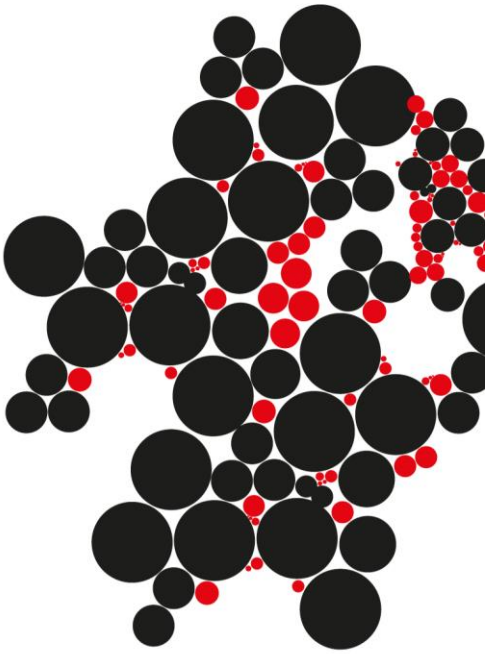
MASTER THESIS



TIME-DOMAIN IMPEDANCE BOUNDARY CONDITIONS IN COMPUTATIONAL FLUID DYNAMICS FOR USE IN THERMOACOUSTIC MODELING

Bart van der Poel

FACULTY OF ENGINEERING TECHNOLOGY
LABORATORY OF THERMAL ENGINEERING

EXAMINATION COMMITTEE
Prof.dr.ir. T.H. van der Meer (chair)
Ir. J.P. Oosterhuis (member)
Dipl.-Ing. S. Bühler (member)
Dr. A.R. Thornton (external member)

UNIVERSITY OF TWENTE.

AUGUSTUS 2013

Time-Domain Impedance Boundary Conditions in Computational Fluid Dynamics for use in Thermoacoustic Modeling

Bart van der Poel - University of Twente

August, 2013

Abstract

Thermoacoustic devices convert heat to work or vice versa using acoustic waves, and consist of several parts that partially reflect acoustic waves. These reflections can be represented by an acoustic impedance, which is a complex quantity that takes into account a phase difference between the pressure and velocity oscillation. Usually one is only interested in a single part of the thermoacoustic device, but to correctly numerically predict these wave phenomena the whole device should be modeled. As solving acoustics using Computational Fluid Dynamics (CFD) requires comparatively high computational resources, using the complete computational domain of the whole device is unattractive. The reduction of the computational domain might be possible with a so-called Time-Domain Impedance Boundary Condition (TD-IBC), which relates the inward traveling acoustic wave to the outward traveling acoustic wave. Standard commercial CFD software does not feature such a boundary condition as of yet. This master thesis discusses the feasibility, theory and implementation of both a non-reflective boundary condition (NRBC) and a TD-IBC. The NRBC and TD-IBC are implemented in *ANSYS CFX* CFD software using *FORTRAN* subroutines. The results showed that for a complex reflection coefficient of $R = 0.5 + 0.5i$, and an excitation frequency of $f = 100Hz$, the measured absolute reflection coefficient $|R|$ was off 1.05%, and the phase differed $\phi = 0.0334rad$. From the work done it can be concluded that it is indeed possible to implement a TD-IBC for application in thermoacoustics.

Preface

The start of this master thesis was characterized by uncertainty, as the exact topic was not known yet. What was known, was that it would be a numerical assignment in the field of thermoacoustics. After some preliminary research, and input from Joris and Simon, the topic of the assignment became clear. Joris and Simon guided me throughout the whole thesis, and despite often being busy, have always found time to discuss matters. Later in the project, a weekly progress meeting with everybody related to the thermoacoustics project started. This meeting provided me with invaluable feedback.

As Joris and Simon were the most prominent contributors to the succes of this thesis, I would like to thank them first. I also would like to thank Theo for helping me choose a master thesis assignment, assigning me to the project, and guiding me throughout the project.

A special note for my colleagues in room N248. Although the ubiquitous presence of my fellow graduation students in room N248 have impaired my work efficiency from time to time, their unique characters have supplied me with plentiful laughter that prevented me from falling into utter despair at numerous stages of my research. During my stay we became accustomed to various avocations, such as “klaverjassen” at 12 o’clock, and “blik-time” at 3 o’clock in the afternoon. While the rudimentary knowledge of some “klaverjassen” players jeopardized the efficacy of the teams during play, their assiduous efforts and lavish endeavors were often eulogized. Thus I would like to thank Daan, Johan, Frans, Sjoerd, Gerald, Maarten, and Sam for creating a collegial and comfortable atmosphere.

B.J. van der Poel (s0217972)
University of Twente
Faculty of Engineering Technology
Laboratory of Thermal Engineering
Sustainable Energy Technology

The exam committee is comprised of the following members:

Prof.dr.ir. T.H. van der Meer (Chair)
Ir. J.P. Oosterhuis (Member)
Dipl.-Ing. S. Bühler (Member)
Dr. A.R. Thornton (External Member)

Contents

Chapter 1

Introduction

The twentieth century has seen unprecedented growth in the fields of science, technology, and medicine, but also huge advances in civil and political rights. These fundamental and far reaching changes throughout society, more often seen as beneficial than not, increased life span, well-being and prosperity for people from all walks of life. But the vastly increased quality of life over the last centuries does come at a price. The dependence on finite reserves of fossil fuels and the increased pollution are two of the most prominent caveats of these changes. To mitigate these negative side effects, one can look at possible (partial) solutions from three angles. In order of priority: 1. To increase efficiency on the demand side (i.e. use less). 2. To use more sustainable energy sources. And 3. to make more efficient use of fossil fuels. This strategy has been formulated by E. Lysen [12] and is called Trias Energetica.

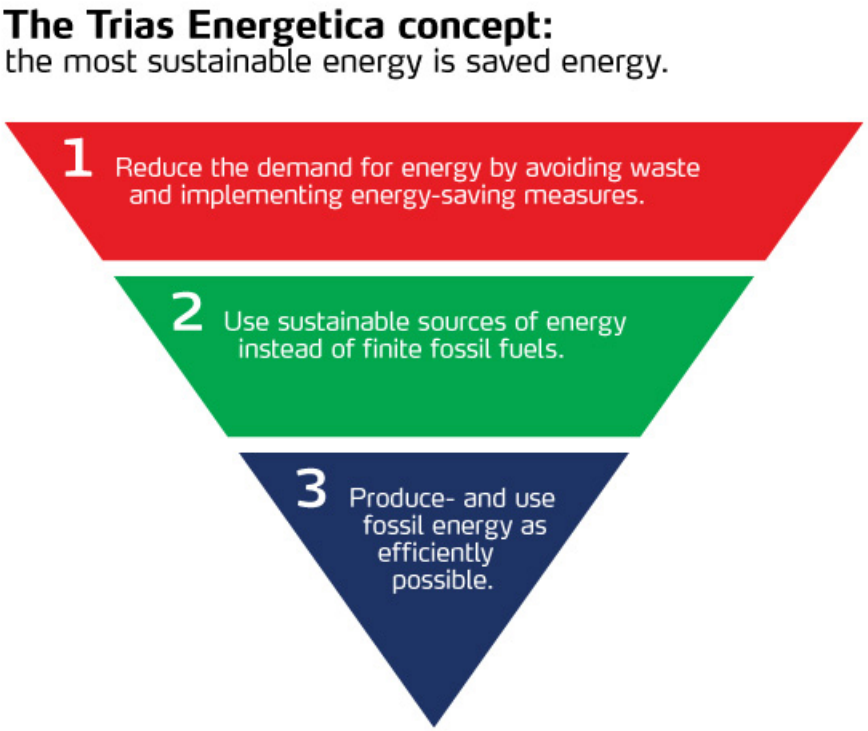


Figure 1.1: Trias Energetica

This master thesis is based on the premise that thermoacoustic devices can function as a potential solution to some of the aforementioned problems society faces today. A thermoacoustic engine is a type of thermoacoustic device that uses sound waves as a means to transport heat

(heat pump) or convert heat into mechanical work (prime mover). The mechanical work done in a prime mover could be used to drive electrical generators and produce electricity. These type of engines could possibly fulfill roles in both point 2 and 3 of the Trias Energetica strategy. Point 2 by using a sustainable heat sources, and point 3, for example, by using part of the gas to produce work or electricity while the waste heat is used to heat a building.

While the thermoacoustic engine and principle has been around for a while, the recently increased awareness on the topic of sustainability has sparked new interest in the technology which had previously been practically dormant for years. The technology features some promising characteristics in the light of sustainability. Firstly, thermoacoustic devices can operate with a small temperature difference between heat source and sink. This opens possibilities for application with sustainable heat sources, as these are generally of relatively low temperature compared to fossil fuel combustion. Secondly, the devices usually have minimal or no moving parts, depending on application. Which can be beneficial to production and maintanance costs, as well as reliability (especially in rough conditions).

The technology has not enjoyed the amount of research that the piston and turbine engines had. In this respect it lags behind considerably, and much experimental, analytical, as well as numerical research needs to be done to bring the technology on par. In this master thesis focus lays on the numerical modeling aspect. To create a physically realistic numerical model of a thermoacoustic engine based on a fluid dynamics code, often the complete domain must be calculated, even if only the performance and flow field of a particular section needs to be determined. Special boundary conditions called "Time-Domain Impedance Boundary Conditions" (TD-IBC) can potentially solve these problems. These boundary conditions are, however, not provided by commercial CFD codes. A literature research is done on these special boundary conditions, and subsequently are implemented using Fortran subroutines in a commercial CFD package called Ansys CFX, and their performance is tested.

