by the continuity equation $\nabla\cdot\mathbf{u} = 0$. There is a relationship between the scalars μ and λ which is not fully understood, so it is common to use Stokes assumption which has been found to be sufficiently accurate for most applications

$$\lambda + \frac{2}{3}\mu = 0, \tag{2.24}$$

$$\lambda = -\frac{2}{3}\mu. \tag{2.25}$$

Using Stokes assumption the complete stress tensor (2.23) can be written as

$$\tau_{ij} = -p\delta_{ij} + 2\mu e_{ij} - \frac{2}{3}\mu(\nabla \cdot \mathbf{u})\delta_{ij}. \tag{2.26}$$

With the term for the stress tensor (2.26) substituted into (2.18) we get a general form of the Navier-Stokes equation

$$\rho\frac{Du_i}{Dt} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\mu e_{ij} - \frac{2}{3}\mu(\nabla \cdot \mathbf{u})\delta_{ij}\right) + \rho g_i. \tag{2.27}$$

2.6 Energy Conservation

From the first law of thermodynamics we can derive an equation for the conservation of energy. The first law of thermodynamic states that

Rate of increase in energy	=	Net rate of heat added	+	Net rate of work done

In order to develop the energy equation we will find expressions for each of the terms in the first law of thermodynamics. We define the total energy $E = i + \frac{1}{2}(u_1^2 + u_2^2 + u_3^2)$, where i is the internal energy and $\frac{1}{2}(u_1^2 + u_2^2 + u_3^2)$ is the kinetic energy. This energy is often called the stored energy per unit mass.

Rate of increase in energy: Let E be the total amount of energy in a control volume. Then the rate of change in energy in the control volume will be expressed using the material derivative,

$$\rho\frac{DE}{Dt}\delta x_1 \delta x_2 \delta x_3. \tag{2.28}$$

Net rate of heat added: This is energy entering the control volume by heat conduction. Let $\mathbf{q} = (q_1, q_2, q_3)$ be the heat flux vector. By doing a similar analysis as in section 2.5 we can obtain an expression for the rate of heat added to the control volume. The following figure shows, for clarity, two of the components of the heat flux vector.

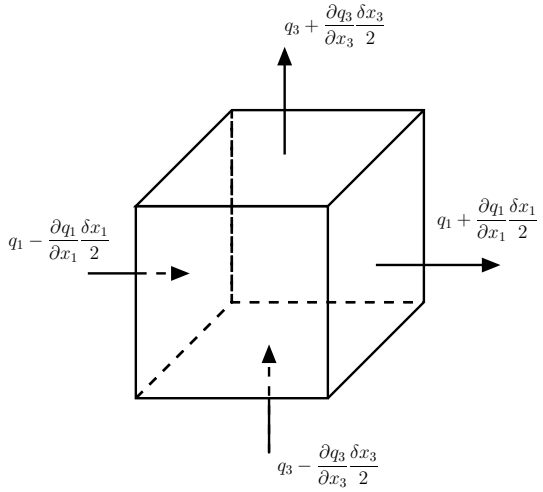


Figure 2.3: The heat flux vector for x_1 and x_3 direction.

We will consider only the x-direction as the other directions follows directly from the x-direction. Therefore by looking at the heat entering the control volume in the x-direction we

get

$$\left[\left(q_1 - \frac{\partial q_1}{\partial x_1}\frac{\delta x_1}{2}\right) - \left(q_1 + \frac{\partial q_1}{\partial x_1}\frac{\delta x_1}{2}\right)\right]\delta x_2\delta x_3 = -\frac{\partial q_1}{\partial x_1}\delta x_1\delta x_2\delta x_3. \tag{2.29}$$

Similarly we get the y- and z-direction. When combined we see that this is the divergence of the heat flux vector $\mathbf{q}$

$$-\frac{\partial q_1}{\partial x_1}\delta x_1\delta x_2\delta x_3 - \frac{\partial q_2}{\partial x_2}\delta x_1\delta x_2\delta x_3 - \frac{\partial q_3}{\partial x_3}\delta x_1\delta x_2\delta x_3 = -\mathrm{div}(\mathbf{q})\delta x_1\delta x_2\delta x_3. \tag{2.30}$$

We can obtain an expression for the heat flux vector by using Fourier's law of heat conduction which relates the heat flux vector to the local temperature gradient.

$$\mathbf{q} = -k\,\mathrm{grad}(T), \tag{2.31}$$

where k is the thermal conductivity of the fluid. Throughout this paper the mathematical operators div and grad will be used interchangeably in order to increase readability. When combining the equations (2.30) and (2.31) we obtain an equation expressed by the temperature for the net rate of heat added to the control volume

$$-\mathrm{div}(\mathbf{q})\delta x_1\delta x_2\delta x_3 = \mathrm{div}(k\,\mathrm{grad}(T))\delta x_1\delta x_2\delta x_3. \tag{2.32}$$

Net rate of work done: The work rate done on the fluid element is equal to the velocity component in the direction of the force times the force. From section 2.5 we know the force acting upon the fluid element is

$$\frac{\partial \tau_{ij}}{\partial x_j}\delta x_1\delta x_2\delta x_3, \qquad \text{surface force} \tag{2.33}$$

$$\rho g_i \delta x_1\delta x_2\delta x_3. \qquad \text{body force} \tag{2.34}$$

Multiplying this by the velocity gives us

$$\frac{\partial (u_i\tau_{ij})}{\partial x_j}\delta x_1\delta x_2\delta x_3, \qquad \text{surface force} \tag{2.35}$$

$$\rho u_i g_i \delta x_1\delta x_2\delta x_3. \qquad \text{body force} \tag{2.36}$$

The energy equation: Collecting all the terms and dividing by the volume we obtain the conservation equation for energy

$$\rho\frac{DE}{Dt} = \rho u_i g_i - \mathrm{div}(\mathbf{q}) + \frac{\partial (u_i\tau_{ij})}{\partial x_j}, \tag{2.37}$$

or expressed with the temperature

$$\rho\frac{DE}{Dt} = \rho u_i g_i + \mathrm{div}(k\,\mathrm{grad}(T)) + \frac{\partial (u_i\tau_{ij})}{\partial x_j}. \tag{2.38}$$

It is common to use a different form of the energy equation, an equation for the internal energy. By removing the kinetic energy from the energy equation one is left with an equation for the internal energy. The kinetic energy can be obtained by multiplying the momentum conservation equation (2.18) with the velocity. This yields

$$\begin{aligned} \rho\frac{Du_i}{Dt}u_i &= \rho u_i g_i + u_i\frac{\partial \tau_{ij}}{\partial x_j}, \\ \rho\frac{D}{Dt}\left(\frac{1}{2}u_i u_i\right) &= \rho u_i g_i + u_i\frac{\partial \tau_{ij}}{\partial x_j}. \end{aligned} \tag{2.39}$$

It can be shown [10] that the kinetic energy equation can, when using the constitutive equation (2.26), be written as

$$\rho \frac{D}{Dt}\left(\frac{1}{2}u_i u_i\right) = \rho u_i g_i + \frac{\partial(u_i \tau_{ij})}{\partial x_j} + p(\nabla \cdot \mathbf{u}) - \phi, \tag{2.40}$$

where ϕ is the rate of viscous dissipation

$$\phi = 2\mu\left(e_{ij} - \frac{1}{3}(\nabla \cdot \mathbf{u})\delta_{ij})\right)^2. \tag{2.41}$$

Equation (2.40) is the kinetic energy. Subtracting this from the energy equation (2.37) gives us the equation for internal energy

$$\rho \frac{Di}{Dt} = -\nabla \cdot \mathbf{q} - p(\nabla \cdot \mathbf{u}) + \phi. \tag{2.42}$$

This equation is also called the heat equation.

2.7 Equations of State

The equation of state is an equation which relates state variables and is based on thermodynamic equilibrium. In thermodynamics we have the two equations

$$\begin{aligned} p &= p(\rho, T), \\ i &= i(\rho, T). \end{aligned} \tag{2.43}$$

For a perfect gas the equations of state are $p = \rho RT$ and $i = C_v T$. When the flow is compressible this equations provide the link between pressure and temperature which determines the density of the fluid.

2.8 Transport Equation

A summary of all the governing equations in fluid dynamics can be found in the following table

Continuity equation	$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho u_i = 0$
Momentum equation	$\frac{D(\rho u_i)}{Dt} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\mu e_{ij} - \frac{2}{3}\mu(\nabla \cdot \mathbf{u})\delta_{ij}\right) + S_i$
Energy equation	$\frac{D(\rho i)}{Dt} = -\nabla \cdot \mathbf{q} - p(\nabla \cdot \mathbf{u}) + \phi$
Equations of state	$p = p(\rho, T)$ and $i = i(\rho, T)$

Table 2.1: Governing equations for a compressible Newtonian fluid

In the next section we will develop numerical schemes for solving the above conservation equations, however working with several different equations would become cumbersome. Which is why we will reduce this to a single equation. If we observe that all the conservation equations have similar structure, we can create a single equation in which all the other equations can be expressed. Usually this equation is called the transport equation.

$$\frac{\partial(\rho\Phi)}{\partial t} + \operatorname{div}(\rho \mathbf{u}\Phi) = \operatorname{div}(\Gamma \operatorname{grad}(\Phi)) + S_\Phi. \tag{2.44}$$

Here Φ represents any flow variable and Γ is the diffusion coefficient. Setting Φ to different values $(1, u, v, w, i)$ and appropriate values for S_Φ and Γ we obtain the original equations. In this form a physical meaning of the terms in equation (2.44) can with little effort be deduced.

$$\begin{array}{c}\text{Rate of} \\ \text{increase of } \Phi\end{array} + \begin{array}{c}\text{Net rate of flow} \\ \text{of } \Phi \text{ out of fluid}\end{array} = \begin{array}{c}\text{Rate of increase} \\ \text{of } \Phi \text{ due to} \\ \text{diffusion}\end{array} + \begin{array}{c}\text{Rate of increase of} \\ \Phi \text{ due to sources}\end{array}$$

3 Computational Fluid Dynamics

Solving the Navier-Stokes equation analytically is in general extremely difficult or even impossible. Only a select set of simple problems can be solved analytically. More complicated and useful problems requires numerical solutions. However simple problems with analytical solutions are of high value for learning fluid dynamics. This section will explore some possible mathematical models being used in computational fluid dynamics so a reasonable model can be chosen for this specific problem. It is also imperative to have a thorough understanding of computational fluid dynamics in order to correctly use and interpret results from existing CFD codes, thus making this section of great importance.

3.1 Discretization

There are three main discretisation methods used for solving partial differential equations, namely the finite difference method, the finite volume method and the finite element method. The two most important properties of any discretization method is the available order of the discretisation schemes used and the ability to handle geometries. The following sections will compare these methods with a more in-depth look at the finite volume method, as it is the discretisation of choice in CFD.

3.1.1 Finite Difference Method

The oldest method of solving partial differential equations numerically was believed to have been developed by Euler. The main idea to this method is that one can estimate the derivative at each nodal point in a grid by a Taylor series expansion of the partial derivatives. This makes it easy to obtain higher order differencing approximations on a uniform grid, which in turn yields better accuracy of the solution. By the means of Taylor series the values at non-nodal positions can be evaluated. The finite difference method can be applied to any grid shape, however it is most commonly used in uniform grids as it requires a high degree of regularity to maintain accuracy. This limitation is not very well suited for computational fluid dynamics where complex grids are very common.

3.1.2 Finite Volume Method

Introduced in the early 70's the finite volume method uses a discretization of the integral form of the conservation equations. The computational domain is decomposed into a finite number of contiguous control volumes on which the conservation equations are solved on each control volume. The control volumes are often made with triangles (in 2D) and tetrahedrals (in 3D) making it able to handle complex geometries. This is in stark contrast of the finite difference method. However, approximations of higher order than two are difficult to develop in three dimensions since the finite volume method requires three levels of approximation; interpolation, differentiation and integration.

3.1.3 Finite Element Method

Similarly to the finite volume method, the finite element method also divides the domain into small volumes, or finite elements. Making it an excellent method for solving problems on unstructured meshes for complex geometries. This has been the main method for solving partial differential equations for quite some time. It has seen its use in structural analysis, electromagnetics and fluid dynamics. Compared to the finite volume method, the implementation for complex geometries is straight forward, which is why FEM is widely used in structural analysis. The down side, however, is that the FEM requires more computational power than the finite volume method [20], which is why it is not widely used for computational fluid dynamics.

3.2 Finite Volume Method

The finite volume method is composed in three stages.

1. Mesh generation
2. Discretization
3. Solve system of equations

In this section the second stage will be examined. The first and last step is also very important, more so the mesh generation. Generating a good mesh is an art in itself and will not be discussed in detail here. We begin with the transport equation given in equation (2.44):

$$\frac{\partial(\rho\phi)}{\partial t} + \operatorname{div}(\rho\phi\mathbf{u}) = \operatorname{div}(\Gamma \operatorname{grad}\phi) + S_\phi. \tag{3.1}$$

This chapter will start by looking at a pure steady state diffusion problem such that a discretisation of the diffusion term in the transport equation can easily be obtained. Following that, a discretisation approach to a steady convection-diffusion problem will be presented which will be the basis for the differencing schemes discussed here. The unsteady convection-diffusion problem can be discussed independently of the steady state problems and will be presented last. Before the discretisation process can take place a suitable mesh is needed. For clarity a one dimensional mesh will be used, extending to other dimensions is straight forward. On this one dimensional mesh we need to place a number of nodes with boundaries to make up small control volumes. We will follow standard CFD convention and use 'P' for a general node, 'W' for west, 'E' for east and so on. Upper case letters signifies nodal points and lower case represents the boundary (or faces) of a nodal point.

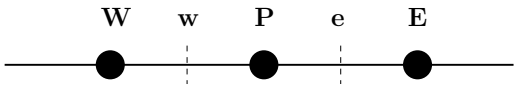


Figure 3.1: One dimensional grid.

Now we are ready to discretise the problem. As stated above, we will first discretise a steady diffusion problem. By removing the transient and convective terms in the general transport equation (2.44) we get the steady diffusion equation

$$\operatorname{div}(\Gamma \operatorname{grad}\phi) + S_\phi = 0. \tag{3.2}$$

Following next is what separates the finite volume method from the finite element method, namely the control volume integration. The steady diffusion equation integrated over the control volume is

$$\int_{\Delta V} \frac{\mathrm{d}}{\mathrm{d}x}\left(\Gamma\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)\mathrm{d}V + \int_{\Delta V} S_\Phi \mathrm{d}V = 0. \tag{3.3}$$

By applying Gauss's divergence theorem to the first volume integral and approximating the source term by the midpoint rule we obtain

$$\int_A \mathbf{n}\cdot\left(\Gamma\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)\mathrm{d}A + \int_{\Delta V} S_\Phi \mathrm{d}V \approx \left(\Gamma A\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_e - \left(\Gamma A\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_w + S_\Phi \Delta V = 0, \tag{3.4}$$

where we have assumed that the source term is constant over the volume. In order to express (3.4) in nodal points (P, W and E) we need to approximate the gradients. Assuming a linear gradient between nodal points we get the following

$$\left(\Gamma A\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_e \approx \Gamma_e A_e\left(\frac{\Phi_E - \Phi_P}{\delta x_{PE}}\right), \tag{3.5}$$

$$\left(\Gamma A\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_w \approx \Gamma_w A_w\left(\frac{\Phi_P - \Phi_W}{\delta x_{WP}}\right). \tag{3.6}$$

By substituting (3.5) and (3.6) into (3.4) and rearranging terms to collect Φ_P, Φ_E and Φ_W we obtain the fully discretised equation for our problem.

$$\Gamma_e A_e \left(\frac{\Phi_E - \Phi_p}{\delta x_{PE}} \right) - \Gamma_w A_w \left(\frac{\Phi_P - \Phi_W}{\delta x_{WP}} \right) + \Delta V S_\Phi = 0 \tag{3.7}$$

$$\left(\frac{\Gamma_e A_e}{\delta x_{PE}} + \frac{\Gamma_w A_w}{\delta x_{WP}} \right) \Phi_P = \left(\frac{\Gamma_w A_w}{\delta x_{WP}} \right) \Phi_W + \left(\frac{\Gamma_e A_e}{\delta x_{PE}} \right) \Phi_E + \Delta V S_\phi. \tag{3.8}$$

The coefficients of Φ_P, Φ_E and Φ_W are conveniently renamed to a_P, a_E and a_W respectively, such that equation (3.8) can be written as

$$a_P \Phi_p = a_E \Phi_E + a_W \Phi_W + \Delta V S_\Phi, \tag{3.9}$$

where

$$a_E = \frac{\Gamma_e A_e}{\delta x_{PE}}, \quad a_W = \frac{\Gamma_w A_w}{\delta x_{WP}}, \quad a_P = a_E + a_W.$$

The equation (3.9) is the complete discretised form for the steady diffusion problem.

Steady state convection-diffusion equation The steady state transport equation without the source term is the equation governing convection and diffusion of a property Φ. Again consider the one dimensional case,

$$\frac{\mathrm{d}}{\mathrm{d}x}(\rho u \Phi) = \frac{\mathrm{d}}{\mathrm{d}x} \left(\Gamma \frac{\mathrm{d}\Phi}{\mathrm{d}x} \right). \tag{3.10}$$

The continuity equation must also be satisfied

$$\frac{\mathrm{d}(\rho u)}{\mathrm{d}x} = 0. \tag{3.11}$$

Integrating over the control volume as before we obtain

$$(\rho u A \Phi)_e - (\rho u A \Phi)_w = \left(\Gamma A \frac{\mathrm{d}\Phi}{\mathrm{d}x} \right)_e - \left(\Gamma A \frac{\mathrm{d}\Phi}{\mathrm{d}x} \right)_w. \tag{3.12}$$

Integrating the continuity equation over the control volume yields

$$(\rho u A)_e - (\rho u A)_w = 0. \tag{3.13}$$

In order to discretise these equations we have to approximate the transported property Φ on the left hand side of equation (3.12), the right hand side was already approximated using central difference in the previous section. We introduce now a notation such that working with different differencing schemes is less cluttered. Let

$$F = \rho u \quad \text{and} \quad D = \frac{\Gamma}{\delta x}, \tag{3.14}$$

where F represents the convective mass flux per unit area and D represents diffusion conductance at cell faces. Using this notation, equation (3.12) can be written

$$F_e \Phi_e - F_w \Phi_w = D_e(\Phi_E - \Phi_P) - D_w(\Phi_P - \Phi_W), \tag{3.15}$$

where

$$F_w = (\rho u)_w, \quad D_w = \frac{\Gamma_w}{\delta x_{WP}}, \tag{3.16}$$

$$F_e = (\rho u)_e, \quad D_e = \frac{\Gamma_e}{\delta x_{PE}}. \tag{3.17}$$

The integrated continuity equation using this notation becomes

$$F_e - F_w = 0. \tag{3.18}$$

We have yet to calculate the value of the transported property Φ to the faces w and e. There are a number of schemes which accomplishes this, some are better than others. In the following sections we shall consider common interpolation schemes used in computational fluid dynamics.

3.3 Discretisation Scheme Properties

Before investigating any schemes, it is important to define three properties connected to a discretisation scheme. The more a scheme coheres to the different properties the better it is expected to perform in computations. The three properties are

- Conservativeness
- Transportiveness
- Boundedness

These properties will be assessed when considering the different discretisation schemes.