1.1 Thermoacoustics

The first observations and descriptions about the thermoacoustic effect are already over two centuries old. In 1777, B. Higgins showed how to produce acoustic oscillations excited purely by a single heat source. His device was simple, just an open glass tube with a hydrogen flame at a suitable position (quarter length) inside, and was called *Higgins' singing flame*. A similar experiment was devised by P.L. Rijke, but now the flame was replaced by a heated gauze (see Fig. 1.2). Rijke also observed that the convective air current was imperative for the acoustic oscillations to occur [7].

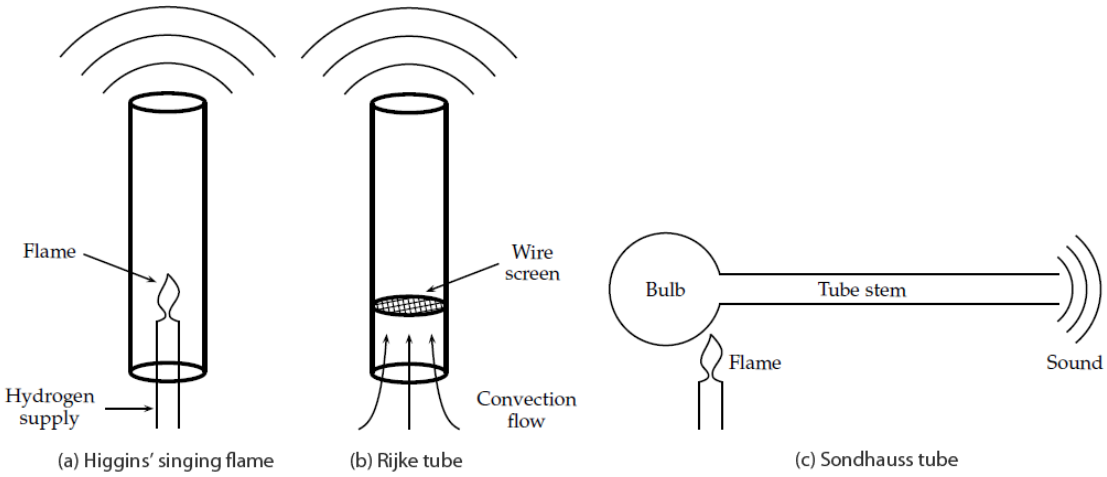


Figure 1.2: a) Higgins', b) Rijke, and c) Sondhauss tubes [7]

Another device is the Sondhauss tube, which has a closed end with a bulb attached to it (see Fig. 1.2). The bulb side was heated and did not need convective air current to operate. Nowadays, Higgins' singing flame and the Rijke tube can be characterized as a quarter wave resonator, while the Sondhauss tube is a half wave resonator. The thermoacoustic effect does not only occur under heating, but also under cooling. Taconis observed, when cooling a gas-filled tube from room temperature to cryogenic temperature, spontaneous oscillations of the gas under extreme cooling. The first, and correct, qualitative explanation about the thermoacoustic effect has been done by Lord Rayleigh in The Theory of Sound [20], saying:

> "If heat be given to the air at the moment of greatest condensation (compression) or taken from it at the moment of greatest rarefaction (expansion), the vibration is encouraged."

Thermoacoustics, as the name implies, is a combination of the fields of thermodynamics and acoustics. Acoustic sound waves are a combination of pressure and particle velocity waves. Temperature variations are usually very small for sound mostly experienced by people, and therefore often neglected in acoustics. In thermoacoustics however, it is essential in describing the thermoacoustic effect.

Thermoacoustic effects are often explained by using the notion of fluid parcels. Fluid parcels can be seen as the smallest volume possible in the medium for which the continuum assumption is still valid. The continuum assumption is the assumption that the behavior of the materials can be modeled as a continuous mass rather than as discrete particles (e.g. molecules), and is the basis of many equations in fluid dynamics (e.g. Navier-Stokes equations).

In free space, gas parcels exposed to an acoustic disturbance behave in a such a way that they predominantly expand and compress adiabatically. The temperature variations as a function of pressure in an adiabatic process can be described by equation 1.1. Where γ is the adiabatic index (1.4 for air), T the temperature, and P the pressure. This means that when the pressure increases, so does the temperature, and vice versa. (under the assumption that $\gamma > 1$, which it always the case in an adiabatic process).

$$T^{\gamma} = Constant \cdot P^{\gamma - 1} \tag{1.1}$$

When gas parcels are allowed to interact thermally with a solid boundary, two types of thermoacoustic effects may occur. Energy (heat) can be supplied to the acoustic wave, which increases acoustic power (called *prime mover*), or energy (heat) can be extracted and pumped, which decreases acoustic power (called *heat pump*). These parts can be combined with an electroacoustic transducer to supply (e.g. loudspeaker) or extract (e.g. microphone) acoustic energy from and to a mechanical/electrical source. Note that it does not necessarily need to be a loudspeaker or microphone to supply or extract acoustic energy, for example, a bi-directional impulse turbine could also be used to extract acoustic energy. Different useful combinations of parts are summed up in Table 1.1. In Fig. 1.3 an example of a heat driven refrigerator is shown. For example, to use industrial waste heat to cool another medium.

Table 1.1: Different thermoacoustic applications (non-exhaustive list)

#	Acoustic source	Acoustic sink	Description
1	Prime mover	Heat pump	Heat driven refrigerator
2	Prime mover	Microphone	Heat conversion to work/electricity
3	Prime mover	Environment	Heat induced loudspeaker
4	Loudspeaker	Heat pump	Work/electricity driven refrigerator

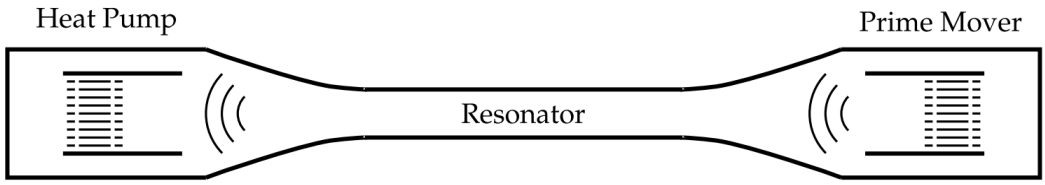


Figure 1.3: Thermoacoustic device that functions as a refrigerator [7]

The acoustic pressure and particle velocity do not necessarily be in phase. When the acoustic wave is a purely traveling wave, the phase angle between the pressure and velocity is zero (see Fig. 1.4b), while a purely standing wave has a phase angle of 90° (see Fig. 1.4a). A standing wave is caused by two traveling waves propagating in opposite directions, and has *nodes* at certain positions. Nodes are positions in space where the acoustic pressure or velocity is constantly zero, and are called *pressure nodes* and *velocity nodes* respectively. Positions where the acoustic pressure or particle velocity periodically reaches the maximum amplitude are called *anti-nodes*. Often, acoustic waves are not purely the one or the other, but a combination of both.

At a pressure anti-node the pressure variations, and thus the temperature variations, are largest, while at a pressure node there is no temperature variation. At a velocity anti-node the displacement is large, while at a velocity node there is no variation. For a standing wave device this means that at a velocity node the temperature variations are largest, while at a velocity anti-node there is no temperature variation. Between the nodes both the temperature and the velocity oscillates (and thus displaces the parcel), which results in a temperature gradient (visualized in Fig. 1.4c), which is known as the critical temperature gradient.

For a traveling wave the situation is somewhat different because the nodes coincide and move with the speed of sound. At the moment of greatest compression (hottest), the fluid parcels moves to the right, it then stops and starts moving to the left while expanding (cooling). This gives circle like temperature-position cycles (see Fig. 1.4d).

In practice, a device will be designed either as a standing or a traveling wave device, but both effects will occur. This will result in oval shaped temperature-positions paths (see Fig. 1.4e).