3.3.1 Conservativeness

When solving these discretised conservation equations it is desirable that the property Φ is to be conserved over the whole domain of computation. To achieve this the flux going out of a control volume face is to be equal the flux going in to the adjacent control volume face. This adds a restriction to the discretisation scheme where the flux through the adjacent faces needs to be consistently represented.

3.3.2 Transportiveness

Consider a nodal point P with adjacent nodes W and E (as in the previous figures) where both W and E contains a source. The influence of these sources on point P can be described using the non-dimensional cell Peclet number. The Peclet number is a measure of the relative strength of convection and diffusion and is defined as

$$Pe = \frac{F}{D} = \frac{\rho u}{\Gamma/\delta x}, \tag{3.19}$$

where F and D is given by (3.14), Γ is the diffusion coefficient and δx is the characteristic length (cell width). When the computation is pure diffusion ($Pe \to 0$) the point P is influenced equally by the two sources at W and E. However, when the fluid is undergoing pure convection ($Pe \to \infty$) and the fluid is flowing from W to E, the point P will get a larger contribution from W than E. To satisfy transportiveness a scheme needs to account for fluid direction.

3.3.3 Boundedness

Boundedness states that a property Φ, in the absence of a source term, is bounded by its boundary values. As an example, consider a steady conduction problem with no sources. Let the bondary take the values of 100 kelvin and 200 kelvin. Boundedness requires that the property T (temperature) inside the domain is less than 200 kelvin and greater than 100 kelvin. Combined with the requirement that all the coefficients of the discretised equations should have the same sign, forms the boundedness property.

A diagonally dominant coefficient matrix is favourable in the task of satisfying boundedness. Scarborough (1958) presented a sufficient condition for a iterative method to be convergent:

$$\frac{\sum |a_n|}{|a'_P|} \begin{cases} \leq 1 \text{ at all nodes} \\ < 1 \text{ at least at one node} \end{cases} \tag{3.20}$$

Here a_P is the central node and a_n is all the neighbouring nodes. When this condition is satisfied the coefficient matrix is diagonally dominant.

3.4 Central Differencing Scheme

As mentioned we need to calculate the values Φ_e and Φ_w for the convection diffusion problem. The central difference scheme for Φ_e and Φ_w when using a uniform grid is

$$\Phi_e = \frac{\Phi_P + \Phi_E}{2}, \tag{3.21}$$

$$\Phi_w = \frac{\Phi_W + \Phi_P}{2}. \tag{3.22}$$

Substituting this into equation (3.15) yields

$$F_e \frac{\Phi_P + \Phi_E}{2} - F_w \frac{\Phi_W + \Phi_P}{2} = D_e(\Phi_E - \Phi_P) - D_w(\Phi_P - \Phi_W). \tag{3.23}$$

When rearranged this becomes

$$\left[\left(D_w + \frac{F_w}{2}\right) + \left(D_e - \frac{F_e}{2}\right) + (F_e - F_w)\right]\Phi_P \tag{3.24}$$

$$= \left(D_w + \frac{F_w}{2}\right)\Phi_W + \left(D_e - \frac{F_e}{2}\right)\Phi_E. \tag{3.25}$$

As before we identify the coefficients of Φ_P, Φ_W and Φ_E as a_P, a_W and a_E respectively. Then the discretised convection diffusion can be, using central differencing, written as

$$a_P\Phi_P = a_W\Phi_W + a_E\Phi_E, \tag{3.26}$$

where

$$a_P = a_W + a_E + (F_e - F_w), \quad a_W = D_w + \frac{F_w}{2}, \quad a_E = D_e - \frac{F_e}{2}. \tag{3.27}$$

Next we will discuss the properties of the central differencing scheme.

3.4.1 Scheme Properties

Conservativeness: The flux at a control volume face is determined by only one function and is therefore consistent and conservative.

Transportiveness: There central differencing scheme does not recognise the direction of flow and can therefore not generally possess the transportive property. The central differencing scheme can be used for small enough Peclet number. Depending of course on the problem needed to be solved.

Boundedness: From the coefficients (3.27) we see that when applying the continuity equation (3.18) we get $a_P = a_W + a_E$. We see then that the coefficients satisfy the Scarborough (3.20) condition. A second condition required for boundedness is that the sign for all the coefficients should be the same. Let both F_w and F_e be positive. Then a_w is always positive, but a_e can be negative for a large enough value of F_e. For this coefficient to be positive the Peclet number has to be constrained by

$$\frac{F_e}{D_e} = Pe < 2. \tag{3.28}$$

A larger Peclet number will result in a negative coefficient.

Numerical accuracy: The numerical accuracy can be determined by a Taylor expansion and is

of second order accuracy. A Taylor expansion around Φ_e gives us

$$\Phi_E = \Phi_e + \left(\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_e \frac{\Delta x}{2} + \mathcal{O}((\Delta x/2)^2) \tag{3.29}$$

$$\Phi_P = \Phi_e + \left(\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_e \frac{\Delta x}{2} + \mathcal{O}((\Delta x/2)^2 \tag{3.30}$$

$$\Phi_E + \Phi_P = 2\Phi_e + 2\mathcal{O}((\Delta x/2)^2) \tag{3.31}$$

$$\frac{\Phi_E + \Phi_P}{2} = \Phi_e. \tag{3.32}$$

Truncating the second order term and higher makes this interpolation of second order.

General behaviour: It is well known that the central differencing scheme can exhibit large over and under shoots, which eventually leads to a divergent solution. To reduce these over and undershoots, an increase in mesh resolution could overcome the problem, but for any practical application this solution falls short due to the high resolution needed.

3.5 Upwind Scheme

The central differencing scheme weights all neighbouring nodes equally when evaluating the control volume faces. This effectively eliminates the transportiveness property. In general the node up stream should have a stronger influence than the node down stream in a strong convective flow, this is the motivation for developing a scheme which considers the flow direction. The upwind scheme accomplishes this by setting the control volume face to the value at the up stream node. If the flow goes from west to east ($F_w > 0, F_e > 0$), we would then set

$$\Phi_w = \Phi_W, \tag{3.33}$$

$$\Phi_e = \Phi_P. \tag{3.34}$$

Similarly, if the flow is reversed we set

$$\Phi_w = \Phi_P, \tag{3.35}$$

$$\Phi_e = \Phi_E. \tag{3.36}$$

For west to east flow the equation (3.8) becomes

$$F_e\Phi_P - F_w\Phi_W = D_e(\Phi_E - \Phi_P) - D_w(\Phi_P - \Phi_W). \tag{3.37}$$

Collecting coefficients for Φ_P, Φ_W and Φ_E gives

$$(D_w + D_e + F_e)\Phi_P = (D_w + F_w)\Phi_W + D_e\Phi_E. \tag{3.38}$$

Rewriting the coefficient for Φ_P in terms of the coefficients of Φ_W and Φ_E gives

$$[(D_w + F_w) + D_e + (F_e - F_w)]\Phi_P = (D_w + F_w)\Phi_W + D_e\Phi_E. \tag{3.39}$$

Substituting the coefficients in this equation with a_P, a_W and a_E, as in equation (3.9), we obtain the same general form of the discretisation

$$a_P\Phi_P = a_W\Phi_W + a_E\Phi_E, \tag{3.40}$$

with central coefficient $a_P = a_W + a_E + (F_e - F_w)$, where a_W and a_E are dependent on flow direction. For flow with ($F_w > 0, F_e > 0$) they become

$$a_W = D_w + F_w, \tag{3.41}$$

$$a_E = D_e. \tag{3.42}$$

For flow with ($F_w < 0, F_e < 0$) they become

$$a_W = D_w, \tag{3.43}$$

$$a_E = D_e - F_e. \tag{3.44}$$

3.5.1 Scheme Properties

Conservativeness: The upwind scheme is consistent in calculating the flux through control volume faces and does therefore inherit the conservative property.

Transportiveness: The upwind scheme is based on the transport property and does therefore satisfy the condition for transportiveness.

Boundedness: From the coefficient a_W and a_E it is apparent that they are always positive which is a requirement for boundedness. The Scarborough condition (3.20) is satisfied when the continuity equation is applied to the central coefficient, which is a sufficient condition for a convergent iterative method.

Numerical accuracy: Taylor expansion shows that the upwind scheme (backward differencing) has first order accuracy.

$$\Phi_w = \Phi_W + \left(\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_W \frac{\Delta x}{2}. \tag{3.45}$$

$$\Phi_w = \Phi_W + \mathcal{O}(\Delta x/2) \tag{3.46}$$

General behaviour: The upwind scheme is known for its numerical stability with no over or under shoots in the solution. It does however introduce unwanted diffusion in space when the flow is not aligned with the grid lines. Which is commonly known as false diffusion. Care has to be taken in high Reynolds flow as the diffusion can be large enough to lead to non-physical results.

3.6 QUICK Scheme

The previously discussed upwind scheme is stable and posses the transportive property, but does however do this at first order accuracy which leads to diffusion in the solution. A higher order scheme is needed. The QUICK (quadratic upstream interpolation for convective kinetics) scheme aims at keeping the transportive property of a upwind scheme but at the same time using a higher order discretisation in order to reduce numerical diffusion errors.

The QUICK scheme utilises two nodal points upstream and one point downstream in order to obtain third order accuracy and transportive property. A quadratic interpolation is used to calculate the face values at the control volume. The direction of the flow dictates the use of nodal points for interpolation, which makes the transportiveness directly implemented in the scheme as with the upwind scheme. Consider a flow going from west to east; $u_w > 0$ and $u_e > 0$. The value at Φ_w is evaluated from a quadratic interpolation through the nodes WW, W and P. The nodes at W, P and E is used to evaluate the value at Φ_e. The faces Φ_w and Φ_e takes the value

$$\Phi_w = \frac{6}{8}\Phi_W + \frac{3}{8}\Phi_P - \frac{1}{8}\Phi_{WW}, \tag{3.47}$$

$$\Phi_e = \frac{6}{8}\Phi_P + \frac{3}{8}\Phi_E - \frac{1}{8}\Phi_W. \tag{3.48}$$

Inserting into equation (3.15) yields

$$\begin{aligned}&\left[F_e\left(\frac{6}{8}\Phi_P + \frac{3}{8}\Phi_E - \frac{1}{8}\Phi_W\right) - F_w\left(\frac{6}{8}\Phi_W + \frac{3}{8}\Phi_P - \frac{1}{8}\Phi_{WW}\right)\right] \\ &= D_e(\Phi_E - \Phi_P) - D_w(\Phi_P - \Phi_W).\end{aligned} \tag{3.49}$$

Collecting the coefficients of Φ_P, Φ_W and Φ_E and then rewriting the coefficient for Φ_P in terms

of the coefficients of Φ_W and Φ_E gives us

$$\begin{aligned}&\left[D_w - \frac{3}{8}F_w + D_e + \frac{6}{8}F_e\right]\Phi_P \\ &= \left[D_w + \frac{6}{8}F_w + \frac{1}{8}F_e\right]\Phi_W + \left[D_e - \frac{3}{8}F_e\right]\Phi_E - \frac{1}{8}F_w\Phi_{WW}.\end{aligned} \tag{3.50}$$

Taking the coefficients of Φ_P, Φ_W and Φ_E to be a_P, a_W and a_E respectively we write equation (3.50) in standard form

$$a_P\Phi_P = a_W\Phi_W + a_E\Phi_E + a_{WW}\Phi_{WW}, \tag{3.51}$$

where

$$a_W = D_w + \frac{6}{8}F_w + \frac{1}{8}F_e \tag{3.52}$$

$$a_E = D_e - \frac{3}{8}F_e \tag{3.53}$$

$$a_{WW} = -\frac{1}{8}F_w \tag{3.54}$$

$$a_P = a_W + a_E + a_{WW} + (F_e - F_w). \tag{3.55}$$

For flow in the opposite direction, $F_w < 0$ and $F_e < 0$ the discretisation becomes

$$a_P\Phi_P = a_W\Phi_W + a_E\Phi_E + a_{EE}\Phi_{EE}, \tag{3.56}$$

where

$$a_W = D_w + \frac{3}{8}F_w \tag{3.57}$$

$$a_E = D_e - \frac{6}{8}F_e - \frac{1}{8}F_e \tag{3.58}$$

$$a_{EE} = \frac{1}{8}F_e \tag{3.59}$$

$$a_P = a_W + a_E + a_{EE} + (F_e - F_w). \tag{3.60}$$

3.6.1 Scheme properties

Conservativeness: The QUICK scheme is conservative as it uses consistent quadratic functions to calculate the control volume face flux.

Transportiveness: As with the upwind scheme, the transportiveness is present in the QUICK scheme by construction.

Boundedness: If the flow satisfies the continuity equation, then $a_P = a_W + a_E$; this satisfies the Scarborough condition (3.20) which is desirable for boundedness. However, the QUICK scheme have stability problems restricted by the Peclet number. For $F_w < 0$ and $F_e < 0$ the coefficient a_E will be negative when the Peclet number is larger than

$$a_E = D_e - \frac{3}{8}F_e < 0 \tag{3.61}$$

$$Pe_e = \frac{F_e}{D_e} > \frac{8}{3}. \tag{3.62}$$

This makes the QUICK scheme conditionally stable.

Numerical accuracy: Taylor expansion shows that the QUICK scheme is of third order accuracy.

Expanding around the east face we obtain

$$\Phi_E = \Phi_e + \left(\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_e \frac{1}{2}\Delta x + \left(\frac{\mathrm{d}^2\Phi}{\mathrm{d}x^2}\right)_e \frac{1}{2!}\left(\frac{1}{2}\Delta x\right)^2 + \mathcal{O}(\Delta x^3) \tag{3.63}$$

$$\Phi_P = \Phi_e - \left(\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_e \frac{1}{2}\Delta x + \left(\frac{\mathrm{d}^2\Phi}{\mathrm{d}x^2}\right)_e \frac{1}{2!}\left(-\frac{1}{2}\Delta x\right)^2 + \mathcal{O}(\Delta x^3) \tag{3.64}$$

$$\Phi_W = \Phi_e - \left(\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_e \frac{3}{2}\Delta x + \left(\frac{\mathrm{d}^2\Phi}{\mathrm{d}x^2}\right)_e \frac{1}{2!}\left(-\frac{3}{2}\Delta x\right)^2 + \mathcal{O}(\Delta x^3) \tag{3.65}$$

Adding these equations with the appropriate coefficients we can cancel out the derivatives and be left with only third order terms.

$$\frac{3}{8}\Phi_E + \frac{6}{8}\Phi_P - \frac{1}{8}\Phi_W = \Phi_e + \mathcal{O}(\Delta x^3). \tag{3.66}$$

General behaviour: The QUICK scheme has third order accuracy and has transportive properties. The solutions are generally low in false diffusion and can produce good results on coarse grids. However, the Peclet number restriction makes this scheme prone to numerical instabilities in complex flows, which can result in overshoots and undershoots.

3.7 TVD Schemes

The previously presented schemes have different advantages and disadvantages, therefore it would be only reasonable to attempt to construct a scheme which has all the good qualities and none of the bad qualities. This has led to the development of the TVD (total variation diminishing) schemes. These schemes are of second order, has low false diffusion, oscillation free solutions and the conservativeness/boundedness/transportiveness properties. The TVD schemes are based on a generalisation of upwind weighted differencing schemes. The upwind differencing (UD) scheme for the face value Φ_e is

$$\Phi_e = \Phi_P. \tag{3.67}$$

This scheme is first order. The linear upwind differencing (LUD) scheme is a second order version of the standard UD scheme and can be expressed as

$$\Phi_e = \Phi_P + \frac{1}{2}(\Phi_P - \Phi_W). \tag{3.68}$$

This is the upwind scheme plus a correction making it second order. Similarly the central differencing (CD) scheme and the Hayase et al. QUICK scheme [7] can be expressed as the upwind scheme plus a correction, namely

$$\Phi_e = \Phi_P + \frac{1}{2}(\Phi_E - \Phi_P), \qquad \text{(CD)} \tag{3.69}$$

$$\Phi_e = \Phi_P + \frac{1}{8}\left(3\Phi_E - 2\Phi_P - \Phi_W\right). \qquad \text{(QUICK)} \tag{3.70}$$

We can now create a generalised expression of these schemes.

$$\Phi_e = \Phi_P + \frac{1}{2}\Psi(\Phi_E - \Phi_P), \tag{3.71}$$

where Ψ is a function determining the properties of the scheme called a flux limiter function. Accurately named so because we are attempting to approximate the convective flux $F_e\Phi_e$ and in order to make a scheme TVD we need to limit the range of values the convective flux $F_e\Psi(\Phi_E - \Phi_P)/2$ can take. This will be discussed in detail later. One can easily see that $\Psi = 0$ is the

standard upwind scheme and $\Psi = 1$ is the central differencing scheme. Some manipulation is needed in order to obtain the Ψ function for the LUD and QUICK schemes.

$$\Phi_e = \Phi_P + \frac{1}{2}\left(\frac{\Phi_P - \Phi_W}{\Phi_E - \Phi_P}\right)(\Phi_E - \Phi_P), \qquad \text{(LUD)} \tag{3.72}$$

$$\Phi_e = \Phi_P + \frac{1}{2}\left[\frac{1}{4}\left(3 + \frac{\Phi_P - \Phi_W}{\Phi_E - \Phi_P}\right)\right](\Phi_E - \Phi_P). \qquad \text{(QUICK)} \tag{3.73}$$

The ratio $(\Phi_P - \Phi_W)/(\Phi_E - \Phi_P)$ is usually expressed with the constant r, which makes the function $\Psi = \Psi(r)$. The generalised scheme is then written as

$$\Phi_e = \Phi_P + \frac{1}{2}\Psi(r)(\Phi_E - \Phi_P), \tag{3.74}$$

where

$$r = \frac{\Phi_P - \Phi_W}{\Phi_E - \Phi_P}. \tag{3.75}$$

The four schemes mentioned can now be expressed as

$$\Psi(r) = 0 \qquad \text{(UD)} \tag{3.76}$$

$$\Psi(r) = 1 \qquad \text{(CD)} \tag{3.77}$$

$$\Psi(r) = r \qquad \text{(LUD)} \tag{3.78}$$

$$\Psi(r) = (3 + r)/4 \qquad \text{(QUICK)} \tag{3.79}$$

If we plot the Ψ functions we get, in figure 3.2, what is known as the $\Psi - r$-diagram which is used to illustrate the region in which the TVD schemes resides.

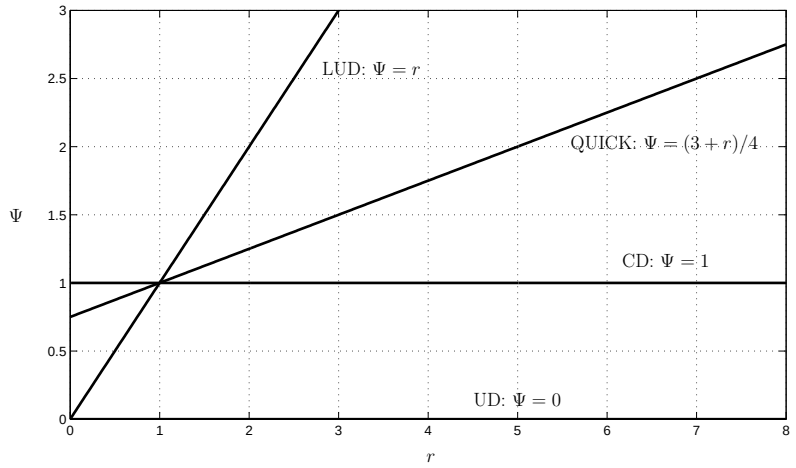


Figure 3.2: $\Psi - r$ -diagram of the four schemes.

3.7.1 Total Variation

Total variation for a conserved property Φ:

$$TV(\Phi^n) = \sum_k |\Phi_k^n - \Phi_{k-1}^n|. \tag{3.80}$$

A system is said to conserve monotonicity if

- it does not create a local extrema

- the value of an existing local minimum is non decreasing and the value of a local maximum is non increasing

For a numerical method to be total variation diminishing it needs to satisfy

$$TV(\Phi^{n+1}) \leq TV(\Phi^n). \tag{3.81}$$

3.7.2 Sweby's TVD Criteria

For a scheme to be TVD it needs to lie within a certain range in the $\Psi - r$ diagram. Sweby (1984) found a necessary and sufficient condition for a scheme to be TVD which limits Ψ for different values of r.

$$\Psi(r) \leq 2r, \qquad 0 < r < 1 \tag{3.82}$$

$$\Psi(r) \leq 2. \qquad r \geq 1 \tag{3.83}$$

The only scheme previously mentioned which is TVD with no restriction on r is the upwind differencing scheme, the other schemes are TVD only for a limited range of r. Sweby also presented a criteria for a TVD scheme to be of second order accuracy which further narrows the possible values the flux limiter function $\Psi(r)$ can take. For a TVD scheme to be of second order accuracy it needs to pass through the point $(1, 1)$ in the $\Psi - r$ diagram. Sweby found that the new possible values for the flux limiter function is now bound by the central difference and linear upwind schemes.

$$\begin{aligned} r \leq \Psi(r) \leq 1 \quad &\text{for } 0 < r < 1, \\ 1 \leq \Psi(r) \leq r \quad &\text{for } r \geq 1. \end{aligned} \tag{3.84}$$

This restriction can be seen as the greyed out area in figure 3.3. In order to simplify any programming of TVD schemes, Sweby proposed another requirement to the flux limiter function called the symmetry property. This requirement makes the limiter function treat both backward and forward facing gradients the same. The symmetry property is

$$\frac{\Psi(r)}{r} = \Psi(1/r). \tag{3.85}$$

When this is satisfied no special coding is needed to treat different gradients.

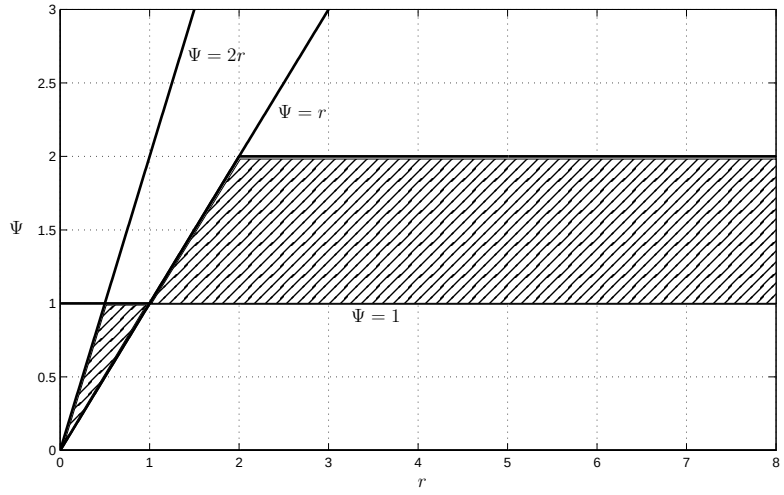


Figure 3.3: $\Psi - r$ -diagram showing the area where a limiter function needs to be in order to be of second order.

3.7.3 Flux Limiter Functions

There exists a great number of flux limiter functions, here we will briefly look at a handful of the most common, which are all of second order.

van Leer (1974)

$$\Psi(r) = \frac{r + |r|}{1 + r}, \quad \lim_{r\to\infty} \Psi(r) = 2$$

van Albada (1982)

$$\Psi(r) = \frac{r + r^2}{1 + r^2}, \quad \lim_{r\to\infty} \Psi(r) = 1$$

UMIST (Lien and Leschziner 1993)

$$\Psi(r) = \max[0, \min(2r, (1 + 3r)/4, (3 + 4r)/4, 2)], \quad \lim_{r\to\infty} \Psi(r) = 2$$

superbee (Roe 1985)

$$\Psi(r) = \max[0, \min(2r, 1), \min(r, 2)], \quad \lim_{r\to\infty} \Psi(r) = 2$$

Sweby (1984)

$$\Psi(r) = \max[0, \min(\beta r, 1), \min(r, \beta)], \ \ (1 \leq \beta \leq 2), \quad \lim_{r\to\infty} \Psi(r) = \beta$$

min-mod (Roe 1985)

$$\Psi(r) = \max[0, \min(r, 1)], \quad \lim_{r\to\infty} \Psi(r) = 1$$

3.7.4 Scheme Properties

TVD schemes satisfies by default conservativeness, transportiveness and boundedness as they are generalisations of existing schemes with these properties.

Numerical accuracy: TVD schemes that follows the limitation (3.84) on the flux limiter are of second order accuracy.

General behaviour: Solutions produced by TVD schemes are in general oscillation free, low in false diffusion and of second order. Complex limiter functions can require more CPU time than regular differencing schemes. As an example, the UMIST scheme uses 15% more CPU time than the standard QUICK scheme. When selecting a TVD scheme to use, there are no arguments for one over the other, the performance difference is negligible.

3.8 Unsteady Convection Diffusion Discretisation

The unsteady transport equation has the form

$$\frac{\partial(\rho\Phi)}{\partial t} + \mathrm{div}(\rho\Phi\mathbf{u}) = \mathrm{div}(\Gamma \mathrm{grad}\Phi) + S_\Phi. \tag{3.86}$$

To account for time we need to integrate over a time step Δt. As usual we also need to integrate over the control volume, which gives

$$\begin{aligned} &\int_t^{t+\Delta t} \left(\int_{\Delta V} \frac{\partial(\rho\Phi)}{\partial t} \mathrm{d}V \right) \mathrm{d}t + \int_t^{t+\Delta t} \left(\int_{\Delta V} \mathrm{div}(\rho\Phi\mathbf{u}) \mathrm{d}V \right) \mathrm{d}t \\ &= \int_t^{t+\Delta t} \left(\int_{\Delta V} \mathrm{div}(\Gamma \mathrm{grad}\Phi) \mathrm{d}V \right) \mathrm{d}t + \int_t^{t+\Delta t} \left(\int_{\Delta V} S_\Phi \mathrm{d}V \right) \mathrm{d}t. \end{aligned} \tag{3.87}$$

We will consider the one dimensional case in which to develop methods for time integration. Each of the terms in equation (3.87) becomes

$$\int_{\Delta V}\left(\int_t^{t+\Delta t}\frac{\partial(\rho\Phi)}{\partial t}\mathrm{d}V\right)\mathrm{d}t=\rho(\Phi_P-\Phi_P^0)\Delta V, \tag{3.88}$$

$$\int_t^{t+\Delta t}\left(\int_{\Delta V}\mathrm{div}(\rho\Phi\mathbf{u})\mathrm{d}V\right)\mathrm{d}t=\int_t^{t+\Delta t}(\rho uA\Phi)_e-(\rho uA\Phi)_e\mathrm{d}t, \tag{3.89}$$

$$\int_t^{t+\Delta t}\left(\int_{\Delta V}\mathrm{div}(\Gamma\mathrm{grad}\Phi)\mathrm{d}V\right)\mathrm{d}t=\int_t^{t+\Delta t}\left(\Gamma A\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_e-\left(\Gamma A\frac{\mathrm{d}\Phi}{\mathrm{d}x}\right)_w\mathrm{d}t, \tag{3.90}$$

$$\int_t^{t+\Delta t}\left(\int_{\Delta V}S_\Phi\mathrm{d}V\right)\mathrm{d}t=\overline{S}_\Phi\Delta V\Delta t. \tag{3.91}$$

In equation (3.89) and (3.90) we do, as before, the substitution $F=\rho u$ and $D=\Gamma/\delta x$. The ΔV terms appearing here is the control volume, which is equal to $\Delta V=A\Delta x$, where A is as usual the face area of the control volume. The width of the control volume is Δx. The right hand side of equation (3.89) becomes

$$F_eA_e\int_t^{t+\Delta t}\Phi_e\mathrm{d}t-F_wA_w\int_t^{t+\Delta t}\Phi_w\mathrm{d}t, \tag{3.92}$$

and the right hand side of equation (3.90) becomes

$$D_eA_e\int_t^{t+\Delta t}(\Phi_E-\Phi_P)\mathrm{d}t-D_wA_w\int_t^{t+\Delta t}(\Phi_P-\Phi_W)\mathrm{d}t. \tag{3.93}$$

For simplicity the terms Φ_e and Φ_w are approximated using central differences

$$\Phi_e=\frac{\Phi_P+\Phi_E}{2}, \tag{3.94}$$

$$\Phi_w=\frac{\Phi_W+\Phi_P}{2}. \tag{3.95}$$

The equation now has the form

$$\begin{aligned}&\rho(\Phi_P-\Phi_P^0)\Delta V+F_eA_e\int_t^{t+\Delta t}\frac{\Phi_P+\Phi_E}{2}\mathrm{d}t-F_wA_w\int_t^{t+\Delta t}\frac{\Phi_W+\Phi_P}{2}\mathrm{d}t\\&=D_eA_e\int_t^{t+\Delta t}(\Phi_E-\Phi_P)\mathrm{d}t-D_wA_w\int_t^{t+\Delta t}(\Phi_P-\Phi_W)\mathrm{d}t+\overline{S}_\Phi\Delta V\Delta t.\end{aligned} \tag{3.96}$$

The time integration of the property Φ needs to be consider in order to advance the calculation. We generalise the integration by means of a weighting parameter θ such that the integration would use values from the current time step, the previous time step or both. The time integral may then be approximated by

$$\int_t^{t+\Delta t}\Phi\,\mathrm{d}t\approx\left[\theta\Phi+(1-\theta)\Phi^0\right]\Delta t. \tag{3.97}$$

Using this integration in equation (3.96) and dividing by $A\Delta t$ we get

$$\begin{aligned}\rho\left(\frac{\Phi_P-\Phi_P^0}{\Delta t}\right)\Delta x+&\frac{F_e}{2}\left[\left(\theta\Phi_P+(1-\theta)\Phi_P^0\right)+\left(\theta\Phi_E+(1-\theta)\Phi_E^0\right)\right]\\&-\frac{F_w}{2}\left[\left(\theta\Phi_W+(1-\theta)\Phi_W^0\right)+\left(\theta\Phi_P+(1-\theta)\Phi_P^0\right)\right]\\&=D_e\left[\left(\theta\Phi_E+(1-\theta)\Phi_E^0\right)-\left(\theta\Phi_P+(1-\theta)\Phi_P^0\right)\right]\\&-D_w\left[\left(\theta\Phi_P+(1-\theta)\Phi_P^0\right)-\left(\theta\Phi_W+(1-\theta)\Phi_W^0\right)\right]\\&+\overline{S}\Delta x.\end{aligned} \tag{3.98}$$

This can be rearranged to give

$$\begin{aligned}
&\left[\left(\frac{F_w}{2}+D_w\right)\theta+\left(D_e-\frac{F_e}{2}\right)\theta+\rho\frac{\Delta x}{\Delta t}+(F_e-F_w)\theta\right]\Phi_P \\
&=\left(\frac{F_w}{2}+D_w\right)(\theta\Phi_W+(1-\theta)\Phi_W^0)+\left(D_e-\frac{F_e}{2}\right)(\theta\Phi_E+(1-\theta)\Phi_E^0) \\
&+\left[\rho\frac{\Delta x}{\Delta t}-(1-\theta)\left(\frac{F_w}{2}+D_w\right)-(1-\theta)\left(D_e-\frac{F_e}{2}\right)\right]\Phi_P^0+\overline{S}\Delta x.
\end{aligned} \tag{3.99}$$

We now recognize the coefficients of Φ_W and Φ_E as a_W and a_E respectively, thus equation (3.99) can be written

$$\begin{aligned}
a_P\Phi_P=a_W\left[\theta\Phi_W+(1-\theta)\Phi_W^0\right]+a_E\left[\theta\Phi_E+(1-\theta)\Phi_E^0\right] \\
+\left[a_P^0-(1-\theta)a_W-(1-\theta)a_E\right]\Phi_P^0+b,
\end{aligned} \tag{3.100}$$

where

$$a_P=\theta(a_W+a_E)+\theta(F_e-F_w)+a_P^0, \tag{3.101}$$

$$a_P^0=\rho\frac{\Delta x}{\Delta t}, \tag{3.102}$$

$$b=\overline{S}\Delta x. \tag{3.103}$$

The time discretisation is heavily dependent upon the parameter θ. Typical choices are $\theta = \{0, 1, 1/2\}$.

The scheme is called **explicit** when $\theta = 0$ is used. It uses only the previous values of Φ in order to advance the solution.

When using $\theta = 1$ the scheme is called **fully implicit** and uses values of Φ from the next time step. This results in a system of equations that needs to be solved for each time step.

Letting $\theta = 1/2$ we get the **Crank-Nicolson** scheme which is also implicit and requires the solution of a system of equations for each time step.

3.8.1 Properties

The properties investigated for spatial discretisation does not apply to the temporal discretisation. For the three methods described we will look at stability and the order of the scheme.

Explicit	Conditionally stable	First order
Implicit	Unconditionally stable	First order
Crank-Nicolson	Unconditionally stable	Second order

3.9 Pressure-Velocity Coupling

In the derivation of the differencing schemes, we assumed that the velocity field was known. This is not the case in a general situation. Therefore some techniques for calculation the entire flow field has been developed. The two most widely used are the SIMPLE and the PISO algorithm. The SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) was proposed in 1972 by Patankar and Spalding. It is an iterative process which guesses the pressure field and then corrects its until a convergence criteria has been fulfilled. The PISO (Pressure Implicit with Splitting of Operators) proposed by Issa (1986) is an extension of the SIMPLE algorithm for calculating unsteady compressible problems. The details of these methods are not in the scope of this paper, therefore an outline of these methods will be presented. In general the SIMPLE algorithm is used for steady state flows and the PISO algorithm for unsteady flows. Therefore the algorithms presented here will be a steady state for the SIMPLE algorithm and unsteady for the PISO algorithm. Details of the following algorithms can be found in [21] and [20].

3.9.1 SIMPLE Algorithm

The SIMPLE algorithm is, as stated before, an iterative process. The first step is to guess a pressure field, p^*, which is then used to solve the discretised momentum equations

$$a_{i,J}u^*_{i,J} = \sum a_{nb}u^*_{nb} + (p^*_{I-1,J} - p^*_{I,J})A_{i,J} + b_{i,J}, \tag{3.104}$$

$$a_{i,J}v^*_{i,J} = \sum a_{nb}v^*_{nb} + (p^*_{I,J-1} - p^*_{I,J})A_{i,J} + b_{i,J}. \tag{3.105}$$

The star, $\cdot^*$, denotes a guessed variable. Then the continuity equation is solved from which a pressure correction equation is obtained,

$$\begin{aligned} a_{I,J}p'_{I,J} = &a_{I+1,J}p'_{I+1,J} + a_{I-1,J}p'_{I-1,J} \\ &+ a_{I,J+1}p'_{I,J+1} + a_{I,J-1}p'_{I,J-1} + b'_{I,J}, \end{aligned} \tag{3.106}$$

where p' is the pressure correction. From this one obtains the corrected pressure and velocities

$$\begin{aligned} p_{I,J} &= p^*_{I,J} + p'_{I,J}, \\ u_{i,J} &= u^*_{i,J} + d_{i,J}(p'_{I-1,J} - p'_{I,J}), \\ v_{I,j} &= u^*_{I,j} + d_{I,j}(p'_{I,J-1} - p'_{I,J}). \end{aligned} \tag{3.107}$$

Following this, all other transport equations are solved, such as temperature and turbulence properties. If the residuals has not converged, a new iteration begins with the guessed variables being the previously solved values. A flowchart of this algorithm can be found in figure 3.4.

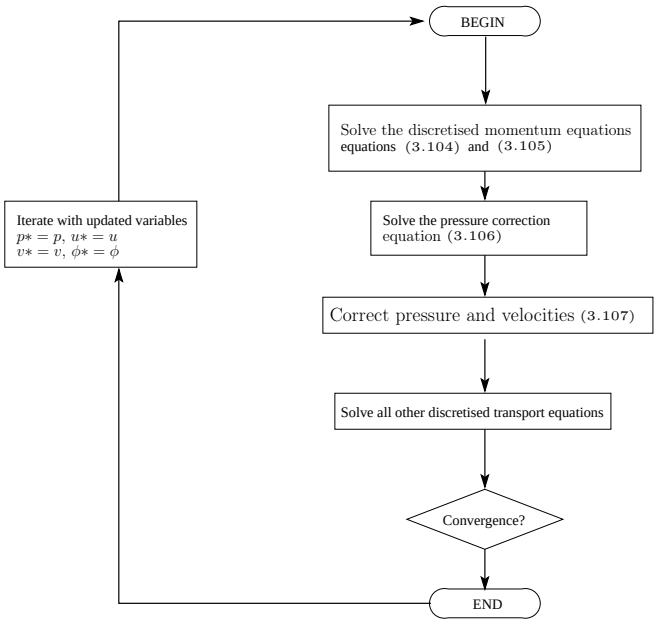


Figure 3.4: The SIMPLE algorithm

3.9.2 PISO Algorithm

The unsteady PISO algorithm is presented here, it can also be used in steady state calculations, but it is commonly used in unsteady problems. The PISO algorithm is similar to the SIMPLE algorithm in that it does a prediction step and a correction step, however the PISO algorithm computes an additional correction. The PISO algorithm starts off with the same three first steps

of the SIMPLE algorithm. However, after the correction of the pressure and velocities the PISO algorithm solves a second pressure correction equation,

$$\begin{aligned} a_{I,J}p''_{I,J} = a_{I-1,J}p''_{I-1,J} + a_{I+1,J}p''_{I+1,J} \\ + a_{I,J-1}p''_{I,J-1} + a_{I,J+1}p''_{I,J+1} + b''_{I,J}. \end{aligned} \tag{3.108}$$

From which the pressure and velocity is corrected for a second time.

$$p^{***}_{I,J} = p^{*}_{I,J} + p'_{I,J} + p''_{I,J}, \tag{3.109}$$

$$\begin{aligned} u^{***}_{i,J} &= u^{*}_{i,J} + d_{i,J}(p'_{I-1,J} - p'_{I,J}) + \frac{\sum a_{nb}(u^{**}_{nb} - u^{*}_{nb})}{a_{i,J}} \\ &+ d_{i,J}(p''_{I-1,J} - p''_{I,J}), \end{aligned} \tag{3.110}$$

$$\begin{aligned} v^{***}_{I,j} &= v^{*}_{I,j} + d_{I,j}(p'_{I,J-1} - p'_{I,J}) + \frac{\sum a_{nb}(v^{**}_{nb} - v^{*}_{nb})}{a_{I,j}} \\ &+ d_{I,j}(p''_{I,J-1} - p''_{I,J}). \end{aligned} \tag{3.111}$$

The outline of the algorithm is presented as a flowchart in figure 3.5.