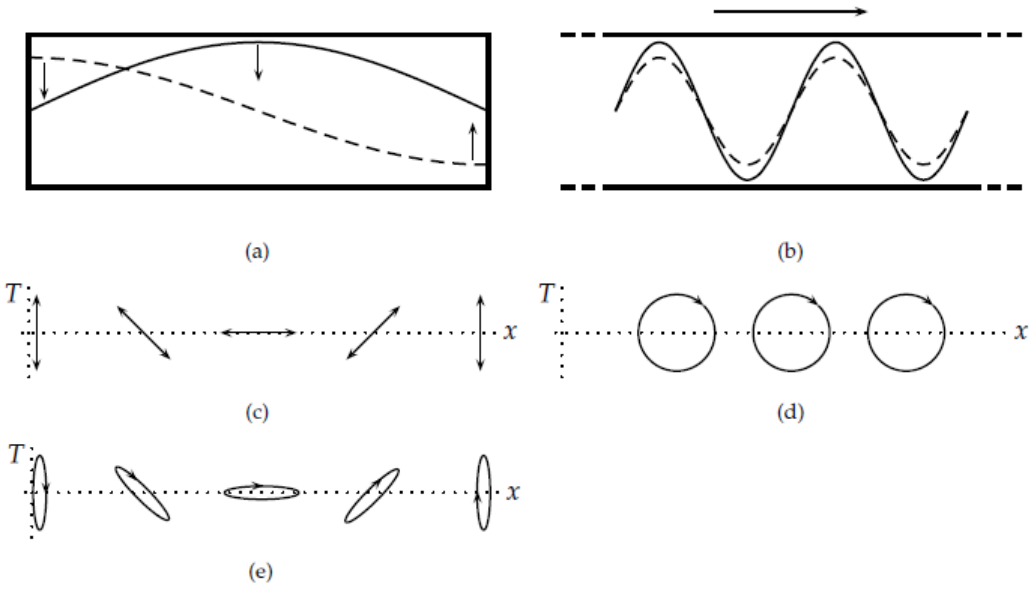


Figure 1.4: Standing and traveling wave temperature position cycles [7]

The *heat pump* and *prime mover* parts of a thermoacoustic device are in basic structure very similar, and often comprised of two heat exchangers with a porous structure between them. One heat exchanger is cold, the other hot. If the part functions either as a heat pump or prime mover depends on the actual temperature gradient achieved between the hot and cold heat exchangers, and the critical temperature gradient. If the temperature gradient at the position of the porous structure is smaller than the critical temperature gradient, the part will function as a heat pump (see Fig. 1.5a). If the temperature gradient is larger, than the part will function as a prime mover (see Fig. 1.5b). A heat pump will cause heat to be "pumped" from the cold to the hot heat exchanger, while a prime mover will cause the heat to flow from the hot to the cold heat exchanger.

The porous structure in between the heat exchangers serves two main purposes. For both traveling and standing wave devices the porous structure provides much more surface area for heat transfer to take place, and thus increasing the amount of power that can be transferred. The other porpuse has everything to do with maximizing the encircled area within the P-v diagram, which will be explained later in this section.

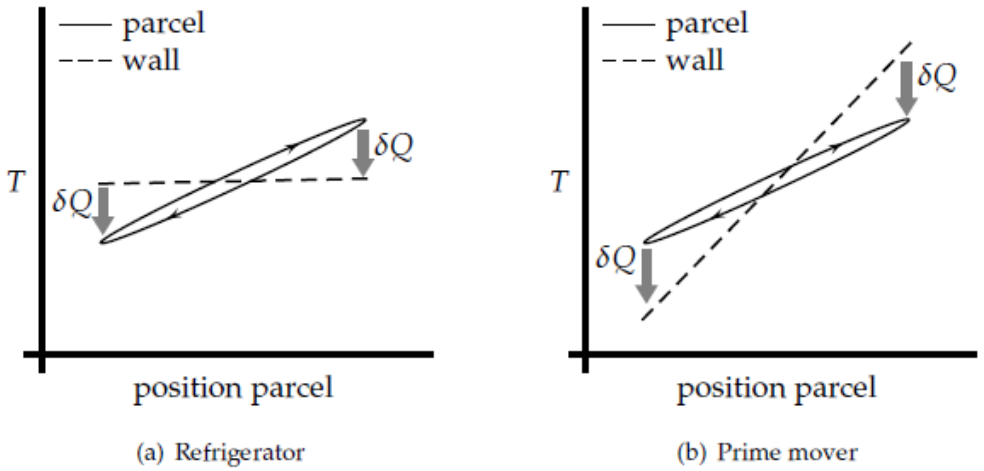


Figure 1.5: Refrigerator and prime mover temperature gradients [7]

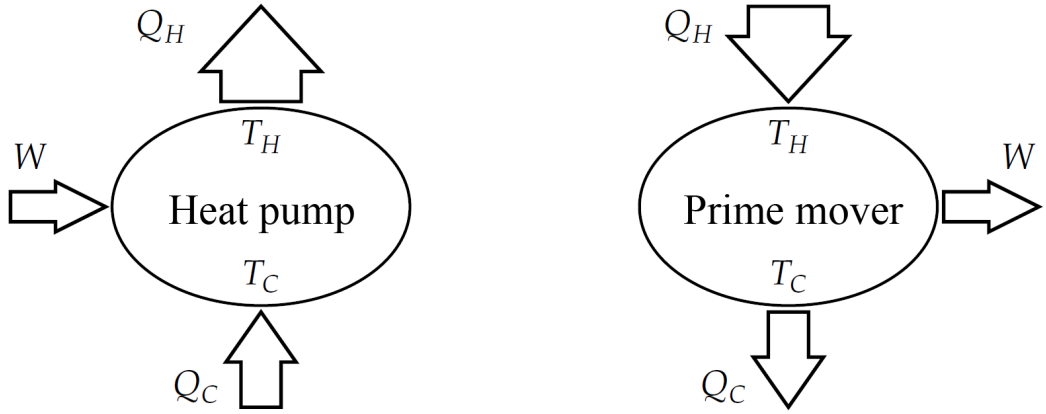
The physical principle of a heat pump and a prime mover can be explained from a thermodynamical perspective by taking into account the first and second law of thermodynamics. The first law states that energy is always conserved, which is mathematically described by the sum of the incoming and outgoing energy flows being equal equal to the change in internal energy (see eq. 1.2). Where ΔU is the change in internal energy, Q the heat added to the system, and W the work done by the system. If steady operation is considered, the change in internal energy ΔU is zero. The second law states that the entropy of an isolated system never decreases. In practive this implies that for a hot and a cold heat exchanger, without any work done, the heat will always flow from the hot to the cold heat exchanger.

$$\Delta U = Q - W \tag{1.2}$$

An overview of the resulting energy flows for a heat pump and prime mover are shown in Figure 1.6. From which it can be seen that (acoustic) work is needed to extract heat from a cold reservoir and move it to the hot reservoir (heat pump function). The other way around, (acoustic) work can be generated when heat can flow from the hot reservoir to the cold reservoir (prime mover function).

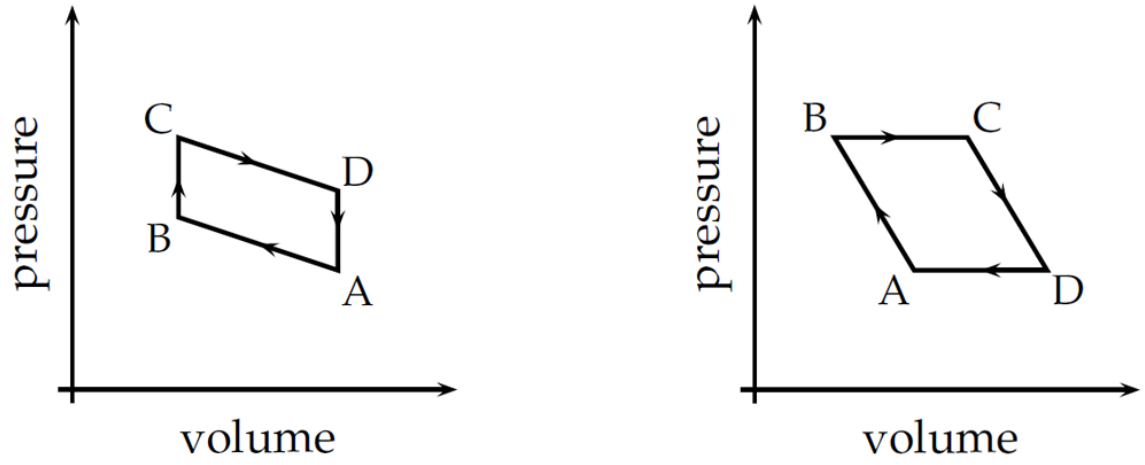
A more detailed explanation of the thermoacoustic effect, and the conversion between acoustic energy and heat in both a standing-wave and traveling wave device, can be given by looking at the thermodynamic cycle the process cycles through. A thermodynamic cycle is a collection of thermodynamic processes of both heat and work, which always returns to its initial state. A pressure-volume diagram (see Fig. 1.7) is a useful way to optimize the efficiency and operation of the cycle. Because the process is a cycle, the process always encircles an area within this diagram by definition. This surface area (within A-B-C-D in Figure 1.7) is equal to the power absorbed or produced. This power, in respect to the heat supplied or extracted, is a figure of merit for efficiency.

For a traveling wave device, the porous structure in between the heat exchangers needs to have good thermal contact (heat is exchanged quickly) for the area (and thus power) of the thermodynamic cycle in the P-v diagram to be maximized. The thermodynamic cycle of the traveling-wave device now resembles a so-called *Stirling cycle* (see Fig. 1.7a), and the porous structure is now called a *regenerator*. A standing wave device needs a time-lag for it to maximize the surface area in the P-v diagram. The time-lag can be accomplished by an imperfect



(a) Heat pump (b) Prime mover

Figure 1.6: Heat pump and prime mover energy flows [7]



(a) Traveling-wave device (b) Standing-wave device

Figure 1.7: Standing and traveling wave thermodynamic cycles in P-v diagram [7]

(slow heat transfer) thermal contact. The porous structure in a standing wave device is called a *stack*, and it resembles a *Brayton* thermodynamic cyle (see Fig. 1.7b).