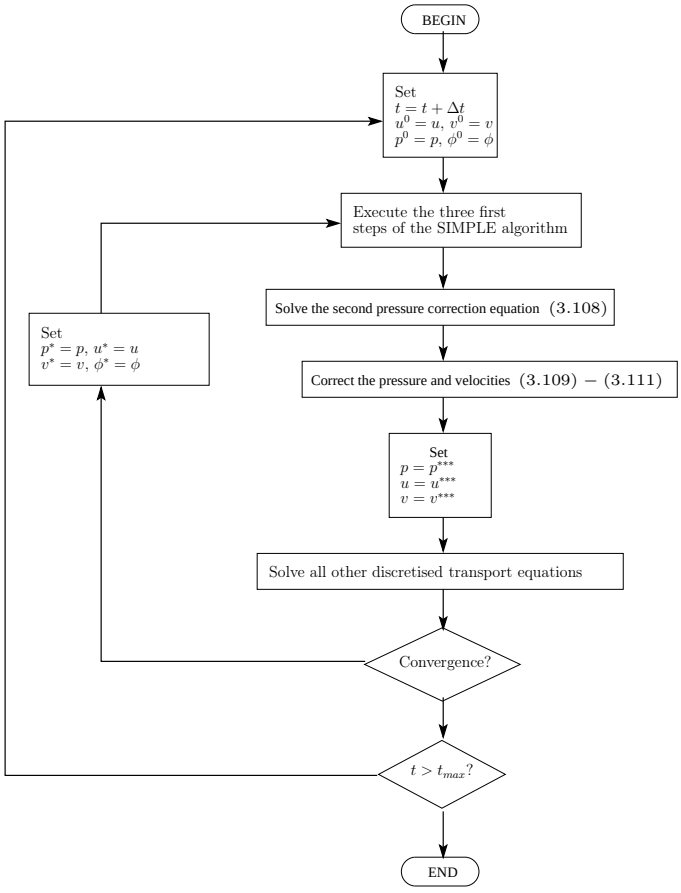


Figure 3.5: The PISO algorithm

4 Turbulence Modelling

Solving the Navier-Stokes equations is very expensive in terms of computing time. Solving the equations directly is called DNS (Direct Numerical Simulation) and requires a large number of grid points for the discretised equations and a small time step in order to capture turbulence interacting on all scales. In order to reduce the cost of DNS a turbulence model is used to model some aspects of the turbulence. The two most commonly used turbulence models is the RANS (Reynolds-Averaged Navier-Stokes) and LES (Large Eddy Simulation). Each model approaches the turbulence modelling differently and will be described in closer detail in this section.

4.1 Reynolds-Averaged Navier-Stokes Equations

The Reynolds-averaged Navier-Stokes (RANS) approach to turbulence is a time averaging of the Navier-Stokes equations. This method needs to only solve for the mean properties of the flow such that the details of the turbulence can be ignored which reduces the computational cost. The RANS turbulence model solves the Reynolds-averaged Navier-Stokes equations which will be derived in this section. We will begin by splitting a property ϕ into a mean value Φ and a time varying fluctuating value ϕ'. This is called a Reynolds decomposition

$$\phi(t) = \Phi + \phi'(t). \tag{4.1}$$

Here Φ is the average value and ϕ' is the fluctuations. The time average of ϕ is defined by

$$\overline{\phi} = \Phi = \frac{1}{\Delta t}\int_0^{\Delta t} \phi(t)\mathrm{d}t, \tag{4.2}$$

and the time average of the fluctuations ϕ' is, by definition, zero

$$\overline{\phi'} = \frac{1}{\Delta t}\int_0^{\Delta t} \phi'(t)\mathrm{d}t \equiv 0. \tag{4.3}$$

The terms in the Reynolds decomposition has a set of rules which will be used when averaging the Navier-Stokes equations. Let $\phi = \Phi + \phi'$ and $\psi = \Psi + \psi'$.

$$\begin{aligned} &\overline{\phi'} = \overline{\psi'} = 0 && \overline{\Phi} = \Phi \\ &\overline{\phi + \psi} = \Phi + \Psi && \overline{\frac{\partial \phi}{\partial s}} = \frac{\partial \Phi}{\partial s} \\ &\overline{\phi\psi} = \Phi\Psi + \overline{\phi'\psi'} && \overline{\phi\Psi} = \Phi\Psi \\ &\overline{\phi'\Psi} = 0 \end{aligned} \tag{4.4}$$

Similarly for divergence and gradient terms we have

$$\begin{aligned} &\overline{\mathrm{div}(\mathbf{a})} = \mathrm{div}(\mathbf{A}), \\ &\overline{\mathrm{div}(\phi\mathbf{a})} = \mathrm{div}(\overline{\phi\mathbf{a}}) = \mathrm{div}(\Phi\mathbf{A}) + \mathrm{div}(\overline{\phi'\mathbf{a}'}), \\ &\overline{\mathrm{div}\,\mathrm{grad}\,\phi} = \mathrm{div}\,\mathrm{grad}\,\Phi. \end{aligned} \tag{4.5}$$

The compressible Navier-Stokes equations without body forces are

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0, \tag{4.6}$$

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j}. \tag{4.7}$$

It would be sufficient to use the time averaging of the compressible Navier-Stokes equations in most situations where the fluctuation in the density is insignificant, whereas it would greatly influence

the turbulence in highly compressible and supersonic flows. However, for generality the Favre averaging will be derived for the compressible Navier-Stokes equations. The Favre averaging is defined by

$$\widetilde{\Phi} = \frac{\overline{\rho\Phi}}{\overline{\rho}}. \tag{4.8}$$

The flow variables can then be decomposed, similarly to Reynolds decomposition, into

$$\Phi = \widetilde{\Phi} + \Phi'', \tag{4.9}$$

where Φ'' contains both the turbulent fluctuation and the density fluctuation. Observe that when the flow is incompressible the Favre averaging becomes the Reynolds averaging; $\widetilde{\Phi} = \overline{\Phi}$ and $\Phi'' \equiv \Phi'$. Also, for Favre averaging, the time average of the fluctuation is not zero $\overline{\Phi''} \neq 0$. The time averaged continuity equation (4.6), when using Favre averaging, becomes

$$\overline{\frac{\partial \rho}{\partial t}} + \frac{\partial}{\partial x_i}(\overline{\rho u_i}) = 0, \tag{4.10}$$

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_i}(\overline{\rho(\widetilde{u}_i + u_i'')}) = 0, \tag{4.11}$$

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_i}(\overline{\rho}\widetilde{u}_i + \overline{\rho u_i''}) = 0, \tag{4.12}$$

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_i}(\overline{\rho}\widetilde{u}_i) = 0, \tag{4.13}$$

where we have used that $\overline{\rho\Phi''} = 0$ and $\overline{\rho\widetilde{\Phi}} = \overline{\rho}\widetilde{\Phi}$. For comparison, the continuity equation without Favre averaging is

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_i}(\overline{\rho}\,\overline{u}_i) + \frac{\partial}{\partial x_i}(\overline{\rho' u_i'}) = 0. \tag{4.14}$$

Similarly we can derive the time average for the momentum equation (4.7) with Favre averaging. Time averaging each term in the momentum equation by itself yields

$$\begin{aligned} \frac{\partial(\overline{\rho u_i})}{\partial t} &= \frac{\partial(\overline{\rho}(\widetilde{u}_i + u_i''))}{\partial t}, \\ &= \frac{\partial(\overline{\rho\widetilde{u}_i} + \overline{\rho u_i''})}{\partial t}, \\ &= \frac{\partial(\overline{\rho}\,\widetilde{u}_i)}{\partial t}, \end{aligned} \tag{4.15}$$

for the first term. The second term, without the divergence for readability, becomes

$$\begin{aligned} \overline{\rho u_i u_j} &= \overline{\rho(\widetilde{u}_i + u_i'')(\widetilde{u}_j + u_j'')}, \\ &= \overline{\rho\widetilde{u}_i\widetilde{u}_j} + \overline{\rho\widetilde{u}_i u_j''} + \overline{\rho u_i''\widetilde{u}_j} + \overline{\rho u_i'' u_j''}, \\ &= \overline{\rho}\widetilde{u}_i\widetilde{u}_j + \overline{\rho u_i'' u_j''}. \end{aligned} \tag{4.16}$$

The two terms on the right hand side simply becomes

$$-\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial \overline{\tau_{ij}}}{\partial x_j}. \tag{4.17}$$

Thus the time averaged momentum equation with Favre averaging becomes

$$\frac{\partial(\overline{\rho}\,\widetilde{u}_i)}{\partial t} + \frac{\partial}{\partial x_j}(\overline{\rho}\widetilde{u}_i\widetilde{u}_j + \overline{\rho u_i'' u_j''}) = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial \overline{\tau_{ij}}}{\partial x_j}. \tag{4.18}$$

Similarly equations for scalar transport and energy can be derived, but will not be presented here. The extra term $\overline{\rho u_i'' u_j''}$ is called the Reynolds stresses. The appearance of this term makes the

Navier-Stokes equation to not be closed. As it is not possible to close the problem with a set of exact equations, details can be found in [6], one has to approximate the Reynolds stresses and the turbulent scalar fluxes with a turbulence model. There are numerous models which can be used to model the turbulence, some are more general than others. In table 4.1 there is a short list of commonly used models grouped by the number of turbulent transport equations to be solved.

Equation type	Model name
Algebraic	Mixing length model
One equation	Prandtl's one equation model Spalart-Allmaras model
Two equation	k-epsilon model k-omega model

Table 4.1: Common turbulence models

The k-epsilon model, being the most used turbulence model due to its simplicity and good performance in most types of flow, will be explored in the following section.

4.1.1 The k-epsilon Turbulence Model

There are several k-epsilon turbulence models, all which has their own strengths and weaknesses. Here we will present the 'standard' k-epsilon model proposed by Launder and Spalding (1974). The k-epsilon turbulence model solves two extra transport equations in order to resolve the turbulent kinetic energy k and the turbulent dissipation ϵ. The basis for the k-epsilon turbulence model is the Boussinesq eddy viscosity assumption, which approximates the Reynolds stresses linearly to the mean strain rate tensor:

$$-\rho\overline{u_i'u_j'} = \mu_t\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i}\right) - \frac{2}{3}\rho k\delta_{ij}. \tag{4.19}$$

The eddy viscosity μ_t is given by

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon}. \tag{4.20}$$

The transport equation for k is

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k\overline{u}_j)}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + \mu_t\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i}\right) - \rho\epsilon, \tag{4.21}$$

and the transport equation for ϵ is

$$\frac{\partial(\rho\epsilon)}{\partial t} + \frac{\partial(\rho\epsilon\overline{u}_j)}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\frac{\mu_t}{\sigma_\epsilon}\frac{\partial\epsilon}{\partial x_j}\right) + C_{1\epsilon}\frac{\epsilon}{k}\mu_t\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i}\right) - C_{2\epsilon}\rho\frac{\epsilon^2}{k}. \tag{4.22}$$

The five constants present in equations (4.20 - 4.22) are usually set to

$$\begin{array}{cc} C_\mu = 0.09 & \\ C_{1\epsilon} = 1.44 & C_{2\epsilon} = 1.92 \\ \sigma_k = 1.0 & \sigma_\epsilon = 1.3 \end{array}$$

4.2 Large Eddy Simulation

Large eddy simulation aims at directly solving only the large scale eddies in a DNS fashion, whereas the small eddies are approximated. In order to separate the large eddies from the small eddies the Navier-Stokes equations are filtered. By specifying a certain cutoff width for a filtering function the large eddies are filtered out from the smaller eddies who's width are smaller than the cutoff

width. As with all filtering functions some information is lost, in this case the information about the smaller eddies are filtered out. In order to capture the interaction between the large eddies and the smaller eddies a model is necessary. These models are known as sub-grid scale models (SGS-models).

4.2.1 Spatial Filtering

In order to filter the Navier-Stokes equations, we need to define a filter operator. This spatial filter operator is defined as

$$\overline{\Phi}(\mathbf{x},t) \equiv \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(\mathbf{x},\mathbf{x}',\Delta)\Phi(\mathbf{x}',t)\mathrm{d}x_1'\mathrm{d}x_2'\mathrm{d}x_3', \tag{4.23}$$

where

$$\begin{aligned} \overline{\Phi}(\mathbf{x},t) &\text{ is the filtered function,}\\ G(\mathbf{x},\mathbf{x}',\Delta) &\text{ is the filter function,}\\ \Phi(\mathbf{x},t) &\text{ is the unfiltered function,}\\ \Delta &\text{ is the cutoff width.} \end{aligned}$$

There are different filter functions for different applications. In research, Gaussian and spectral filtering functions are commonly used. A more practical filter function is the top-hat which is widely used in finite volume implementations:

$$G(\mathbf{x},\mathbf{x}',\Delta) = \begin{cases} 1/\Delta^3 & |\mathbf{x}-\mathbf{x}'| \le \Delta/2 \\ 0 & |\mathbf{x}-\mathbf{x}'| > \Delta/2 \end{cases} \tag{4.24}$$

The choice of Δ determine the size of the smallest eddy that is to be resolved. This cutoff width can be of any size larger than the cell size, as selecting a value less would be pointless. There are different ways of selecting a suitable Δ. It is common to select a cutoff width of the same order as the grid. The most commonly used is the cube root of the cell volume

$$\Delta = \sqrt[3]{\Delta x \Delta y \Delta z}. \tag{4.25}$$

For structured grids it would suffice to take the maximum of $\Delta x, \Delta y$ and Δz.

4.2.2 Filtering The Navier-Stokes Equations

In Cartesian coordinates the incompressible Navier-Stokes equations are

$$\frac{\partial(\rho u_i)}{\partial x_i} = 0, \tag{4.26}$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right). \tag{4.27}$$

When filtering these equations we obtain

$$\frac{\partial(\rho \overline{u}_i)}{\partial x_i} = 0, \tag{4.28}$$

$$\frac{\partial(\rho \overline{u}_i)}{\partial t} + \frac{\partial(\rho \overline{u_i u_j})}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)\right). \tag{4.29}$$

The only term here which cannot be directly computed is the convection term $\overline{u_i u_j}$ since $\overline{u_i u_j} \neq \overline{u_i}\,\overline{u_j}$. This is where we need to introduce a model in our large eddy simulation. By introducing

$$\frac{\partial(\rho\overline{u_i u_j})}{\partial x_j} = \frac{\partial(\rho\overline{u_i}\,\overline{u_j})}{\partial x_j} + \left(\frac{\partial(\rho\overline{u_i u_j})}{\partial x_j} - \frac{\partial(\rho\overline{u_i}\,\overline{u_j})}{\partial x_j}\right) \tag{4.30}$$

equation (4.29) can be written

$$\begin{aligned}\frac{\partial(\rho\overline{u}_i)}{\partial t}+\frac{\partial(\rho\overline{u_i}\,\overline{u_j})}{\partial x_j}=&-\frac{\partial\overline{p}}{\partial x_i}+\frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial\overline{u}_i}{\partial x_j}+\frac{\partial\overline{u}_j}{\partial x_i}\right)\right)\\&-\left(\frac{\partial(\rho\overline{u_iu_j})}{\partial x_j}-\frac{\partial(\rho\overline{u_i}\,\overline{u_j})}{\partial x_j}\right).\end{aligned}\tag{4.31}$$

The last two terms are the ones we are going to approximate, which is called subgrid-scale Reynolds stress

$$\tau_{ij}=\rho\overline{u_iu_j}-\rho\overline{u_i}\,\overline{u_j}.\tag{4.32}$$

The name 'stress' is not directly applicable here as most of what τ_{ij} represents comes from the convective momentum transport due to the action of the unresolved eddies. These SGS-stresses requires a turbulence model. The Smagorinsky model will be presented in the following section.

4.2.3 Filtering Compressible Navier-Stokes Equations

Favre filter is a density-weighted filtering operation defined as

$$\widetilde{\Phi}=\frac{\overline{\rho\Phi}}{\bar{\rho}}.\tag{4.33}$$

By using the Favre averaging the averaged Navier-Stokes equations become

$$\frac{\partial\widetilde{\rho}}{\partial t}+\operatorname{div}(\bar{\rho}\widetilde{\mathbf{u}})=0\tag{4.34}$$

$$\frac{\partial(\bar{\rho}\widetilde{\mathbf{u}})}{\partial t}+\operatorname{div}(\bar{\rho}\widetilde{u}\widetilde{\mathbf{u}})=-\frac{\partial\bar{p}}{\partial x}+\mu\operatorname{div}(\operatorname{grad}(\bar{u}))+F_x\tag{4.35}$$

$$\frac{\partial(\bar{\rho}\widetilde{\mathbf{u}})}{\partial t}+\operatorname{div}(\bar{\rho}\widetilde{v}\widetilde{\mathbf{u}})=-\frac{\partial\bar{p}}{\partial y}+\mu\operatorname{div}(\operatorname{grad}(\bar{v}))+F_y\tag{4.36}$$

$$\frac{\partial(\bar{\rho}\widetilde{\mathbf{u}})}{\partial t}+\operatorname{div}(\bar{\rho}\widetilde{w}\widetilde{\mathbf{u}})=-\frac{\partial\bar{p}}{\partial z}+\mu\operatorname{div}(\operatorname{grad}(\bar{w}))+F_z\tag{4.37}$$

where bar denotes averaging and tilde denotes Favre averaging.

4.2.4 Smagorinsky Turbulence Model

This model proposed by Smagorinsky (1963) is a widely used eddy viscosity model and was one of the earliest models to be used in large eddy simulation. The approximation used in this model is that the local SGS-stresses is proportional to the local strain rate of the flow

$$\tau_{ij}=-2\mu_{SGS}\overline{S}_{ij}+\frac{1}{3}\tau_{ij}\delta_{ij},\tag{4.38}$$

where

$$\overline{S}_{ij}=\frac{1}{2}\left(\frac{\partial\overline{u}_i}{\partial x_j}+\frac{\partial\overline{u}_j}{\partial x_i}\right).\tag{4.39}$$

The proportionality constant here is the dynamic SGS viscosity μ_{SGS}. This viscosity term is evaluated from

$$\mu_{SGS}=\rho C_S^2\Delta^2|\overline{S}|,\tag{4.40}$$

where $|\overline{S}|=\sqrt{2\overline{S}_{ij}\overline{S}_{ij}}$, Δ is the filter length scale and C_S is a problem dependent parameter called the Smagorinsky constant. This constant is generally agreed upon to be approximately 0.2, however values ranging from 0.065 to 0.24 have been used successfully [6, 21].

Setting the parameter C_S to a constant is not easily justified as one can imagine, this constant is highly dependent on geometry, the Reynolds number or other flow properties. To improve this constant the dynamic SGS model was proposed (Germano et al., 1991). In this model the local values of C_S are computed for each time step. The details of dynamic models will not be explored here, more information can be found in [6].