The difference between the *Stirling* and *Brayton* cycles is the manner in which the path between states is traversed. A state can be seen as a point within the thermodynamic cycle for which the state variables are known (for example: pressure and temperature), and between different states the type of thermodynamic process varies. A gas turbine or a jet engine are examples of Brayton cycles. A stirling engine is an example of a Stirling cycle. Without going into too much detail about these cycles, the Brayton cycle in the standing-wave device is theoretically considered the least efficient of the two, because of the imperfect heat transfer accompanying the time-lag that increases entropy. An increase in entropy results in a new state that is "less able" to do work compared to a reference state.

The stack and regenerator also disturb the flow in other ways. The increased surface area causes extra viscous losses. Also, because of the increased viscous effects, part of the acoustic waves are reflected. These reflections are usually unwanted in a traveling wave device, as the

reflections will cause standing waves to form.

To give a more intuitive explanation, the thermoacoustic effect can be thought of as fluid parcels that move energy over a small distance each, but together over a large distance. A bit like firefighters did long ago, standing in a long line passing the buckets of water over from one person to another. From this analogy comes the term "Bucket-brigade effect", which is graphically represented in Figure 1.8.

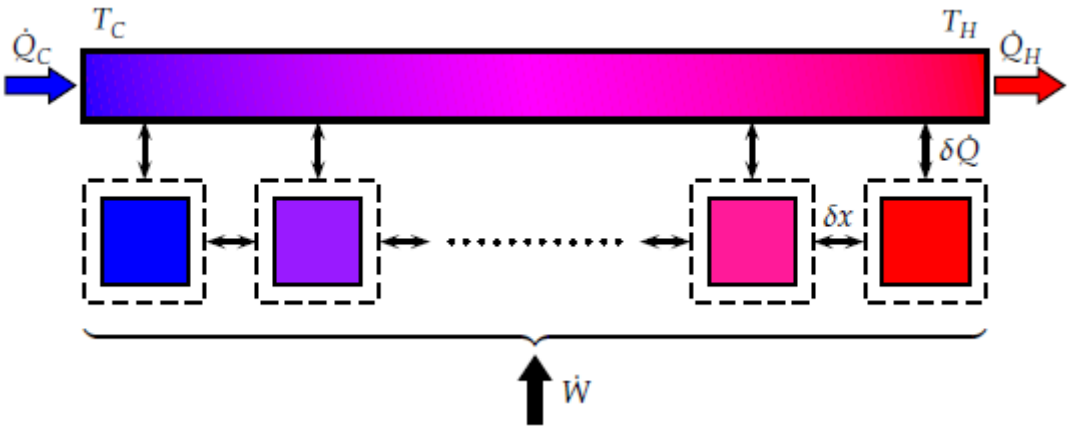


Figure 1.8: Bucket-brigade effect [7]

A fluid parcel interacts thermally with the boundary and heat is transferred to the fluid parcel. The fluid parcel then moves a small distance to either a hotter or colder region of the boundary, before it transfers the heat back to the boundary. If the region is colder, the system functions as a prime mover. If the heat is released at the hotter region, the system functions as a heat pump.

1.2 Thermoacoustics in computational fluid dynamics

In numerical analysis of thermoacoustic devices often frequency domain solvers are used. For example, thermoacoustic software *DeltaEC* uses the second order Helmholtz equation by assuming sinusoidal time dependence of the wave equation, and an energy equation added optionally [25]. Many other assumptions were also made by using these equations as a basis, among which laminar and one dimensional flow are two. For a lot of thermoacoustic research, frequency domain solvers are sufficient, but for certain applications a time-domain solver is needed, or preferred. Such applications include analysis of complex flow patterns, and streaming effects.

The Helmholtz equation has frequency and position as its independent variables, and can be evaluated using impedance, or the complex reflection coefficient, at bounding surfaces or points of the domain. Complete thermoacoustic parts can be represented by an impedance or reflection coefficient. This simplifies the thermoacoustic problems considerably, and decreases domain size.

Impedance is a quantity that relates the amount of sound pressure that is generated to acoustic particle velocity. It can be a characteristic of the medium, called *characteristic impedance*, or from an acoustic component. The reflection coefficient relates the pressure of the incoming acoustic wave to the reflected acoustic wave at a bounding surface of an acoustic component. Impedance and reflection are complex quantities defined in the frequency domain.

For time-domain fluid dynamics equations such as the Navier-Stokes equations, and its derived Euler equations and wave equation, these quantities cannot be used directly because they are only defined in the frequency domain. Conventional boundary conditions, such as a constant pressure outlet, usually behave acoustically fully reflective. One can circumvent these problems in CFD by modeling the whole domain. This is computationally very expensive, especially considering one is usually only interested in a certain part of the thermoacoustic device.

A potential solution to decrease domain size, while maintaining correct acoustical properties, is to use so-called Time-Domain Impedance Boundary Conditions (TD-IBC). These type of boundary conditions mimic the impedance, or the complex reflection coefficient, in the time-domain. Several papers are written about this topic in literature, but none are actually available in commercially available time-domain CFD packages such as *ANSYS CFX* and *ANSYS FLUENT*. The goal of this master thesis is to implement and test a time-domain impedance boundary condition in *ANSYS CFX*, and provide a theoretical reference on the matter for future work.

1.3 Thesis outline

This report is sectioned in a very conventional structure, in chronological order: theory, method, results, and conclusions. The actual process during the research did not strictly follow this order, especially between method and results. Within the method chapter often references towards the "future" results chapter will be made, while the method chapter itself is (i.e. the boundary conditions), in a way, also a result of this thesis.

Chapter 2 is the Theory chapter, which gives the underlying theory of constructing a TD-IBC. Also related theory such as the Navier-Stokes Characteristic Boundary Conditions (NSCBC), the Linear Relaxation Method (LRM), Plane-Wave Masking (PWM), and the multi-microphone method are discussed.

Chapter 3 covers the methodology of simulations and construction of the boundary conditions in *ANSYS CFX*. An overview of the test cases are given, and their rationale is explained. Also an overview and the main design choices of the programmed *Fortran* subroutines in *ANSYS CFX* is discussed.

The last two chapters, chapters 4 and 5, covers the results, conclusions, and recommendations. The results chapter features the influence of the relaxation factor on the reflection coefficient, the influence of mean flow, an analysis of pressure drift, and an analysis of the error in the complex plane of the TD-IBC as the most important topics. To prevent clutter and a plethora of numbers, results are mainly presented by graphs. The actual data used to construct the graphs can be found in the appendix. In the last chapter a short overview of the conclusions and recommendations are given.

References to the appendix are not given specifically in the report, but just referenced to the appendix as a whole. The specific page numbers and headings can be found in the table of contents of the appendix, at the beginning of the appendix.

Chapter 2

Theory

Different methods for constructing a Time-Domain Impedance Boundary Condition (TD-IBC) exist. A method created by Fung uses a direct inversion of the reflection coefficient R from the frequency domain to the time-domain, instead of the (equivalent) impedance Z (see section 2.2), but is not always stable and relatively difficult to implement [4][5]. For this master thesis a different method is used, a method created by Polfike et al and based on a so-called *filter method* [9][6]. The reason behind this choice is the inherent modularity of the approach. This modularity allows for a step-wise implementation and debugging, and can more easily be tested against theory and other implementations at various points in the process. The TD-IBC consists of two main parts, a non-reflective part, and a TD-IBC specific part. The non-reflective part is based on a method for constructing boundary conditions called the *Navier-Stokes Characteristic Boundary Condition* (NSCBC) formalism (see section 2.3) [22][15]. The non-reflective part can function independently as a non-reflective boundary condition when the TD-IBC specific part is not added to the formulation. The TD-IBC specific part consists of the filter method (see section 2.6).

Optionally other terms can be added to the boundary specification. To solve stability problems discussed in section 2.4, a so-called *Linear-Relaxation Method* (LRM) can be used [22]. The downside of the LRM method is additional reflections. These additional reflections can subsequently be mitigated by adding a *Plane-Wave Masking* (PWM) term [16]. Strictly speaking the LRM method is not required, for example for certain short simulations which remain stable within its simulation time, but in practice the LRM term must be added to prevent stability problems.

The construction of the TD-IBC is presented graphically in Figure 2.1. All contributions can be added as a sum of terms, similar to the superposition principle in linear systems.

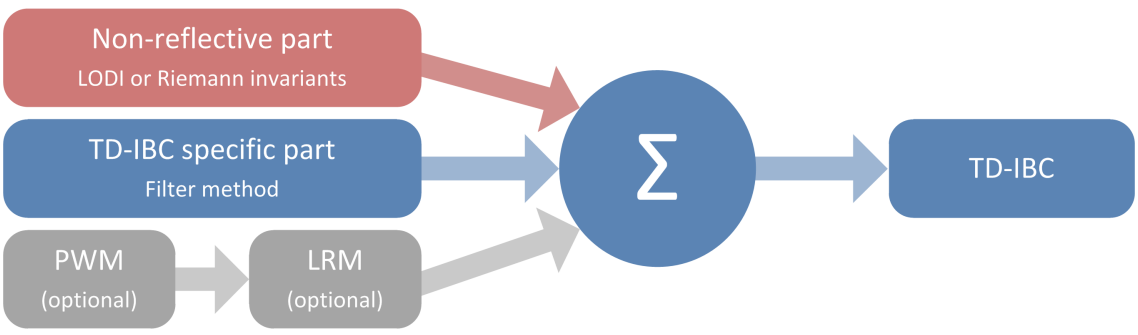


Figure 2.1: Construction of a TD-IBC boundary condition

ANSYS CFX already provides a non-reflective boundary condition as a beta feature. Conveniently, this built-in non-reflective boundary condition is also based on the NSCBC formalism using the Locally One-Dimensional Inviscid (LODI) relations, and the LRM method for stability [26]. See section 2.3.1 and 2.3.2 for details about the LODI relations. The *ANSYS CFX* non-reflective boundary condition (NRBC) will be used as a reference in the results chapter.