5 OpenFOAM

The focus of this paper is not to develop new code for solving fluid dynamics with the finite volume method. It focuses on using existing code and analyse the output concerning sound generated by corrugated pipes. The choice of OpenFOAM (Field Operation And Manipulation) for this paper was an easy task as it is both free and of high quality. A circumstantial reason for selecting this code is that the University of Bergen currently uses OpenFOAM on their supercomputer running in parallel.

5.1 Introduction

OpenFOAM is mainly a C++ toolbox for computational fluid dynamics, and consists of numerical solvers, pre-processors and post-processors. All of which are open source under the GNU General Public License. The toolbox comes with a large set of pre compiled applications divided into two categories; solvers and utilities. The different solvers are distributed in twelve categories ranging from continuum mechanics to electromagnetics to finance. In utilities there are a number of different applications for mesh generation and manipulation, pre-/post-processing and conversion tools.

5.2 Selecting a Solver

OpenFOAM has a vast range of solvers for incompressible and compressible flows. This problem needs a compressible solver in order to capture the sound waves. The *rhoPimpleFoam* solver seems to be a good choice for our problem. It is a transient compressible solver with turbulence. It uses a merged PISO-SIMPLE algorithm.

5.3 Boundary Conditions

To obtain a good numerical simulation the boundary conditions need to be properly specified. More often than not, the boundary conditions is the source of a bad simulation. Here we will prescribe the boundary conditions for the problem at hand. Only the velocity and pressure will be given in detail, the remaining boundary conditions for other transport values are given in appendix. In all the simulations, the geometry consists of four patches on which boundary conditions are applied; inlet, outlet, walls and axis. The axis will primarily be either a symmetry plane or axis of rotation.

5.3.1 Inlet

For the inlet we will use a velocity inlet.

Velocity (U): A constant velocity parallel to the pipe. A velocity of 8 m/s in the x-direction would be written as

```
type            fixedValue;
value           uniform (8 0 0);
```

Pressure (p): Since the inlet velocity has been explicitly set, the pressure needs to be able to vary. Therefore we will set the pressure to have a gradient of zero. If we wanted to drive the flow by pressure, we would set the pressure to a constant number and let the velocity vary at the inlet. In OpenFOAM this boundary condition is written

```
type            zeroGradient;
```

This is of course a Neumann boundary condition where we specify the gradient of the variable, such that the actual value is able to change.

5.3.2 Outlet

The outlet is a difficult patch to specify boundary conditions on. We want the outlet to be a free (or open) boundary condition and should ideally not influence the solution upstream. However this is difficult using standard boundary conditions (Dirichlet/Neumann). To make it simpler the outlet boundary should be moved sufficiently far away from the domain of computational interest. This will help in minimizing the non-physical effects from a insufficiently defined boundary.

Velocity (U): The velocity will be a mix of Dirichlet and Neumann boundary condition. In OpenFOAM this is called *inletOutlet*, which sets the boundary to Dirichlet if the velocity gradient is directed into the domain and sets the boundary to Neumann if the gradient is directed out of the domain.

```
type            inletOutlet;
inletValue      uniform (0 0 0);
value           uniform (0 0 0);
```

Pressure (p): The pressure boundary condition can be difficult to set. As this is supposed to be a free boundary (atmosphere), a fixed value of atmospheric pressure could suffice. This boundary condition is unfortunately very reflective. However, in OpenFOAM there is a boundary condition called *waveTransmissive* which is very successful in eliminating the reflected pressure waves off the boundary condition.

```
type            waveTransmissive;
field           p;
phi             phi;
rho             rho;
psi             psi;
gamma           1.3;
fieldInf        1e5;
lInf            1;
value           uniform 1e5;
```

5.3.3 Walls

The walls represents the physical walls of the pipe and the boundary conditions for velocity and pressure are easy to determine. The turbulence parameters needs more attention, as one can use a wall function to account for behaviour in the boundary layer. In this case we will not need a wall function as we aim at fully resolving the boundary layer with enough grid points. The requirements for this can be found in [20].

Velocity (U): The velocity will be a no-slip boundary condition at the walls.

```
type            fixedValue;
value           uniform (0 0 0);
```

Pressure (p): The pressure does not have a constant value at the walls and will be therefore prescribed a Neumann boundary condition.

```
type            zeroGradient;
```

5.3.4 Other

For a compressible and turbulent simulation we need to specify more than just the velocity and pressure. The other variables needed by OpenFOAM in a compressible simulation are

- Temperature (T)

- α_{SGS} (alphaSgs)
- μ_{SGS} (muSgs)

The temperature is closely connected to the density and pressure and needs to be specified. In this case the temperature will be set to 300 degrees Kelvin in the entire domain. The two turbulence parameters α_{SGS} and μ_{SGS} are computed at run time and is given the *zeroGradient* boundary condition. Note that, if we were to use a wall function it would be defined in the α_{SGS} file.

5.4 Numerical Schemes

5.4.1 Time Derivative

When running LES we want to reduce the temporal diffusion as much as possible which is why we select a small time step and employ a second order time discretisation scheme. A first order scheme would be too diffusive in time. In OpenFOAM we have two second order schemes, namely the *backward* and the *Crank-Nicolson* scheme. It will be sufficient to use the backward scheme as the increase in accuracy and storage requirements from the Crank-Nicolson scheme would not be worth the increase in computational cost.

5.4.2 Spatial Discretisation

OpenFOAM requires a discretisation scheme for each of the three operators; gradient, divergence and Laplace. For most of the time, the gradient and Laplace discretisation schemes are never changed.

The gradient discretisation scheme is set to be the central difference, any other schemes would be very rarely used. The other choices OpenFOAM has for the gradient is a second order least squares and a fourth order least squares.

The only choice of discretisation for the Laplacian is the Gauss scheme with a interpolation scheme. For interpolation the central difference is often used, but other schemes are available [14].

The choice of discretisation scheme for the divergence operator will have significant influence in the solution. Same as the laplacian, the only scheme available is the Gauss scheme with an interpolation. Common choices of interpolation schemes are the upwind, linear and TVD schemes discussed in section 3. The choice of discretisation scheme is discussed in the Cases section.

5.5 Mesh Generation

Second to a properly defined problem is the generation of a good mesh. Any computation done with a bad mesh can in general not be trusted, which is why much care is needed when generating a mesh. When running LES a sufficiently fine mesh is needed in order to capture the eddies which are of interest. Therefore we can expect to have to generate a mesh with a large amount of cells. The cell density will be larger at the walls in order to resolve the boundary layer behaviour. The cells of a three dimensional mesh are usually either tetrahedrons or hexahedrons. If possible one should use hexahedral cells which offers superior accuracy to tetrahedra with the same amount of cells. Therefore OpenFOAM's own meshing software, *blockMesh*, is a decent way of producing the mesh for this problem as it produces hexahedral cells. The block mesh utility has limited capabilities when it comes to complex geometries, but should be sufficient for generating a corrugated pipe.

There are two important parts of a mesh which requires special care when constructing, the inlet and the outlet. These two parts are difficult to obtain a physical solution at, as errors can propagate upstream/downstream and introduce inaccuracies in the solution. A common solution to this problem is to move the inlet and outlet far away from the domain of interest such that any errors produced are negligible. In my simulations, the inlet cannot be easily moved a significant

amount in the upstream direction and may therefore be a source of error. The reason behind this will be explained in section 7.

5.6 Turbulence Model

For the problem of sound generated in corrugated pipes we need a turbulence model which can resolve the existence of pressure waves. In OpenFOAM we have the choice between Reynolds-averaged simulation (RAS) and large eddy simulation (LES). We will choose for the latter one. The RAS model does a time averaging which is unable to account for pressure waves. The LES model resolves all the eddies down to a specified cut-off width and models the smaller eddies, making it suitable for resolving the pressure waves. DNS (direct numerical simulation) is also a possibility, but for this type of problem it would be prohibitive to compute the solution as the mesh resolution needed would be far greater than LES and a much smaller time step would be needed.

The LES requires a sub-grid scale model. In OpenFOAM we have a number of models to choose from. The most frequently used are

- Smagorinsky
- Dynamic Smagorinsky
- One equation eddy
- Spalart Allmaras

However, in OpenFOAM the dynamic Smagorinsky model is not available for compressible calculations. In the paper [15] the Smagorinsky sgs model was successfully used in computing the flow through a corrugated pipe, which will also be used in these simulations. The Smagorinsky sub grid scale model requires a constant C_s which determines the behaviour of the model. The C_s constant usually lies between $C_s = 0.065$ and $C_s = 0.2$ [18, 16] and for pipes the value $C_s = 0.1$ was found to be optimal [16]. In OpenFOAM the Smagorinsky constant is not directly specified, it is calculated from two constants named C_e and C_k. The implementation of the Smagorinsky model in OpenFOAM has roots in the paper by Sullivan [18]. They are related to the Smagorinsky constant by

$$C_s^2 = C_k \sqrt{\frac{2C_k}{C_e}}. \tag{5.1}$$

The value of these constants are discussed in section 6.2.

6 Flow over Corrugations

This paper will mainly deal with two cases. This first case will examine the flow over corrugations in OpenFOAM. The second case will be a study of the flow through a small pipe with the length of 294 mm. In this section we will deal with setting up different parameters for the simulation, such as sub grid scale model for the LES, solver settings and mesh size. The second case will examine the onset of a standing wave inside the corrugated pipe.

6.1 Geometry

The pipe has an inner radius of 5.5 mm and and an outer radius of 8 mm, such that the cavity depth is 2.5 mm as shown in figure 6.1. The width of the cavity is 2 mm. The bend into the cavity is a quarter circle with radius 0.5 mm and the inside of the cavity is a square. The actual pipe is not square inside the cavity, this but should not influence the sound generating mechanism as there is little flow at the bottom of the cavity.

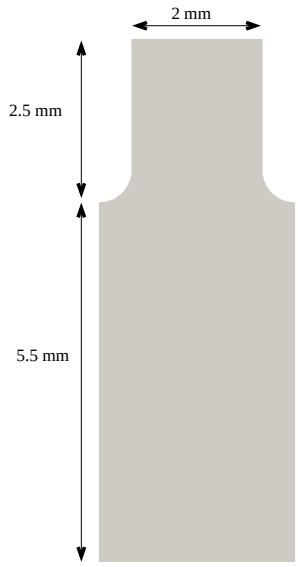


Figure 6.1: Single corrugation

In most computational fluid dynamics problem the dimensionless Reynolds number tells a great deal about the problem in regards to turbulence. We need to ensure that the velocity in the pipe is large enough to generate turbulence in order for the vortex shedding to occur, which is important for the sound generating mechanism [15]. When calculating the Reynolds number for a flow around a cylinder the characteristic length is the diameter of the cylinder. Thus, with this geometry the dimensionless Reynolds number,

$$Re = \frac{\rho v L}{\mu} = \frac{vL}{\nu}, \tag{6.1}$$

is approximately 6000 with a velocity of $6m/s$, a characteristic length of $L = 0.016m$ and the kinematic viscosity at atmospheric pressure and temperature $K = 300$ is $\nu = 1.568 \cdot 10^{-5} m^2/s$. It has been discussed, Cadwell [3], as to which Reynolds number is needed in order for the pipe to sing. Classic turbulence theory suggests a Reynolds number of approximately 2000 [2] is needed in order for turbulence to evolve. This however, as we will see, is not the case for a corrugated pipe where the turbulence mechanism is different. Cadwell presented two methods of determining the onset of turbulence which would generate sound. One could either use the diameter D of the pipe (diameter-induced turbulence) or the corrugation length d (corrugation-induced turbulence) to calculate the

Reynolds number. From experimental values he found that singing could occur at Reynolds numbers lower than 2000 when using the diameter to compute the Reynolds number. These experiments lead to the conclusion that the singing is caused by corrugation-induced turbulence and suggested a Reynolds number for corrugated pipes at which turbulence develops,

$$Re_{corr} \approx 500. \tag{6.2}$$

Thus, by setting the characteristic length to $l = 0.002m$, the new Reynolds number is approximately 765. Both calculations of the Reynolds number should yield a turbulent flow.

6.2 Simulation Parameters

In 2009 SINTEF published a paper (Popescu [15]) describing the flow induced acoustics in corrugated pipes. In the paper they used Fluent 6.3 to carry out numerical simulations on a single corrugation and this section will attempt to produce similar results using OpenFOAM 2.0. However, after some comparisons between a single corrugation and a section with three corrugation it was clear that the single corrugation had trouble maintaining stability during simulation. This seems to be caused by the inlet being very close to the outlet. As a result of these tests the geometry will feature three corrugations when doing the initial analysis such that any complications related to the two boundary conditions being very close to each other are eliminated. This should have no impact on the flow as it is still an infinite pipe being computed.

The geometry of the single corrugation with key measurements is presented in figure 6.1. In the paper by Popescu [15] a rotational symmetry was used, which will also be used here as well as a pure 2D simulation. The boundary conditions proposed in section 5.3 will be used with small modifications to the inlet and outlet in order to recycle the flow. The boundary conditions at the inlet and outlet will be a periodic boundary condition which will be equivalent to an infinite length pipe such that the necessary features of the flow is able to properly develop. In this case the behaviour of the flow inside and around the cavity is what we wish to capture. In OpenFOAM there are several different boundary conditions which could accomplish this and even a dedicated solver. However, each boundary condition behaves differently and needs to be tested. The dedicated solver was not an option as it was not able to solve for compressibility or LES. Initially the *cyclic* boundary condition looked promising, but is unable to drive the flow and is therefore not suitable for periodic inlet and outlets. The boundary condition named *directMappedFlowRate* should accomplish what is needed in order to recycle the outlet to the inlet, see listing 6 in the appendix. This should be equivalent to the mass flow boundary condition used by Popescu [15]. As the simulation will be using LES for the turbulence and the Smagorinsky sub grid scale model, the Smagorinsky constant needs to be determined. However, there is no universal Smagorinsky constant which would fit all types of problems, but as discussed previously a value of 0.1 is a good starting point for pipe flow. In OpenFOAM the Smagorinsky constant is computed by equation (5.1) where both C_e and C_k can be varied. A natural choice is to select these two constants in close vicinity to the standard values in such a way that the Smagorinsky constant becomes 0.1. The two constants were chosen to be $C_e = 1.59$ and $C_k = 0.043$. The solver used is the *rhoPimpleFoam* which in essence is the PISO algorithm with several outer correctors, where the PISO algorithm uses one outer corrector. This allows for relaxation of the solution, however from several test runs it appears that the relaxation factors is off little use. Therefore no relaxation of the solver will be used. If this had been a steady state simulation, then the relaxation of the solution would have been much more influential of the convergence rate.

When computing a LES problem the numerical schemes of the convective flow are of great importance. It is desirable to use a low-dissipation scheme which is of high order. This means that the standard upwind scheme is unsuitable as it is first order and highly diffusive. The central difference is of second order and is a viable candidate, but is too diffusive. OpenFOAM does however have a modified central difference scheme which is filtered in order to reduce staggering by introducing small amounts of upwind [13]. In OpenFOAM this discretisation scheme is called *filteredLinear* and will be used for the momentum. All other discretisations will use central difference. The mesh requires sufficient resolution close to the wall to resolve the effects from the

boundary layer and a sufficient resolution over all for the LES in order to capture the eddies of interest. At the wall the cell size is $\Delta x \approx 1.6 \cdot 10^{-5} m$, which grows to $\Delta x \approx 6 \cdot 10^{-5} m$ at the axis of the pipe. The mesh is shown in figure 6.2. Total cell count for a single corrugation is 10400 cells.

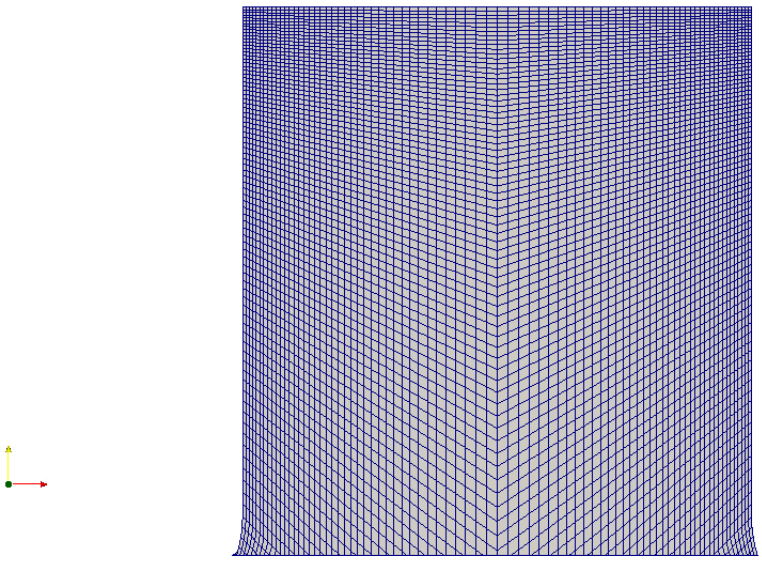


Figure 6.2: Mesh resolution inside a corrugation

As this set up results in an infinite pipe, it is not possible for a standing wave to be maintained and the results should be used to gain insight in the ability for a pipe to sing [15]. Nonetheless, the vortex shedding frequency should be obtainable and thus also the Strouhal. The dimensionless Strouhal number is defined as

$$St = \frac{fL}{v}, \tag{6.3}$$

where f is the vortex shedding frequency, L is the characteristic length and v is the velocity of the fluid. In the case for a flow past a cylinder the characteristic length L is the hydraulic diameter and is set to be the diameter of the cylinder. The Strouhal number is constant for any diameter and fluid velocity for a flow past a cylinder. For flow inside a corrugated pipe, the diameter may not be a good choice for the characteristic length. In the paper by Popescu [15] they used the pitch length as the characteristic length and found that the Strouhal number varied for different pitch lengths, where it should remain constant as with the Strouhal number for a flow past a cylinder. Therefore the pitch length does not represent a very good choice for characteristic length for corrugated pipes. However, since the Reynolds number is calculated using the pitch length, it does seem intuitive to use the pitch length as the characteristic length. The cavity width would be a better choice as it remains constant even if the pitch length changes. This was also noted by Tonon [19] and Kristiansen and Wiik [9].

6.2.1 2D Symmetric Mesh

The two dimensional mesh is a cut out of the pipe in the direction of the axis with a plane of symmetry at the axis in order to reduce the computational cost. The velocity field was initiated by the solver called *potentialFoam*, the solver is described as *"Simple potential flow solver which can be used to generate starting fields for full Navier-Stokes codes"* in the user guide [14]. This will help avoid problems which may occur when starting the simulation from zero. The inlet was set to have a velocity of $8\,m/s$ and *potentialFoam* produced the flow field depicted in figure 6.3.