2.1 Boundary Conditions

Differential equations in fluid dynamics need additional constraints at the boundaries to be able to find a solution, thermoacoustic applications in CFD will be no exception to this. These type of differential equations are called *boundary value problems*, and the accompanying constraints are called *boundary conditions*. Mathematically, a problem that has a solution that is unique, exists, and does not behave chaotically is called a well-posed problem. To state that a problem is well-posed in CFD problems based on the Navier-Stokes equations is in general not possible, because it has never been proven that a solution always exists. One can therefore, in this case, only say that a problem is more well-posed, or less well-posed, compared to a reference problem based on the Navier-Stokes equations.

Different kind of boundary conditions exist of which certain types are considered "stronger" than others, i.e. resulting in a problem being more well-posed. In practice this statement means that a solver (software that calculates the solution) is usually more able to find a solution to the problem if the boundary condition is considered stronger. A few important boundary conditions are summed up in Table 2.1. The boundary conditions are explained using a model differential equation where y is the dependent variable of the independent variable x, Ω is the domain, $\partial\Omega$ is the boundary of the domain, $\alpha(x)$ and $\beta(x)$ are given values on the boundary, and a and b are constants which may, or may not, be a function of x.

Table 2.1: A few important types of boundary conditions

Boundary Condition	Formula
Dirichlet	$y(x \in \partial\Omega) = \alpha$
Neumann	$y'(x \in \partial\Omega) = \alpha$
Cauchy	$y(x \in \partial\Omega) = \alpha$ and $y'(x \in \partial\Omega) = \beta$
Robin	$a \cdot y(x \in \partial\Omega) + b \cdot y'(x \in \partial\Omega) = \alpha$

The *Dirichlet* boundary condition specifies the values of the dependent variable (y) at the boundary, and the *Neumann* boundary condition specifies the derivative of the dependent variable (y') at the boundary. A Dirichlet boundary condition is considered stronger than a Neumann boundary condition (in differentiation information is lost, the integration constant). The others are combinations of aforementioned boundary conditions. The Cauchy boundary condition specifies both y and y', while the Robin boundary condition specifies y via a linear combination of both y and y'.

After derivation of the theory about non-reflective and TD-IBC boundaries later on in the this chapter, it will become clear that the resulting boundary conditions are of a "weaker" type than a constant Dirichlet boundary condition often used in CFD (e.g. constant pressure outlet). This can result in stability and convergence problems.

2.2 Governing equations

The most important set of equations in fluid dynamics are the *Navier-Stokes* equations. These equations can describe many flow problems, and lie at the basis of many derived and simpler equations. Important equations in acoustics, such as the wave equation, Helmholtz equation, and the Euler equations are also derived from the Navier-stokes equations. The notation and form of the Navier-Stokes equations differ. In this thesis a compact differential form (eq. 2.2) using four column vectors (eq. 2.1) is used to enhance readability, but other forms exist. The differential form is also used in the characteristic analysis later on in section 2.3.

$$\mathbf{U} = \begin{Bmatrix} \rho \\ \rho\vec{u} \\ \rho E \end{Bmatrix} \quad \vec{\mathbf{F}}^{inv} = \begin{Bmatrix} \rho(\vec{u} - \vec{u}_{\partial V}) \\ \rho\vec{u}(\vec{u} - \vec{u}_{\partial V}) + p\bar{\bar{I}} \\ \rho E(\vec{u} - \vec{u}_{\partial V}) + p\vec{u} \end{Bmatrix} \quad \vec{\mathbf{F}}^{visc} = \begin{Bmatrix} \vec{0} \\ \bar{\bar{\tau}} \\ \bar{\bar{\tau}}\vec{u} - \vec{q} \end{Bmatrix} \quad \mathbf{J} = \begin{Bmatrix} 0 \\ \rho\vec{f} \\ \rho\vec{f} \cdot \vec{u} + \dot{Q} \end{Bmatrix} \tag{2.1}$$

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot (\vec{\mathbf{F}}^{inv} - \vec{\mathbf{F}}^{visc}) = \mathbf{J} \tag{2.2}$$

In equation 2.1 and 2.2, $\mathbf{U}$ is a column vector with the conserved quantities, $\vec{\mathbf{F}}^{inv}$ are the inviscid flux terms, $\vec{\mathbf{F}}^{visc}$ are the viscous flux terms, and $\mathbf{J}$ are the source terms. ρ is the density, $\vec{u}$ is the velocity vector, E the total specific energy (eq.2.3), p the pressure, $\bar{\bar{\tau}}$ the stress tensor, $\vec{q}$ the heat conduction, $\vec{f}$ the gravity (or any other arbitrary volumetric force field), and $\dot{Q}$ as a volumetric heat source. The three-dimensional Navier-Stokes equations contain 7 unknowns and 5 equations, leaving the system underdetermined. To solve the system of equations, two additional equations need to be supplied, which are usually state equations such as the ideal gas law (eq. 2.4).

$$E = e + \frac{1}{2}|u|^2 \tag{2.3}$$

$$P = \rho RT \tag{2.4}$$

In acoustics, the viscous terms and heat conduction terms are generally small, and can be neglected. From the compact form of the Navier-Stokes equations (eq. 2.2), the Euler equations (eq. 2.5) can easily be constructed by removing the viscous flux terms $\vec{\mathbf{F}}^{visc}$ vector.

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}}^{inv} = \mathbf{J} \tag{2.5}$$

From the Euler equations (eq. 2.5) a partial differential equation important to acoustics can be derived, which is called *the wave equation* (2.6). An important form of the wave equation is the scalar velocity potential form in equation 2.7, because only one variable instead of three needs to be solved for the three spatial dimensions. Apart from its application in acoustics, the wave equation can also be used to describe other wave phenomena such light waves and water waves.

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u \tag{2.6}$$

$$\vec{\nabla}^2 \phi - \frac{1}{c_0^2}\phi_{tt} = 0 \qquad \vec{\nabla}\phi = \vec{u} \tag{2.7}$$

The wave equation is a hyperbolic second-order linear Partial Differential Equation (PDE). Solutions of hyperbolic equations exhibit wave-like behavior. Any disturbance made to the

initial conditions are not felt immediately throughout the domain, but instead are propagated at a finite *propagation speed*. For the wave equation this propagation speed is equal to the *characteristic speed* c_0. The wave equation is also a linear PDE, which means solutions can be added to form other solutions (called *superposition principle*). The general solution to the wave equation is given by equation 2.8. The functions f and g represent right and left traveling waves respectively, and the solution is thus a sum of both (which can be a solution according to the *superposition principle*).

$$u(x,t) = f(x - ct) + g(x + ct) \tag{2.8}$$

Another important equation in acoustics is the *Helmholtz equation*. The Helmholtz equation can be seen as a time-independent version of the wave equation. Using separation of variables, and assuming time-harmonic behavior, the solution can be written in the form shown in equation 2.9. Substitution of this form of the solution in the wave equation (eq. 2.7) results in the Helmholtz equation (eq. 2.10) (see appendix for detailed derivation). The wave equation can also be expressed in the pressure amplitude $\delta\hat{p}$ instead of the velocity potential ϕ (eq. 2.11). The solution of the Helmholtz equation for the pressure, is of the form given by equation 2.12, which is, unlike the solution of the wave equation, not an equation of space x and time t, but of space x and (angular) frequency ω. An important advantage of this equation is that frequency dependent quantities such as impedance and reflection can be used directly.

$$\phi(\vec{x},t) = \phi(\vec{x},\omega)e^{i\omega t} \tag{2.9}$$

$$\vec{\nabla}^2\phi(\vec{x},\omega) + k^2\phi(\vec{x},\omega) = 0 \qquad where \quad k = \frac{\omega}{c} \tag{2.10}$$

$$\frac{\partial^2\delta\hat{p}}{\partial x^2} + k^2\delta\hat{p} = 0 \tag{2.11}$$

$$\delta\hat{p}(x,f) = A(f)\cdot e^{-ikx} + B(f)\cdot e^{ikx} \tag{2.12}$$

An often used property to describe the acoustic behavior of media or acoustic components is the *acoustic impedance* Z (eq. 2.13). The acoustic impedance relates the acoustic pressure amplitude $\delta\hat{p}$ to the acoustic velocity amplitude $\delta\hat{u}$, and is used for acoustic components. A special type of impedance is the so-called *characteristic acoustic impedance* Z_0, and is used to describe the acoustic behavior of a medium (eq. 2.13). A high impedance means that a lot of acoustic pressure is needed to move the fluid parcels, while a low impedance means the fluid parcels are relatively easily moved.

$$Z = \frac{\delta\hat{p}}{\delta\hat{u}} \qquad Z_0 = \rho_0 c_0 \qquad Z \in \mathbb{C} \tag{2.13}$$

The acoustic impedance is equivalent to the reflection coefficient R, given by equation 2.14. Where f and g are so-called *characteristic wave amplitudes* given by equation 2.8, and explained in section 2.3. The reflection coefficient R is the ratio of the acoustic pressure that is reflected ($\delta\hat{p}^-$) from an acoustic component over the outward traveling acoustic pressure ($\delta\hat{p}^+$), and is a measure for reflectivity. The part that is not reflected is transmitted ($\delta\hat{p}^{tr}$). The reflection coefficient R can become complex, signifying a phase difference between the incoming and outgoing waves.

$$R = \frac{\delta\hat{p}^-}{\delta\hat{p}^+} \qquad R \in \mathbb{C} \tag{2.14}$$

$$R = \frac{Z - Z_0}{Z + Z_0} \tag{2.15}$$

In acoustics three special cases can be distinguished, a *hard wall*, a *pressure release surface*, and a *matched impedance*. These cases are explained using Figure 2.2, which represents two domains connected by an interacting surface (interface). When it will take a lot more acoustic pressure to move the fluid parcels in domain 2 compared to domain 1 ($Z_2 >> Z_1$), the interface is called a *hard wall* and the reflection coefficient R will be 1. When on the other hand the fluid parcels move far more easily in domain 2 compared to 1 ($Z_2 << Z_1$), the interface is called a *pressure release surface* and the reflection coefficient R will be -1. The last special case is the *matched impedance*, in which the impedance on both sides are equal ($Z_2 = Z_1$), and is thus non-reflective ($R = 0$).

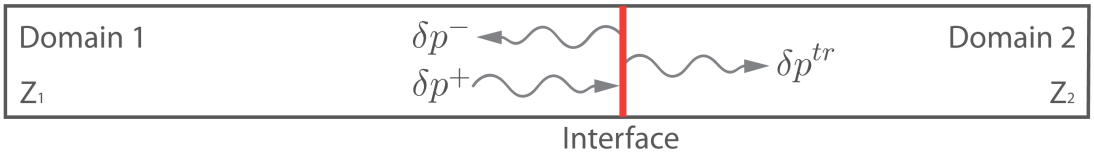


Figure 2.2: Reflection caused by an impedance difference

A non-reflective boundary is equivalent to a *matched impedance* surface. A *hard wall* can be simulated in CFD by using a constant velocity outlet (e.g. wall boundary condition), and a *pressure release surface* by using a constant pressure outlet boundary condition.

2.3 Navier-Stokes Characteristic Boundary Conditions

As is mentioned in the introduction (sec. 1.2), the conventional boundary conditions in CFD software fully reflect acoustic waves back into the domain. These conventional boundary conditions are however quite "strong" in the sense of well-posedness (see sec. 2.1 for more information). Ideally a non-reflective boundary condition, as the name implies, should let acoustic waves propagate through the boundary undisturbed, but at the same time be as well-posed enough for the solver to be able to find a solution. In practice however, a non-reflective boundary condition is always a concession between well-posedness and reflectivity, which will become clear in section 2.4.

The NSCBC formalism used in the non-reflective part of the TD-IBC is a method to construct boundary conditions based on the characteristic wave relations of the (modified) hyperbolic Euler equations. The characteristic wave relations can be derived by a method called *method of characteristics*, and applies to hyperbolic PDEs such as the Euler equations. The *method of characteristics* is a technique to decouple a set of coupled partial differential equations into a set of decoupled ordinary differential equations (ODEs), in order to solve the problem more easily. The hyperbolic one-dimensional wave equation (eq. 2.6) is considered one of the simplest hyperbolic equations, and is used in the next paragraph as an example to explain a few important properties of *hyperbolicity* and the *method of characteristics*. The solution of hyperbolic PDEs are considered "wave-like", which means that disturbances made to the initial conditions propagate at a finite speed (called *characteristic speed*) trough space and time, and hence, are not felt immediately in every point of the domain. In case of the one-dimensional wave equation, the characteristic speed is equal to the sound speed c_0 and couples both independent variables x (space) and t (time), of which so-called *characteristic curves* (or just *characteristics*) can be drawn ,originating from an initial condition in a diagram called the *characteristic*

diagram (see Fig. 2.3b). By using the *method of characteristics* a set of *characteristic waves* (also called *Riemann invariants*) can be constructed, which are variables that are constant along the accompanying *characteristic curves*. An expression for these *characteristic waves* are comprised of the dependent variables, and their coefficients from the originating equation. For the one-dimensional wave equation the *characteristic waves* f and g are given by equation 2.16. f and g are equivalent to the same variables in equation 2.8 for the general solution to the wave equation.

$$f = \frac{1}{2}\left(\frac{\delta p}{\rho_0 c_0} + \delta u\right) \qquad g = \frac{1}{2}\left(\frac{\delta p}{\rho_0 c_0} - \delta u\right) \tag{2.16}$$

Where δp is the acoustic contribution to the pressure (or simply acoustic pressure), and δu the acoustic contribution to the velocity (or simple acoustic velocity).

Characteristics and *characteristic waves* can also be shown graphically, as is done in Figure 2.3. In Figure 2.3a one can see the acoustic wave traveling to the right, and in Figure 2.3b one can see the accompanying *characteristic* in the *characteristic diagram*.

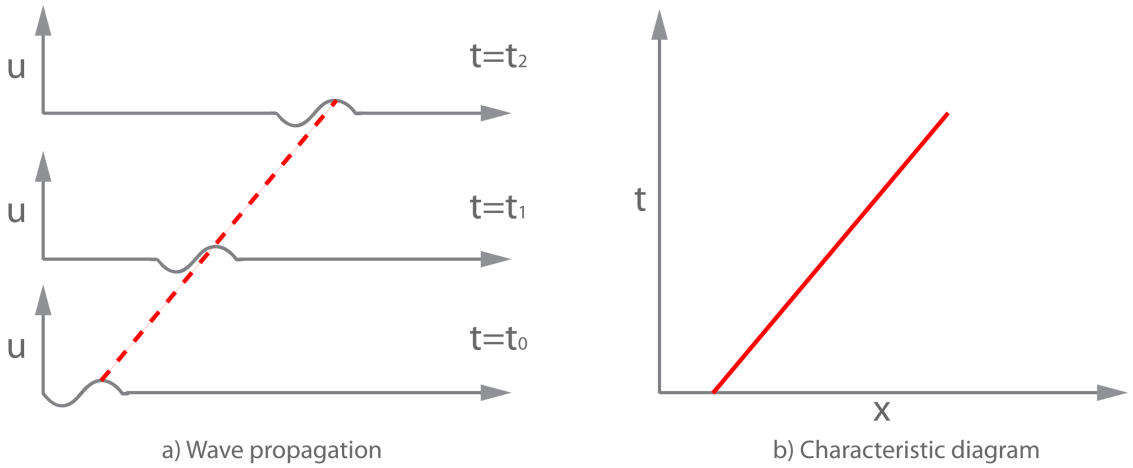


Figure 2.3: Characteristics for the wave-equation

The Euler equations are more complex than the above wave equation, and characteristic analysis in section 2.3.1 will result in five different characteristics speeds and characteristic wave expressions. The advantage of the *method of characteristics* is that different contributions to the solution can be separated, of which each can be integrated in time via relatively easy expressions to obtain the next value in time. As one will observe in the characteristic analysis in section 2.3.1, two of the five characteristic wave expressions are essentially the same as the characteristic wave expressions given in equation 2.16 for the wave equation, except for the used NSCBC form being a function of the gradients of the primitive quantities pressure and velocity. The constructed non-reflective boundary condition, the NSCBC formalism exploits these properties to calculate and set the pressure at the boundary for the next timestep accordingly.

More extensive analysis and a more detailed derivation of the NSCBC formalism is added to the appendix.

2.3.1 Characteristic analysis

To be able to derive the characteristic wave relations, a modified version of the Euler equations must be used. One might think that the use of the Euler equations to construct the NSCBC method severely limits the types of flow that can be modeled, but this is not necessarily the case as the NSCBC method is only used locally (near the boundary) to calculate and specify boundary values, and does not bother with the calculation of the interior of the domain. The approximation at the boundaries is accurate for flows that are pre-dominantly hyperbolic in nature, which is the case for flows with a Reynolds number (see eq. 2.17) significantly higher than one. In the asymptotic limit of $Re \to \infty$, the momentum equation of the Navier-Stokes equations leads to the hyperbolic Euler equations. On the other hand, if the Reynolds number goes to zero, the flow is dominated by viscosity μ and will result in the elliptic Stokes equation [24]. Often thermoacoustic devices use low viscocity fluids such as air or helium as a medium, to minimize viscous losses, and so the approximation is justified. It is therefore possible to use the full Navier-Stokes equations as the governing equations for the interior of the domain, and use the NSCBC method based on the modified Euler equations to calculate the values at the boundary.

$$Re = \frac{\rho U_\infty L}{\mu} \tag{2.17}$$

Where U_∞ is the free stream velocity, and L is the characteristic length of the domain. In the characteristic analysis leading to the NSCBC formalism, Thompson modifies the one-dimensional Euler equation in such a way that it fits the quasi-linear advection equation (see eq. 2.18) [22]. This quasi-linear advection equation is a model equation often used to determine if a system of PDEs is hyperbolic, and subsequently decouple the set of PDEs and put in characteristic form.

$$\frac{\partial \mathbf{U}^p}{\partial t} + \mathbf{A}^1(\mathbf{U}^p)\frac{\partial \mathbf{U}^p}{\partial x_1} = 0 \tag{2.18}$$

Where $\mathbf{U}^p$ denotes the primitive quantity vector instead of the conserved quantity vector $\mathbf{U}$, because it simplifies derivation [22]. The primitive quantity vector is given in the following expression: $\mathbf{U}^p = \{\rho, p, \vec{u}\}^{-1}$. What exactly constitutes "modified" in the modified Euler equations are three assumptions and modifications, listed below.