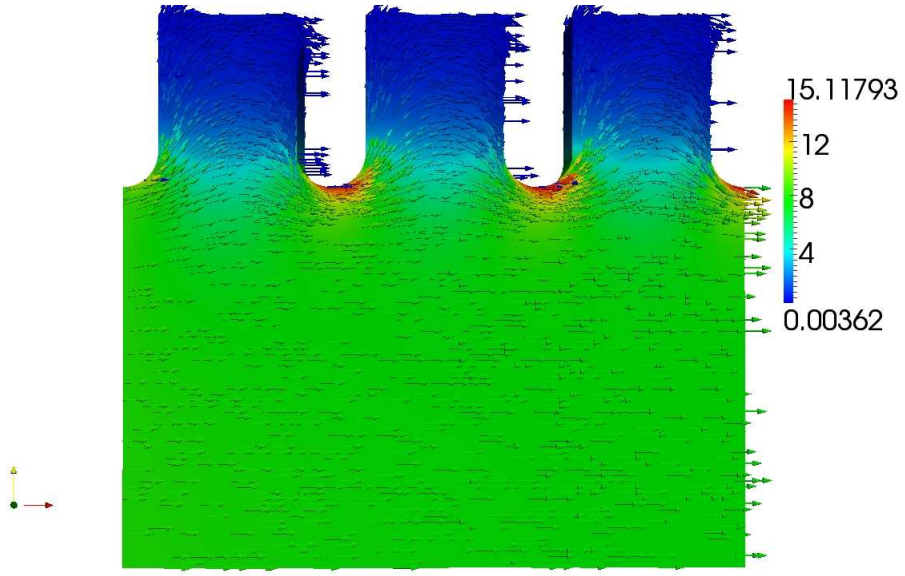


Figure 6.3: Initial value for the velocity field

When the simulation starts, it can be seen that a vortex forms at the edge of the corrugation and the vortex travels across the width of the corrugation until hitting the other edge. After this, the vortex travels back to the center of the corrugation where it remains, see figure 6.4. The vortex appears close to the open pipe, such that most of the fluid inside the corrugation remains at rest. The formation of secondary vortices closer to the corrugation bottom and at the corrugation entrance has been observed, which most likely has little influence of the overall sound generation mechanism. The vortex inside the corrugation appears at any velocity. However, the mere existence of the vortex does not guaranty any sound generation as the vortex itself needs to be strong enough to make any significant change in the pressure which in turn generates the acoustic field.

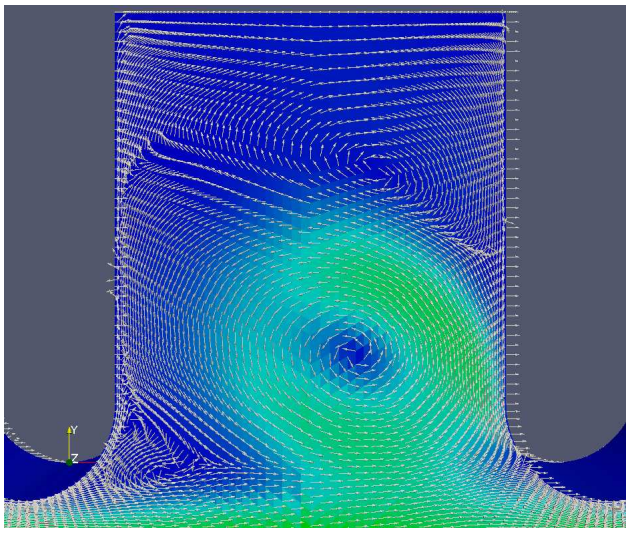
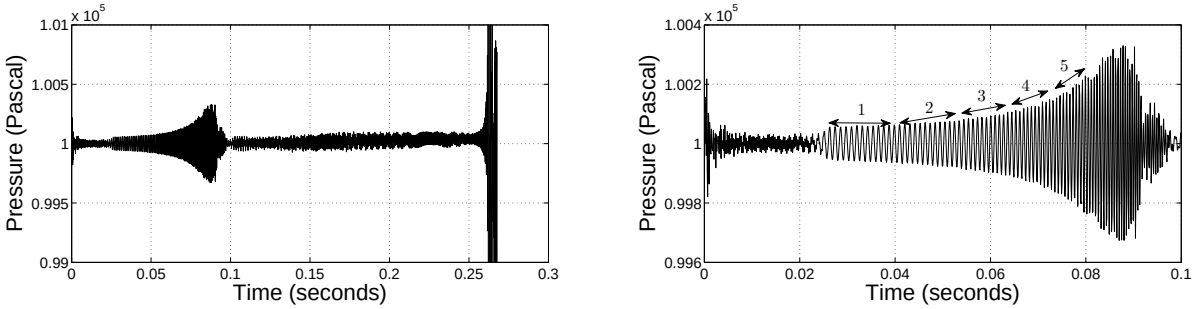


Figure 6.4: Vortex (t=0.003s)

It is clear that the flow over a corrugation gives rise to a vortex inside the corrugation, which is what was to be expected. The shedding of this vortex is the main component in the generation of

sound waves generated by corrugated pipes. The center of the corrugation was probed to extract the pressure over time. If we look at how the pressure behaves over time we observe a periodic event. This event grows in frequency and amplitude over time until it reaches a point ($t = 0.09s$) where it seems to be reset, see figure 6.5a. This happens twice and the simulation diverges the second time.



(a) Pressure inside the corrugation over time

(b) Pressure over time up to $t = 0.1s$

Figure 6.5: Pressure inside the corrugation over time

If we take a closer look at the pressure between $t = 0s$ and $t = 0.09s$ we see a clear sinusoidal behaviour with increasing frequency over time. The reason for the increase in frequency is difficult to determine as there may be several factors causing this behaviour. It could be because this is an infinite pipe and the frequency increases with the distance travelled, or it could be a consequence of the periodic boundary conditions which causes a drift in the velocity. Taking this event to be the vortex shedding we can calculate the Strouhal number at sections (see the sections in figure 6.5b) in time where the frequency is approximately constant. As the velocity increases, so should the frequency in order to keep the Strouhal number constant. The following figures shows the frequency plotted against the velocity and the calculated Strouhal number.

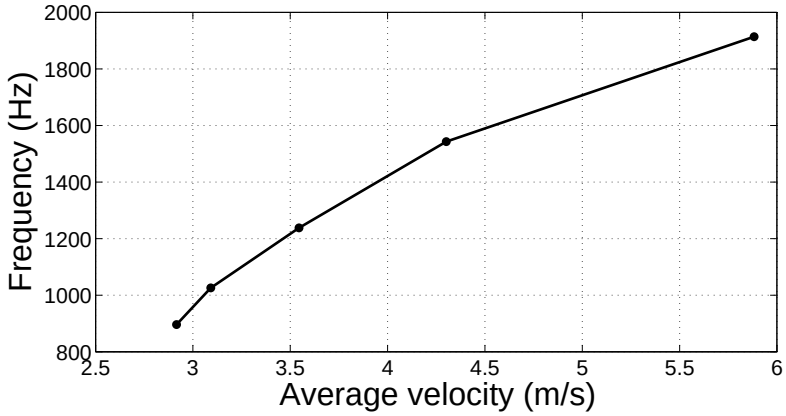


Figure 6.6: Frequency plotted against average velocity in the system

In figure 6.6 we see that as the velocity increases and the frequency increases linearly with the velocity. This is expected as the Strouhal number should ideally remain constant for any velocity.

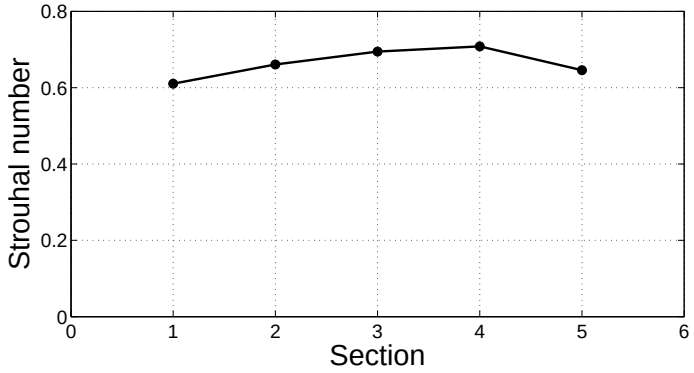


Figure 6.7: Strouhal number over time

From figure 6.7 we see that the Strouhal number for each frequency is nearly constant, but varies between 0.6 and 0.7. This deviation may be attributed to the averaging of the velocity used in the calculation of the Strouhal number. It is common to observe Strouhal numbers around 0.6 in corrugated pipes [12, 19]. This is a good indication that the solver selected and the solver parameters are able to resolve the physics of the singing corrugated pipe. However, the solution is not at all stable and we will next discuss the inadequacy of the boundary condition.

When observing unwanted results, it can be insightful to look at important properties of the flow. In this case we will look at the average velocity and mass over time which is plotted in figures 6.9 and 6.8. This can give valuable information about the behaviour of the solution and give indication of what the problem might be.

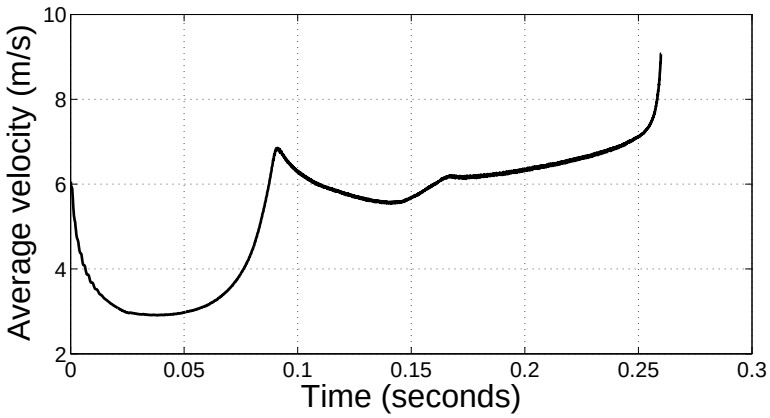


Figure 6.8: Average velocity in the system over time

The average total velocity in the system over time is increasing steadily past $0.15s$ and ultimately diverges. The initial drop in the average velocity figure is explained by how the initial value was obtained, namely the use of the *potentialFoam* solver described previously. This method of setting the initial flow field seem to not fully describe the flow field as there is no vortex appearing inside the corrugations and subsequently the velocity in some areas is higher, see figure 6.3. This causes the solution to undergo a spin up phase in order to reach the correct flow field. As the simulation starts, the flow begins to form vortices in the corrugations and the higher velocity areas disappears reducing the average velocity of the entire domain.

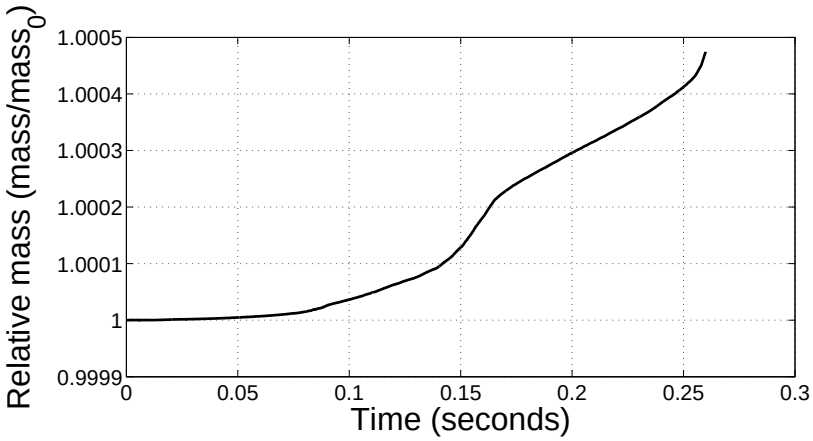


Figure 6.9: Relative total mass in the system over time

The increase in mass, seen in figure 6.9, is almost 0.05% when the simulation diverges, which is not too good as the solver should be able to maintain the total mass in the system. There is, however, virtually no increase in mass from the start of the simulation up to $t = 0.05s$, which could suggest that the simulation past that point already began to run badly. This is further backed up by the average total velocity shown in figure 6.8, where at $t = 0.05s$ the average velocity began to increase. By calculating the mass flow rate at the inlet, see figure 6.10, and outlet it is clear that the mass flow is not constant over time and follows the same curve as the average velocity in the interval 0 to 0.1 seconds. Figure 6.10 does not show the mass flow at the outlet boundary as it is identical to the inlet mass flow.

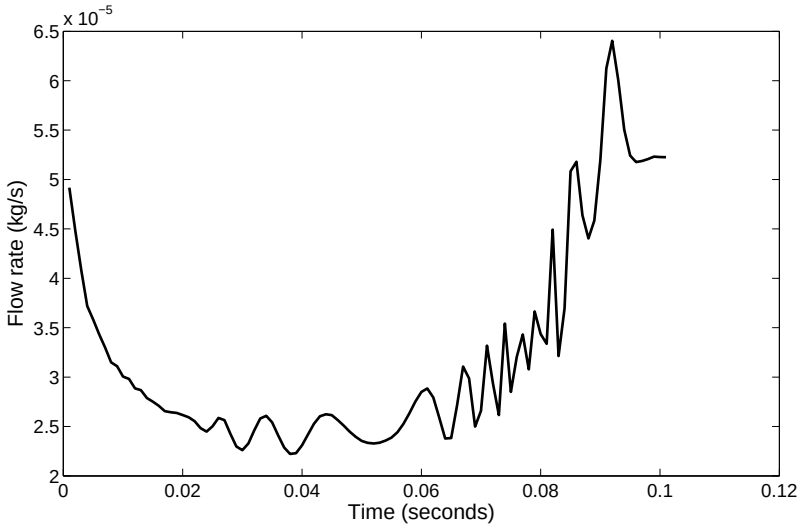


Figure 6.10: Mass flow at the inlet boundary

The most likely source to this behaviour must be within the implementation of the boundary condition. As there is little documentation on how to use the *directMappedFlowRate* boundary condition, there is a chance that the boundary condition was not set correctly. As the name suggests, this boundary condition uses flow rate in order to drive the flow. In order to obtain more information about the boundary condition one has to study the source code. When inspecting the source code for this boundary condition, it is apparent that it does only preserve the velocity component which is normal to the boundary. In this case the velocity component in the x-direction

is preserved over the boundary, but not the y-direction, which is set to zero. This may have an adverse affect on how the velocity field develops over time as the inlet does not carry over the correct velocity components from the outlet. This explains why the mass flow is able to remain the same through the boundary, and may very well have a connection to the over all behaviour of the solution. Therefore one can argue for that this boundary condition may be unsuitable for simulating an infinite pipe. The apparent inadequacy of the *directMappedFlowRate* boundary condition shifts my attention to a different boundary condition in OpenFOAM called *directMapped*. Originally this boundary condition was discarded as it was not apparent in how it would drive the flow except for the option in the code called *average*. The average option rescales the flow at the boundary such that the average velocity is equal a pre set value. The averaging did not seem appropriate in this case as it may average out the oscillations caused by the flow over corrugations. Nonetheless this boundary condition has to be explored.

The changes to the boundary condition for the inlet velocity can be seen in listing 7 in the appendix. The same initial field, see figure 6.3, is used here as with the previous boundary condition. The boundary condition will be tested with the average option set to true and false. The simulation quickly diverged when the average option was set to true and was not investigated further. With the average option set to false the simulation was stable and produced interesting results. The total average velocity shows the same behaviour with the dip in the beginning and then increasing. This time, however, the velocity settled at $5m/s$ as shown in figure 6.11. By inspecting the source code and the simulation result, it was found that contrary to the *directMappedFlowRate* boundary condition the *directMapped* boundary condition *does* preserve all velocity components from the outlet boundary.

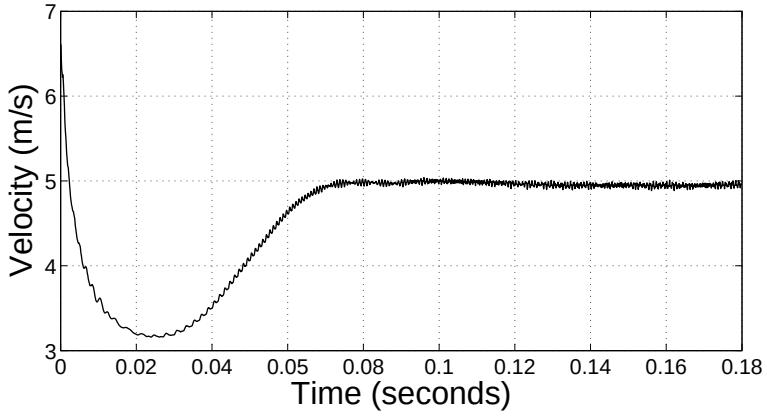


Figure 6.11: Total average velocity

When computing the total mass in the system over time, see figure 6.12, we can see a continuous increase in mass over time, as with the previous case. This may suggest that the increase in mass is a bi-product of the periodic boundary condition or even a fundamental problem with the solver. It is more likely that the boundary condition is the problem here as the solver has most likely undergone careful testing.

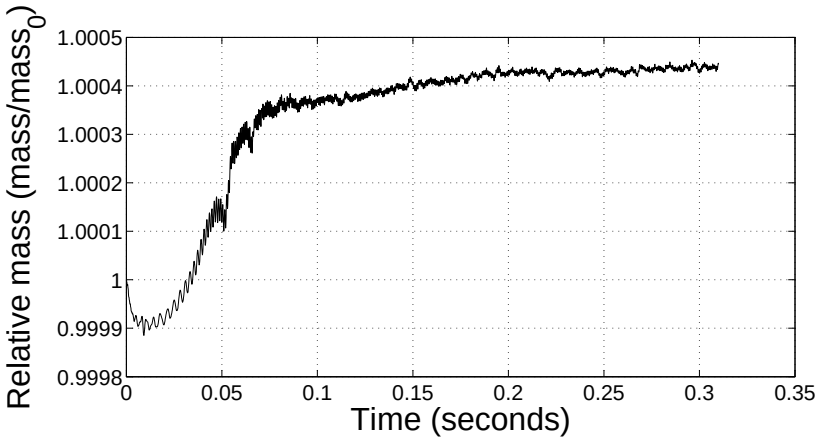


Figure 6.12: Total relative mass in the system

In order to compute the Strouhal number the pressure needs to be plotted over time and a fast Fourier transform has to applied to retrieve the frequencies present in the pressure. The pipe will be probed in two locations; the first inside the corrugation and the second a distance outside the corrugation. The probes can be seen in figure 6.13.