1. A quasi-linear form realised by using a primitive quantity vector $\mathbf{U}^p$.

2. Neglecting transverse terms, because else the system cannot be put in characteristic form for mathematical reasons (see appendix). This basically results in a system with one dominant direction of propagation: the x-axis.

3. Neglecting source terms because they do not significantly contribute to the analysis, and only complicate matters.

A more detailed explanation about the assumptions, modifications (i.e. quasi-linear form) and the full derivation of the characteristic form of the above modified quasi-linear Euler equation (eq. 2.18) is added to the appendix.

Eventually Thompson arrives at five expressions for the temporal derivative of the primitive field quantities (see eq. 2.19). These temporal derivatives are a function of the so-called amplitude variations $\mathcal{L}_i$, which are time variations in the amplitude of the characteristic waves (see eq. 2.20). The amplitude variations are in their turn functions of the gradients of the primitive quantities at the boundary. The complete set of equations given by equations 2.19 and 2.20 are

called Locally One-Dimensional Inviscid (LODI) relations.

$$\begin{array}{rcl} \frac{\partial \rho}{\partial t} & = & -\frac{1}{c^2}[\mathcal{L}_2 + \frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1)] \\ \frac{\partial p}{\partial t} & = & -\frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1) \\ \frac{\partial u_1}{\partial t} & = & -\frac{1}{2\rho c}(\mathcal{L}_5 - \mathcal{L}_1) \\ \frac{\partial u_2}{\partial t} & = & -\mathcal{L}_3 \\ \frac{\partial u_3}{\partial t} & = & -\mathcal{L}_4 \end{array} \tag{2.19}$$

$$\begin{array}{rcl} \mathcal{L}_1 = & \lambda_1(\frac{\partial p}{\partial x_1} - \rho c \frac{\partial u_1}{\partial x_1}) \\ \mathcal{L}_2 = & \lambda_2(c^2 \frac{\partial \rho}{\partial x_1} - \frac{\partial p}{\partial x_1}) \\ \mathcal{L}_3 = & \lambda_3 \frac{\partial u_2}{\partial x_1} \\ \mathcal{L}_4 = & \lambda_4 \frac{\partial u_3}{\partial x_1} \\ \mathcal{L}_5 = & \lambda_5(\frac{\partial p}{\partial x_1} + \rho c \frac{\partial u_1}{\partial x_1}) \end{array} \tag{2.20}$$

The $\mathcal{L}_1$ and $\mathcal{L}_5$ amplitude variations are the relevant variations for the construction of a non-reflective boundary condition. It is assumed that the non-reflective boundary is flat and has a surface normal pointing outwards (of the domain) along the positive x-axis. If this is not the case, the equations must be transformed in terms of a local coordinate system with one coordinate perpendicular to the boundary. This does not add anything to the analysis, but does require extra work, and is a potential source of error. Based on the aforementioned assumptions, $\mathcal{L}_1$ is the amplitude variation of the incoming characteristic wave, and $\mathcal{L}_5$ of the outgoing characteristic wave. In Figure 2.4 the incoming and outgoing amplitude variations of the characteristic waves are visualized.

Figure 2.4: Amplitude variation waves $\mathcal{L}_1$ and $\mathcal{L}_5$

The temporal derivatives of the different primitive variables can be used in conjunction with a numerical time integration scheme to calculate the values at the next timestep. These variables can be set in a CFD code to provide a Dirichlet boundary condition. In the case of a non-reflective boundary condition, the most ideal primitive quantity to set is the pressure, because of two reasons. The first being the fact that only the "acoustic" characteristics $\mathcal{L}_1$ and $\mathcal{L}_5$ are used the specification of the temporal derivative of the pressure. The second reason is that specifying the pressure instead of the velocity at an outlet boundary condition is considered more stable in *ANSYS CFX* [23]. However, this does not restrict non-reflective boundary conditions to be of the "pressure outlet" type, velocity specified and other type of boundaries could also be created based on the NSCBC method.

As $\mathcal{L}_1$ is the amplitude variation of the incoming, and thus reflected, wave, it should ideally be zero in a non-reflective outlet boundary condition. By explicitly assigning a value of zero to $\mathcal{L}_1$, and calculating $\mathcal{L}_5$ via one-sided differences of the gradients of the pressure and velocity near the boundary, an expression for the time-derivative of the pressure for a non-reflective boundary condition is constructed.

2.3.2 Determination of outgoing wave amplitude variation

The method based on the LODI relations to determine the outgoing amplitude variation $\mathcal{L}_5$ is not the only method of determining $\mathcal{L}_5$. Another method to determine $\mathcal{L}_5$ uses a slightly different, but equivalent, expression for the characteristic wave f (eq. 2.21, based on eq. 2.16). The characteristic wave f, which now has units of velocity ($^m/_s$), cannot be determined at the boundary because both values for pressure and velocity are needed. As the whole boundary condition is constructed such that pressure is specified, the pressure itself cannot be used in calculating the pressure, because it would create an infinite recursive loop. The pressure and velocity is retrieved on a sample plane near the boundary, and offset in time (using the sound speed) towards the boundary. The method is not mentioned by Thompson [22] and Poinsot [15], but mentioned by Polifke [16] in conjunction with the Plane-Wave masking method.

$$f = \frac{1}{2}\left(\frac{\delta p}{\rho_0 c_0} + \delta u\right) \tag{2.21}$$

The quantities used in the expression for characteristic wave f are not pressure and velocity, but the acoustic pressure and velocity. These quantities cannot be directly retrieved from the flow solution, and must be derived from expressions. The expressions for determining the acoustic contributions of the pressure and velocity are the same as used in the Plane-Wave masking method, and treated in section 2.5.

2.4 Linear relaxation method

The NSCBC method to construct a Non-Reflecting Boundary Condition (NRBC) will result in a drift of the mean pressure with time, and ultimately convergence problems [15]. Fundamentally, boundary conditions carry information from the boundary to the interior in order to find a solution to the problem. By eliminating the amplitude variation $\mathcal{L}_1$ of the incoming characteristic wave, no information about the pressure is able propagate from the boundary to the interior [17]. In other words, the boundary is "invisible" to the interior of domain, and the problem effectively becomes ill-posed.

The most used, and relatively simple, solution to the problem is the Linear Relaxation Method (LRM) [15][17]. This method is also used by *ANSYS CFX* in its NRBC to prevent pressure drift [26]. In the LRM method, the expression for $\mathcal{L}_1$ is adjusted to include a term that restores a deviation from a target pressure, represented by p_∞ (eq. 2.22). The target pressure p_∞, sometimes called the *far-field* pressure, is the pressure to which one wants the pressure to be restored. The relaxation factor K is a factor that represents the amount by which the deviation is restored.

$$\mathcal{L}_1 = K \cdot (p - p_\infty) \tag{2.22}$$

Of course, this term will by definition cause reflections. The non-reflective boundary condition using the LRM method for specifying $\mathcal{L}_1$ will always be a compromise between drift (convergence) and reflection. The better the target pressure is enforced using relaxation factor K, the higher the reflection coefficient R (eq. 2.14), and vice versa. But also frequency plays a part, the higher the frequency, the less time the LRM term has to correct the deviation in pressure, resulting in a lower reflection coefficient, while for a low frequency the LRM term can easily correct for the pressure deviation within the period of the wave, and results in a higher reflection coefficient. Selle devised an analytical expression to determine the theoretical complex reflection coefficient R as a function of frequency (see Fig. 2.5) [17]. The magnitude

$||R_{theo}||$ and phase ψ of the complex reflection coefficient R are given by equation 2.23.

$$||R_{theo}|| = \frac{1}{\sqrt{1+(\frac{2\omega}{K})^2}} \qquad and \qquad \psi = -\pi - arctan(\frac{2\omega}{K}) \tag{2.23}$$

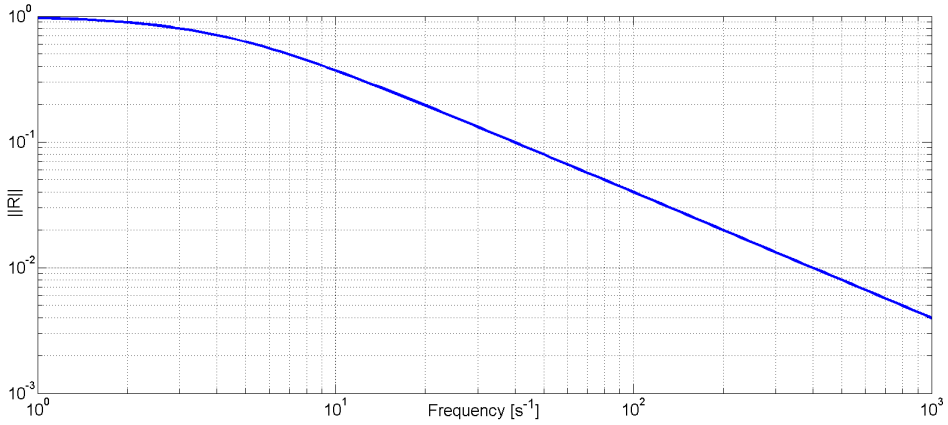


Figure 2.5: Theoretical reflection coefficient vs. frequency ($K = 50^1/_s$)