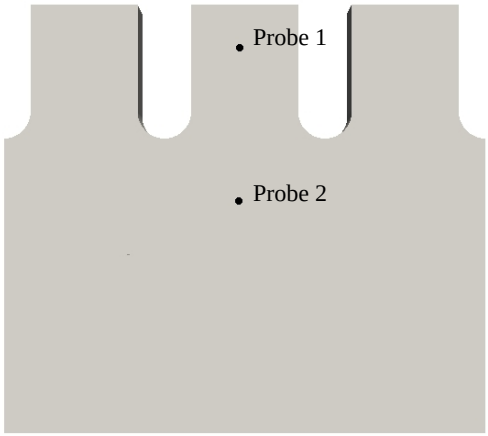
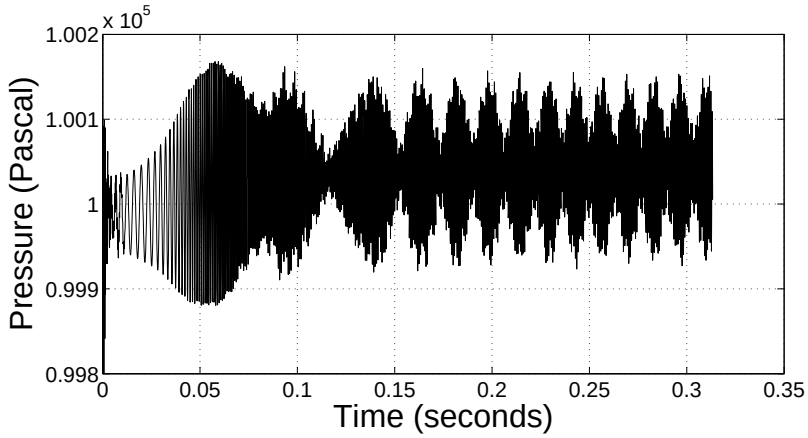


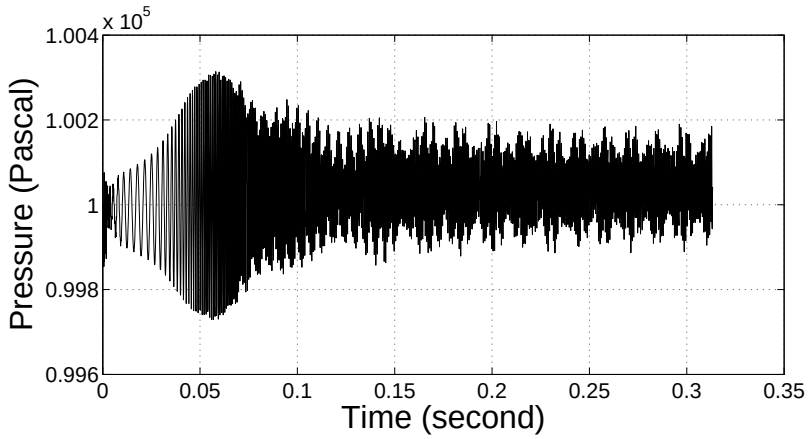
Figure 6.13: Probe locations

The pressure fluctuations at the probes are shown in figure 6.14. At the first probe, figure 6.14a, there is an obvious repeating fluctuation in the pressure as if there was a wave train passing through. The spin up phase is also clearly present up till $t = 0.1$ where the simulation stabilises as the transients[1] fade away and the structured fluctuation in the pressure is allowed to form.

[1]Transients is a short burst of energy caused by sudden change in events



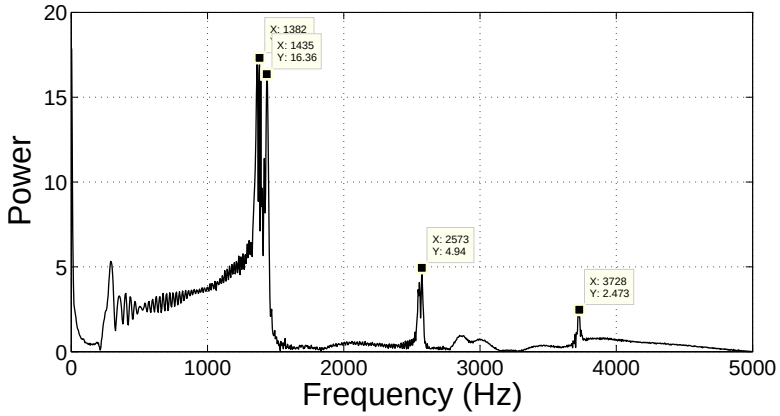
(a) Pressure over time at probe 1



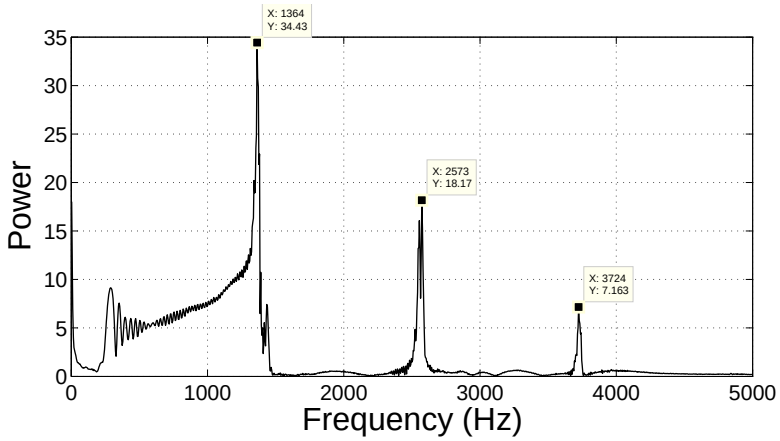
(b) Pressure over time at probe 2

Figure 6.14: Pressure over time at different probe locations.

The second probe contains no distinct fluctuating parts. This does not necessarily mean that there are none. In order to obtain the frequencies of the pressure fluctuations, the fast Fourier transform will be used. The resulting frequencies are shown in figure 6.15.



(a) FFT of the pressure at probe 1



(b) FFT of the pressure at probe 2

Figure 6.15: FFT of the pressure fluctuations at the probes

As we can see, the two probes measures the same frequencies which even seems to be higher harmonics of the first first frequency. The Strouhal number for this simulation is $St = 0.5456$ when using the most prominent frequency $f = 1364$ and velocity $v = 5$. This simulation appears to be a good basis for checking the flow over corrugation and predict the Strouhal number associated with the average flow velocity. Unfortunately this result came too late in order to run more simulations with different initial velocity conditions. The method of setting the initial condition is not optimal and needs to be improved further such that the initial behaviour can be avoided.

6.2.2 Axisymmetric Mesh

The axisymmetric mesh consist of a single corrugation which is a 4 degree wedge of a complete pipe, where the wedge was set up such that the axis of the pipe was aligned with the x-axis and the pipe body extruded in the y-plane. As this is a type of mesh better suited for pipe flow calculations, we should expect better results than with the pure two dimensional mesh. Unfortunately the axisymmetric mesh simulation was consistently unstable and seemed to have problems at the center of the pipe where the axis of rotation is. The source of this instability is unknown. One source of error could be the mesh, however great care was put in developing the proper mesh to the requirements that OpenFOAM needs in order to run an axisymmetric case. The output from OpenFOAMs own tool (*checkMesh*) for checking mesh quality reported no problems in regard to the wedge generated. However, as the problem seems to be closely connected to the axis of the pipe, it is reasonable to assume that the mesh is the problem. Due to my limited experience with

meshing, this problem has to be left unsolved at this time. The results obtained in the pure two dimensional simulation are very promising and should act as an incentive for further work with the axisymmetric mesh such that the results can be further improved.

7 Flow Through a Corrugated Pipe

Simulating only the flow over a corrugation is not enough to obtain practical data concerning the sound emitted by a corrugated pipe. The natural step forward is to simulate a physically accurate pipe such that key acoustic properties of the pipe can be predicted and possibly countered. In this section a corrugated pipe will be simulated and the problems arising from boundary condition and LES will be discussed.

7.1 Geometry

This case will simulate a pipe of 294 mm using the boundary conditions described in section 5.3. The inlet will have a constant velocity of 8 m/s in the x-direction. The inlet boundary posed a challenge and may be the source of errors in this simulation. From a qualitative test of different lengths of the pipe, it was concluded that the longer the pipe is, the easier it sings. The complete pipe consists of 98 corrugations, making it 294 mm long. In addition to the pipe, there is a square domain outside the end of the pipe which represents the atmosphere. This box also helps eliminating any problems which may occur, see 5.5, at the outlet boundary conditions as explained before, with the exception of the inlet. As the flow moves farther away from the outlet of the pipe, the less important is the accuracy of the solution as we have moved away from the domain of interest. This allows us to gradually increase the size of the cells towards the boundary condition in this box and subsequently reducing the total amount of cells which in turn reduces the computational cost. The inlet is more difficult to move away from the domain of interest as the geometry of the inlet may significantly influence the generation of sound. If the inlet is moved downstream and a flat pipe is used to transport the flow between the inlet and the corrugated pipe, I found from experiments that the sound generating mechanism disappears. Therefore for this simulation the inlet will be placed at the beginning of the corrugated pipe. See geometry in figure 7.1. The mesh will have the same resolution per corrugation as the single corrugation, which results in a mesh with $5.6 \cdot 10^6$ cells. Combined with the small time step needed, approximately $10^{-7}s$, this is a very time consuming calculation, as large eddy simulations usually are. In order be able to do analysis of this simulation the data produced by needs to be stored with an appropriate time interval. However, as the mesh consist of a very large amount of cells, the necessary storage requirements would go into hundreds of gigabytes if the sampling time would be allowed to be every $5 \cdot 10^{-5}s$. This high sampling rate is needed in order to resolve the frequencies produced by a singing corrugated pipe. To avoid this storage problem, nine probes were placed in areas of interest which captures the data at run time. This reduces the storage cost down to a couple of megabytes. These probes can be seen in figure 7.1 with the corresponding coordinates in table 7.1. All the data for each cell will be stored every $10^{-2}s$ such that visual inspection of the solution can be done and will therefore not use unmanageable amount storage space.

Probe	x-coordinate	y-coordinate
1	0.300072285030473	0.00483425133840653
2	0.293898677744827	0.00529938613390042
3	0.14453174757383	0.0055350234786035
4	0.145418467191904	0.00689454256071798
5	0.144461066095217	0.000523670586583546
6	0.151593132959457	0.00379370914884357
7	0.311242999020088	0.0140585531646709
8	0.438076555149122	0.0608618660095729 0
9	0.53595232101649	0.0493470700251767

Table 7.1: Probe coordinates

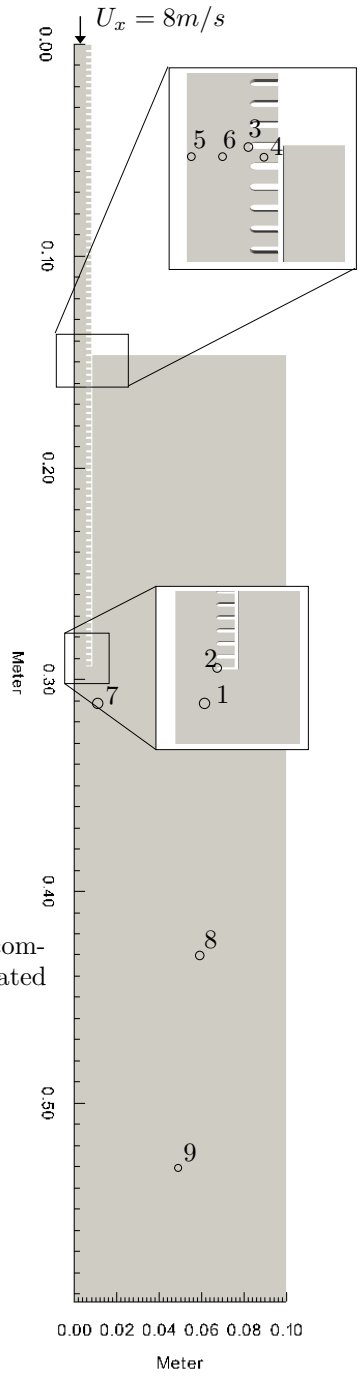


Figure 7.1: Geometry of the computational domain for the corrugated pipe with probe locations

7.2 Results

The simulation ran until $t = 0.04s$[2], which may be insufficient in order to obtain the standing wave when comparing to the run time from Popescu [15] at $t \approx 0.2s$. The required computational cost was far too great to accompany in the allotted time I had in my master's thesis. There are steps which can be done in order to reduce the computational cost, where the most effective would be reducing the mesh resolution in less important areas. There exists also a different approach to aero acoustics, which has not been discussed in this paper, called acoustic analogy. This is a hybrid approach to aero acoustics based on computing the flow field with a more cost efficient way, such as RANS, and then computing the acoustic source using an analogy such as Lighthill's analogy [11]. This hybrid approach is beyond the scope of this paper, but is an interesting approach to general aero acoustics. It can be seen in figure 7.2 that there are pressure waves emitted at the end of the pipe which travels in a radial direction.

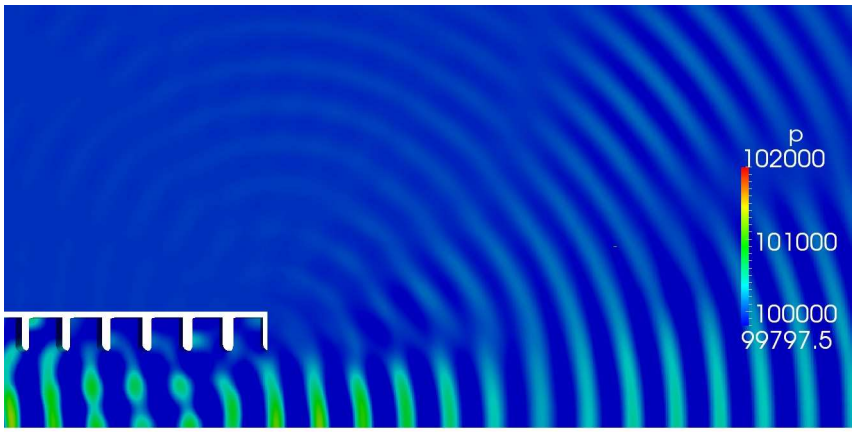
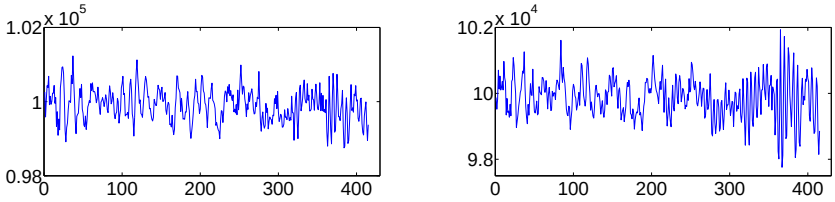


Figure 7.2: Pressure field at the outlet at time $t = 0.0438151$

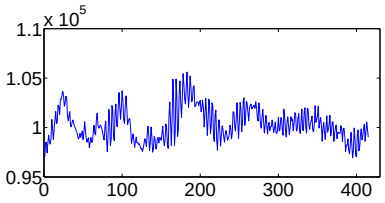
These waves represents a difference in pressure travelling through the air and is appropriately called sound waves as the propagation speed is the speed of sound. The frequency of these waves can be calculated by the relationship between velocity and wavelength; $v = \lambda f$. The velocity is the velocity of sound in air, $v = 343m/s$ and the wavelength is $\lambda = 0.003m$. This gives a frequency of $f = 114333Hz$ which is far beyond the frequency which humans can hear (approximately 20 kHz is the threshold for humans). This is far from what frequencies a corrugated pipe produces with such a modest velocity. In order to investigate further, the data produced by the nine probes needs to be analysed. The pressure data from the probes can be viewed in figure 7.2.

[2]The simulation ran for 4 weeks on 128 cores

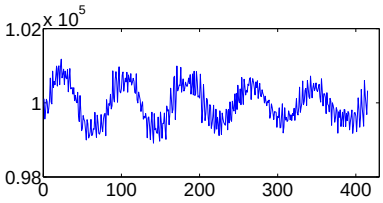


(a) Pressure at location 1

(b) Pressure at location 2

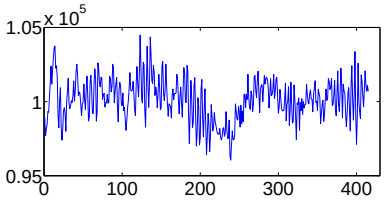


(c) Pressure at location 3

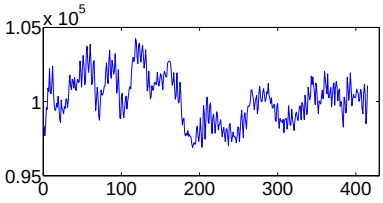


(d) Pressure at location 4

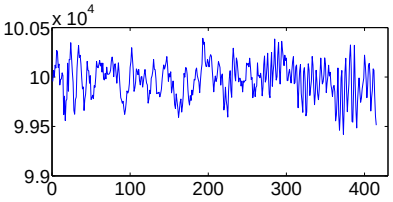
Figure 7.3: Pressure over time at different probe locations. (The axis units have been removed to reduce clutter. x-axis: Time (seconds), y-axis: Pressure (Pascal)).



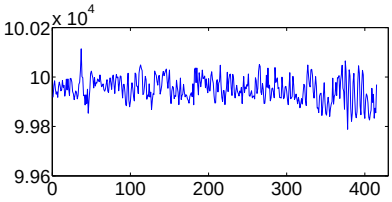
(e) Pressure at location 5



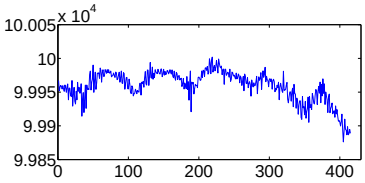
(f) Pressure at location 6



(g) Pressure at location 7



(h) Pressure at location 8



(i) Pressure at location 9

Figure 7.3: (Cont.) Pressure over time at different probe locations. (The axis units have been removed to reduce clutter. x-axis: Time (seconds), y-axis: Pressure (Pascal)).

From all the pressure plots from the probes, it is clear that probe 4 has a distinct sinusoidal form. This probe is located inside a corrugation and has the frequency $f = 235.7Hz$ which was found by a fast Fourier transform shown in figure 7.4. This frequency shows that there is a periodic event happening and is most likely connected to the appearing and disappearing of vortices inside the corrugations. More on this below. The probes 7 to 9 are located outside the pipe and one could expect them to resolve the high frequency which was found previously. This is, however,

not the case. The reason it does not show the frequency found to be $f = 114333Hz$ is because the sampling rate for the probes only allows frequencies up to $10000Hz$. This limitation comes from the Nyquist frequency, which allows us only to capture frequencies which is at most half the sampling frequency, therefore in this case the maximum frequency we are able to capture is $\frac{1}{2}\frac{1}{10^{-5}} = 10000Hz$.

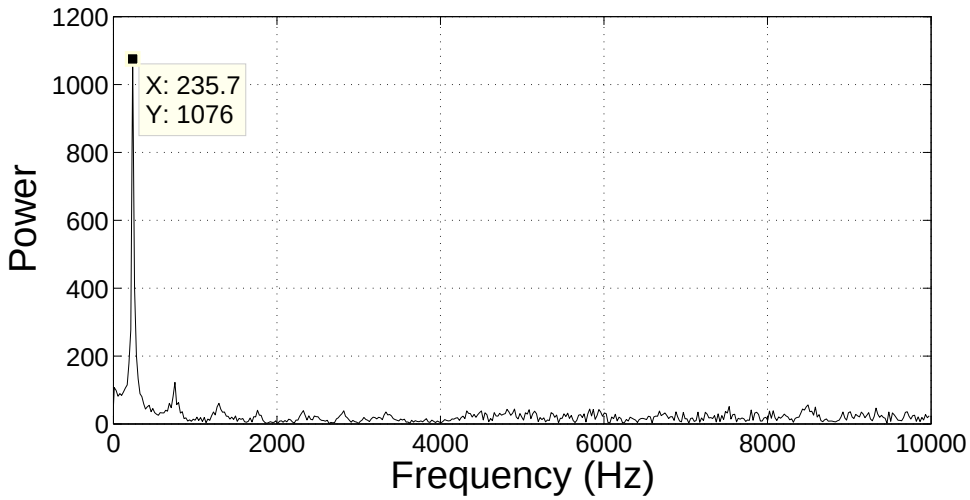


Figure 7.4: FFT of the pressure at probe 4

From the previous simulation, flow over corrugations, we can expect to find similar vortices in the corrugations of the pipe. The figure 7.5 shows the instantaneous velocity field of a couple of corrugations midway into the pipe at the time $t = 0.0438151$. As we can see, there are vortices present in the corrugations. Some vortices are more prominent than others, which may be a result of the periodic high velocity and low velocity parts. The mechanism which determines when a vortex is present or not relates to the standing wave occurring in a corrugated pipe. From [15] it is shown that in areas with a mean acoustic pressure there is little to no rotation in the corrugation, whereas in areas with an extreme value of the acoustic pressure is present the vortices are quite strong. This suggests that there may be something similar to a standing wave in figure 7.5 and may be attributed to the high frequency present in the system likely caused by the inlet, which is to be discussed next.

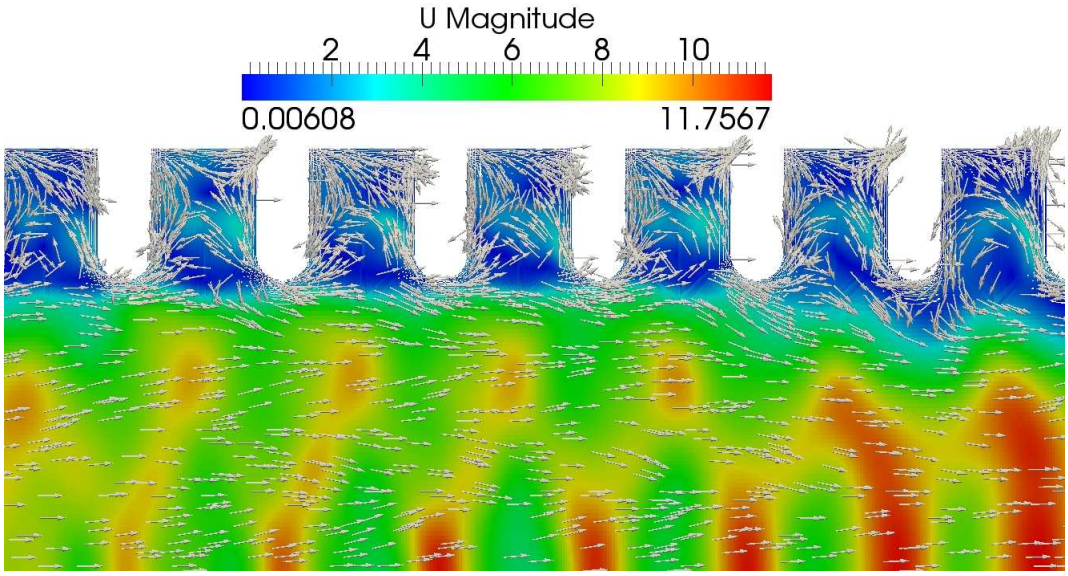


Figure 7.5: Velocity field snapshot taken at the center of the pipe at time $t = 0.0438151$

From visually inspecting the solution is was immediately clear that the inlet boundary condition was far from optimal and seems to be the source of the high frequency produced at the outlet. Figure 7.6 shows the inlet velocity field at time $t = 0.0438151$.

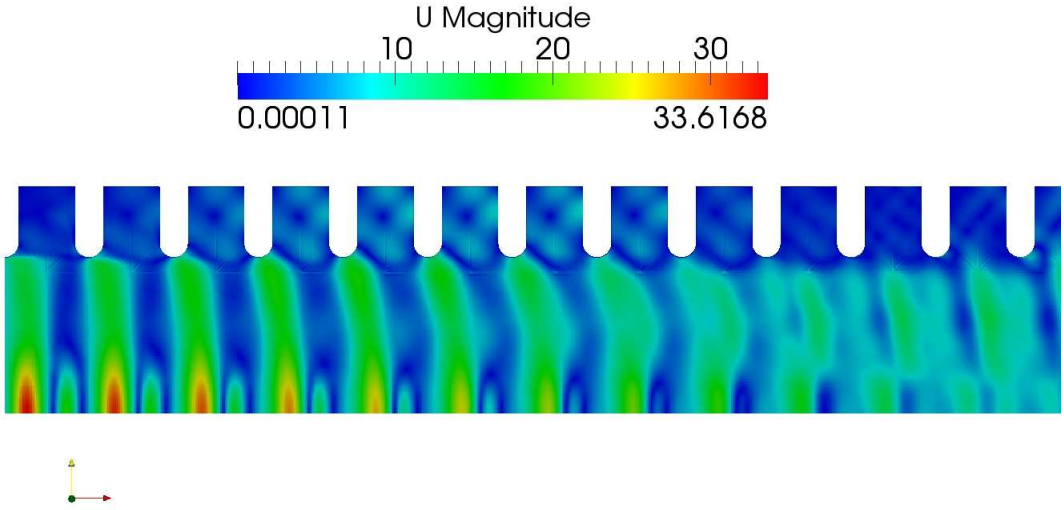


Figure 7.6: Velocity field at the inlet

The inlet is most likely causing problems downstream. As we can see from figure 7.6, the velocity magnitude is far greater than what would seem to be natural. The source of the high frequency found outside the pipe seems to come from errors produced by the inlet boundary condition. The boundary condition imposed on the inlet was a flow parallel to the axis of the pipe with no turbulent structures. Therefore, in order for the solver to cope with this uniform flow it produces packets of high velocity near the axis. These high velocity packets influences the solution down stream as the high frequencies produced by these packets may be kept alive by the corrugated pipe. The velocity of these packets do slow down to a more physical level further

down the pipe and can be viewed as transients. However, there is a constant production of these high velocity packets at the inlet which would produce transients through out the simulation. In addition to the flow, we have pressure waves going both upstream and downstream in the pipe, as this is the nature of sound waves. The boundary condition for pressure at the inlet does not allow the energy from these waves to exit the system and is consequently reflected back into the pipe. This allows energy to build up over time and may be a strong contributor to the behaviour which we observe at the inlet. In order to let the pressure waves to exit the domain through the inlet, we would need the boundary condition defined at the outlet. However, if one were to do this, the inlet would be over determined as the pressure and velocity is defined with a specific value. This is a bad configuration of the boundary conditions and would lead to a divergent solution. In order to avoid this problem, we need to define a non reflective boundary condition for the pressure at the inlet. As it currently stands, I do not know if there is such a boundary condition in OpenFOAM which could work for the inlet.

Unfortunately there was not enough time to attempt to correct the inlet boundary condition due to the lengthy simulation time. In order to produce a better inlet flow field one could define a fully developed flow at the boundary or use the boundary condition used in the previous case and recycle the flow a little bit downstream back to the inlet. This would ensure that the flow would develop a much better turbulent flow field at the inlet after some time. Another approach would be to use the converged solution of a steady state RANS simulation as the initial value. This could possibly remove transients from the solution and subsequently the standing wave could develop at a much earlier point in time.

8 Summary and Discussion

When a gas flows through a corrugated pipe at a high enough velocity there is a distinct "singing" which can be heard. This paper has explored this phenomenon using computational fluid dynamics, with code implemented in OpenFOAM. Throughout this thesis the necessary boundary conditions and initial values have been discussed, as well as mesh generation. When I began working on this thesis I had no prior experience with computational fluid dynamics, which is why a big part of this work was to develop the theory behind CFD code. The work flow for setting up the correct boundary conditions, initial values, solver settings and discretisation schemes is, as I experienced, an iterative process stretching over a long period of time. As I progressively became more experienced with both CFD and the workings of OpenFOAM, the time spent on each iteration would shorten, which in turn allowed me to perform even more iterations. The process was plagued with divergent simulations, bad meshes and insufficiently defined boundary conditions. However, in the end, most aspects of setting up a CFD case fell in place and some promising results were produced.

The two cases investigated was the flow over corrugations and the flow through a corrugated pipe. The first case was split into two types of mesh; symmetric and axisymmetric. The axisymmetric simulation had problems which I was unable to solve. The most likely reason for the simulation diverging is the the mesh. Creating a good mesh is somewhat of an art itself and with time and experience a working mesh could have been produced. The difference in the solution between a symmetric and axisymmetric mesh is something worth investigating in order to determine if the extra work needed to generate a good enough mesh for axisymmetric calculations is worth it. The symmetric mesh only had boundary condition problems, which was resolved by discarding the *directMappedFlowRate* boundary condition and using the *directMapped* boundary condition. The *directMappedFlowRate* exhibited a drift in the velocity, which resulted in a divergent solution. It was found that this boundary condition does not preserve the complete velocity field when mapping it back to the inlet. Only the velocity component normal to the boundary is preserved. This makes it unusable for simulating an infinite pipe as there is information lost through the boundary. The second boundary condition did not suffer from this limitation and proved to be a stable boundary condition. Simulation using the *directMapped* boundary condition resulted in a Strouhal number which was in agreement with previous work. There were unfortunately not enough time to compute the Strouhal number for different velocities, which would have been the next natural step for this case.

The second case featured a corrugated pipe of finite length, which was given a uniform inlet velocity. This simulation was very computational expensive due to the high resolution mesh needed. The computational time could have been reduced by decreasing the amount of cells by removing superfluous parts of the mesh. However, most of the cells are located in the corrugated pipe and can not be altered too much in order to keep the mesh resolution needed by a large eddy simulation. The results from the simulation yielded sound frequencies far greater than expected, which was traced back to a problematic inlet boundary condition. Ideally the inlet should have been moved away from the domain of computational interest to reduce the transients continuously produced at the inlet. However, as the corrugated pipe is sensitive to flat sections of pipe before or after the corrugations, the inlet was positioned at the start of the corrugated pipe. As a result, the simulation was unsuccessful in obtaining the standing wave present in a corrugated pipe. The inlet boundary needs to be improved before any further analysis can be done. If I had the time to work more on this project, I would have begun to investigate different approaches to the inlet problem. There are three solutions I have been considering following the result from the simulation. I could prescribe a fully developed flow at the inlet boundary condition such that the flow hitting the corrugation is as physically valid as possible. The boundary condition used in the first case could also be used to recycle the flow field down stream back to the inlet. This would allow the flow to develop realistic turbulence at the inlet over time. Another way, perhaps in combination with the other suggestions, is to calculate the steady state solution using RANS as a initial value for

the whole domain. This would allow turbulent areas to start with an approximate turbulence and reduce the time spent on computing the spin up phase which is prone to transients and may be uninteresting.

The phenomenon of singing corrugated pipes is an interesting subject and should be investigated further, as the practical implications of being able to predict and reduce the sound generated is of high value.

A Appendix

All boundary conditions

Listing 1: Boundary file for U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);

boundaryField
{
    wall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    inlet
    {
        type            fixedValue;
        value           uniform (8 0 0);
    }
    outlet
    {
        type                inletOutlet;
        inletValue          uniform (0 0 0);
        value               uniform (0 0 0);
    }
    axis
    {
        type            symmetryPlane;
    }
    defaultFaces
    {
        type            empty;
    }
}
```

Listing 2: Boundary file for p

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 100000;

boundaryField
{
    axis
    {
        type            symmetryPlane;
    }
    wall
    {
        type            zeroGradient;
    }
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            waveTransmissive;
        gamma           1.3;
        field           p;
        phi             phi;
        rho             rho;
        psi             psi;
        fieldInf        100000;
        lInf            1;
        value           uniform 1e5;
    }
    defaultFaces
    {
        type            empty;
    }
}


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

Listing 3: Boundary file for T

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 300;

boundaryField
{
    axis
    {
        type            symmetryPlane;
    }
    wall
    {
        type            fixedValue;
        value           uniform 300;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 300;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 300;
        value           uniform 300;
    }
    defaultFaces
    {
        type            empty;
    }
}


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

Listing 4: Boundary file for alphaSgs

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alphaSgs;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    axis
    {
        type            symmetryPlane;
    }
    wall
    {
        type            zeroGradient;
    }
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    defaultFaces
    {
        type            empty;
    }
}


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

Listing 5: Boundary file for muSgs

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      muSgs;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    axis
    {
        type            symmetryPlane;
    }
    wall
    {
        type            zeroGradient;
    }
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    defaultFaces
    {
        type            empty;
    }
}


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

Modified boundary conditions for Case 1

Listing 6: Boundary file for U using *directMappedFlowRate*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                      |                                            |
| \\      /  F ield              | OpenFOAM: The Open Source CFD Toolbox      |
|  \\    /   O peration          | Version:  2.0.1                            |
|   \\  /    A nd                | Web:      www.OpenFOAM.com                 |
|    \\/     M anipulation       |                                            |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);

boundaryField
{
    wall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    inlet
    {
        type            directMappedFlowRate;
        phi             phi:
        rho             rho;
        neighPhi        neighPhi;
        value           uniform (0 0 0); // placeholder
    }
    outlet
    {
        type                inletOutlet;
        inletValue          uniform (0 0 0);
        value               uniform (0 0 0);
    }
    axis
    {
        type            symmetryPlane;
    }
    defaultFaces
    {
        type            empty;
    }
}
```

Listing 7: Boundary file for U using *directMapped*

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                      |                                            |
| \\      /  F ield              | OpenFOAM: The Open Source CFD Toolbox      |
|  \\    /   O peration          | Version:  2.0.1                            |
|   \\  /    A nd                | Web:      www.OpenFOAM.com                 |
|    \\/     M anipulation       |                                            |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);

boundaryField
{
    wall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    inlet
    {
        type            directMapped;
        value           uniform (8 0 0);
        setAverage      false;
        average         (0 0 0);

    }
    outlet
    {
        type                inletOutlet;
        inletValue          uniform (0 0 0);
        value               uniform (0 0 0);
    }
    axis
    {
        type            symmetryPlane;
    }
    defaultFaces
    {
        type            empty;
    }
}
```

Solver settings

Listing 8: Discretisation schemes

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
ddtSchemes
{
    default          backward;
}

gradSchemes
{
    default          Gauss linear;
    grad(p)          Gauss linear;
}

divSchemes
{
    default          none;
    div(phi,U)       Gauss filteredLinear2V 1 0;
    div(phiU,p)      Gauss linear;
    div(phi,h)        Gauss linear;
    div(phi,K)       Gauss linear;
    div((muEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default          none;
    laplacian(muEff,U)                  Gauss linear corrected;
    laplacian(mut,U)                    Gauss linear corrected;
    laplacian(DkEff,k)                  Gauss linear corrected;
    laplacian(DepsilonEff,epsilon)      Gauss linear corrected;
    laplacian(DREff,R)                  Gauss linear corrected;
    laplacian(DomegaEff,omega)          Gauss linear corrected;
    laplacian((rho*(1|A(U))),p)         Gauss linear corrected;
    laplacian(alphaEff,h)               Gauss linear corrected;
}

interpolationSchemes
{
    default          linear;
}

snGradSchemes
{
    default          corrected;
}

fluxRequired
{
    default          no;
    p                ;
}
// ************************************************************************* //
```

Listing 9: Solution settings

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
solvers
{
    "(p|rho)"
    {
        solver          PCG;
        tolerance       1e-06;
        relTol          0.01;
        preconditioner  DIC;
    }
    "(p|rho)Final"
    {
        $p;
        relTol          0;
    }
    "(U|h|k|epsilon|omega)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-6;
        relTol          0.1;
    }
    "(U|h|k|epsilon|omega)Final"
    {
        $U;
        tolerance       1e-06;
        relTol          0;
    }
}

PIMPLE
{
    momentumPredictor yes;
    nOuterCorrectors 15;
    nCorrectors     4;
    nNonOrthogonalCorrectors 1;
    rhoMin          rhoMin [ 1 -3 0 0 0 ] 0.5;
    rhoMax          rhoMax [ 1 -3 0 0 0 ] 2.0;

    residualControl
    {
        "(U|k|epsilon|p)"
        {
           relTol          0;
           tolerance       1e-05;
        }
   }
}
relaxationFactors
{
    "(p|rho|U|T).*" 1;
    "(h|k|epsilon|omega).*" 1;
}
```

References

[1] Salomè. http://www.salome-platform.org/.

[2] K. Avila, D. Moxey, A. de Lozar, M. Avila, D. Barkley, and B. Hof. The Onset of Turbulence in Pipe Flow. *Science*, 333(6039):192–196, July 2011.

[3] L.H. Cadwell. Singing corrugated pipes revisited. *American Journal of Physics*, 62:224–227, March 1994.

[4] F.S. Crawford. Singing Corrugated Pipes. *American Journal of Physics*, 42:278–288, April 1974.

[5] H.K. Dahle. *Lecture Notes in Continuum Mechanics*. University of Bergen Department of Mathematics, 2010.

[6] J.H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer, 2002.

[7] T. Hayase, J.A.C. Humphrey, and R. Greif. A consistently formulated QUICK scheme for fast and stable convergence using finite-volume iterative calculation procedures. *J. Comput. Phys.*, 98(1):108–118, 1992.

[8] V. Kop'ev, M. Mironov, and V. Solntseva. Aeroacoustic interaction in a corrugated duct. *Acoustical Physics*, 54:197–203, 2008.

[9] U.R. Kristiansen and G.A. Wiik. Experiments on sound generation in corrugated pipes with flow. *The Journal of the Acoustical Society of America*, 121(3):1337–1344, 2007.

[10] P.K. Kundu and I.M. Cohen. *Fluid Mechanics*. Academic Press. Elsevier, 2010.

[11] M.J. Lighthill. On sound generated aerodynamically. i. general theory. *Proceedings of the Royal Society A Mathematical Physical and Engineering Sciences*, 211(1107):564–587, 1952.

[12] G. Nakiboğlu, S.P.C. Belfroid, J.F.H. Willems, and A. Hirschberg. Whistling behavior of periodic systems: Corrugated pipes and multiple side branch system. *International Journal of Mechanical Sciences*, 52(11):1458 – 1470, 2010. Special Issue on Non-linear Oscillations.

[13] OpenCFD. *OpenFOAM - The Open Source CFD Toolbox - Documentation*. OpenCFD Ltd., 2.0 edition, 2011.

[14] OpenCFD. *OpenFOAM - The Open Source CFD Toolbox - User's Guide*. OpenCFD Ltd., 2.0 edition, 2011.

[15] M. Popescu, Stein T. J., and Wei S. Flow-induced acoustics in corrugated pipes. *Commun. Comput. Phys.*, 10(1):120–139, 2011.

[16] M. Rudman and H.M. Blackburn. Large eddy simulation of turbulent pipe flow. *Second International Conference on CFD in the Minerals and Process Industries*, 1999.

[17] M.P Silverman and G.M Cushman. Voice of the dragon: the rotating corrugated resonator. *European Journal of Physics*, 10(4):298, 1989.

[18] P.P. Sullivan, J.C. McWilliams, and C. Moeng. A subgrid-scale model for large-eddy simulation of planetary boundary-layer flows. *Boundary-Layer Meteorology*, 71:247–276, 1994.

[19] D. Tonon, B.J.T. Landry, S.P.C. Belfroid, J.F.H. Willems, G.C.J. Hofmans, and A. Hirschberg. Whistling of a pipe system with multiple side branches: Comparison with corrugated pipes. *Journal of Sound and Vibration*, (329):1007–1024, 2010.

[20] J. Tu, G.H. Yeoh, and C. Liu. *Computational Fluid Dynamics: A Practical Approach*. Butterworth Heinemann. Butterworth-Heinemann, 2008.

[21] H.K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education Limited, 2007.