ANSYS CFX uses an expression proposed by Selle to determine an optimal relaxation factor K (eq. 2.24) [17]. The parameters that can be used to determine the optimal value for K are the free stream mach number $\mathcal{M}$, the sound speed c_0, a characteristic length L, and a coupling parameter σ. The scaling method from Selle is based on domain dimensions and eigenfrequencies of the domain. Two optimal coupling parameters σ are mentioned by Selle, a value of 0.58 based on numerical simulations, and a value of 0.27 based on theory. See appendix for more information.

$$K_{opt} = \sigma\frac{(1-\mathcal{M}^2)}{L} \tag{2.24}$$

The *ANSYS CFX* NRBC is tested and compared to the theoretical values by Oosterhuis [14]. Results showed a good agreement with theoretical values for higher relaxation factors, but showed a significantly higher reflection coefficient for lower relaxation factor. Proposed hypothesis for the phenomenon where a loss in accuracy due to numerical discretization schemes used, and a built-in mechanism in *ANSYS CFX* that is deliberately adjusting the boundary values to prevent pressure drift and convergence problems.

2.5 Plane-Wave Masking

As is discussed in section, 2.4 the LRM method is always a compromise between pressure drift and reflection. Polifke introduces a method called Plane-Wave Masking (PWM) to minimize the reflection caused by the LRM method, especially at lower frequencies (see Fig. 2.5). The PWM method determines the amplitude of the characteristic wave f (eq. 2.21) at a sample plane near the boundary, and off-sets it in time until it reaches the boundary. The acoustic contribution of the characteristic wave f traveling towards the boundary is then subtracted from the LRM term (eq. 2.25), effectively "masking" the pressure deviation caused by acoustics. What theoretically remains are pressure deviations caused by mean pressure drift and turbulence.

$$\mathcal{L}_1 = K \cdot (p - p_\infty - \rho_0 c_0 f) \tag{2.25}$$

The acoustic pressure and velocity cannot be directly retrieved from the flow solution. To determine the acoustic pressure and velocity, Polifke et al proposes to separate the pressure into three different contributions (see eq. 2.26). Where p_0 is the mean pressure, δp is the acoustic pressure, and p' is the turbulent component of the pressure. The velocity u is separated analogously in equation 2.27.

$$p(\vec{x}, t) = p_0(\vec{x}) + \delta p(x, t) + p'(\vec{x}, t) \tag{2.26}$$

$$u(\vec{x}, t) = u_0(\vec{x}) + \delta u(x, t) + u'(\vec{x}, t) \tag{2.27}$$

Polifke assumes that the characteristic wave f is a plane-wave that travels perpendicular to the sample plane. Mathematically this is equal to saying that the integrated area average over the sample plane, signified by the angled brackets, is equal to f (see eq. 2.28). The acoustic wave can be assumed a plane-wave if the acoustic boundary layer is thin, which it is in the numerical problem defined in section 3.

$$f \approx \langle f \rangle \tag{2.28}$$

Then Polifke reasons that if the turbulent length scale is sufficiently small, the area average of the turbulent contribution to both pressure and velocity practically vanishes (see eq. 2.29). No turbulence models will be used in the setup of the numerical problem, and thus this is indeed the case in this report (see Chapter 3). With the turbulent contribution assumed to be zero, equations 2.26 and 2.27 can be rewritten for the area averaged acoustic pressure and velocity (see eq. 2.30).

$$\langle p' \rangle \approx 0 \qquad \langle u' \rangle \approx 0 \tag{2.29}$$

$$\tilde{p} \equiv \langle p - \bar{p} \rangle \qquad \tilde{u} \equiv \langle u - \bar{u} \rangle \tag{2.30}$$

The area averaged acoustic pressure and velocity are now used in the expression for characteristic wave f (eq. 2.21), which results in an area averaged expression for f (eq. 2.31).

$$f = \frac{1}{2}\left(\frac{\tilde{p}}{\rho_0 c_0} + \tilde{u}\right) \tag{2.31}$$

The mean pressure can be taken as the target pressure at the boundary p_∞, or as a time averaged pressure. The pressure is retrieved from the solution field.

2.6 Time-Domain Impedance Boundary Conditions

The TD-IBC specific part is based on the LRM model and PWM for specifying $\mathcal{L}_1$ (eq. 2.25), but extended by an external perturbation g^* which is calculated using so-called filter methods [9][6]. Using filter-methods (explained later on) the new expression for $\mathcal{L}_1$ is given by equation 2.32. The external perturbation g^* within the brackets, preceded by the relaxation factor K, is for "masking" the contribution of g^* to the LRM term, i.e. reduce unwanted reflections caused by g^*. The temporal derivative of g^* is the actual TD-IBC contribution.

$$\mathcal{L}_1 = K(p - \rho c(f + g^*) - p_\infty) - \rho c \frac{\partial g^*}{\partial t} \tag{2.32}$$

From an arbitrary reflection coefficient, or equivalently, impedance (see sec. 2.2), an external perturbation g^* with the desired properties mimicking the aforementioned reflection coefficient can possibly be obtained. Possibly, because for certain reflection coefficients that are a function of multiple frequencies, the process of determining g^* can become very difficult, of which the details will be explained in section 2.6.1 and 2.6.3.

To construct g^*, first the definition of the reflection coefficient 2.14 in the frequency domain is rewritten with the backward traveling wave $\hat{g}$ expressed as the product of the reflection coefficient R and the forward traveling characteristic wave amplitude $\hat{f}$ in equation 2.33.

$$R(\omega) = \frac{\widehat{g}(\omega)}{\widehat{f}(\omega)} \rightarrow \widehat{g}(\omega) = R(\omega)\widehat{f}(\omega) \tag{2.33}$$

Then the product (eq. 2.33) is transformed to the time domain using the property that a product of two functions in the frequency domain is equivalent to a convolution in the time-domain, resulting in equation 2.34. Where τ is a time coordinate, comparable to t. The integral is evaluated from 0 to ∞ instead of $-\infty$ to ∞, because values for τ lower than 0 correspond to future values, which are not available in the present. This is equivalent to stating the system must be causal in time.

$$g(t) = (R * f)(t) = \int_0^\infty R(\tau) f(t-\tau) d\tau \tag{2.34}$$

The evaluation of a convolution integral (see eq. 2.34) is computationally very expensive and difficult. To be able to solve this convolution integral, equation 2.33 is converted into the Laplace domain by exchanging the arguments via the relation $s = i\omega$, and the Z-transform is used to solve the equation at discrete timesteps. The Z-transform is also called the "discrete Laplace transform", and allows for a relatively efficient evaluation of equation 2.33. Using the Z-transform, equation 2.33 can be rewritting in the z-domain as equation 2.35. It is then assumed that $R(z)$ can be approximated by the transfer function syntax in equation 2.36.

$$G(z) = R(z)F(z) \tag{2.35}$$

$$R(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}{1 - b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m}} \tag{2.36}$$

The expression for $R(z)$ in equation 2.36 can be substituted in equation 2.35, and rewritten into equation 2.37. By using the time shift property of the z-transformation (see eq. 2.38) one arrives at a time-domain expression for the external perturbation g^* in equation 2.39.

$$(1 - b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m})G(z) = (a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n})F(z) \tag{2.37}$$

$$\mathcal{Z}\{x(i-k)\} \leftrightarrow z^{-k}X(z) \tag{2.38}$$

$$g(i) = a_0 f(i) + a_1 f(i-1) + \ldots + a_i f(i-n) - b_1 g(i-1) - \ldots - b_m g(i-m) \tag{2.39}$$

Equation 2.39 is rewritten in short summation notation in equation 2.40, and expressed in time t and time increment Δt.

$$g(t) = \sum_{i=0}^{n} a_i f(t - i\delta t) - \sum_{j=1}^{m} b_j g(t - j\delta t) \tag{2.40}$$

Which is the sum of the present and past incoming and outgoing characteristic wave amplitudes multiplied by a constant. The constants are called filter coefficients, and are determined based on the desired complex reflection coefficient as a function of frequency in a process called *complex-curve fitting*, which is discussed in section 2.6.1.

2.6.1 Determination of filter coefficients

The filter coefficients can be determined from the complex reflection in different ways, two of which are discussed and used. If only a certain reflection coefficient at one given frequency is needed, one can relatively easily calculate the corresponding filter coefficients from the time delay t_{delay} corresponding to the phase delay at the aforementioned frequency (see eq. 2.41) by noting that the different filter coefficients correspond to a different time delay of the incoming and outgoing waves (see eq. 2.40). The filter coefficient in question can be calculated via equation 2.42. Equation 2.42 can also be used to estimate the minimum number of filter coefficients.

$$t_{delay}(f,\phi) = (\frac{\psi}{2\pi}) \cdot f^{-1} \qquad where \quad \psi = phase(R) \quad [rad] \tag{2.41}$$

$$t_{delay} = n \cdot \Delta t \quad \rightarrow \quad a_n = abs(R) \tag{2.42}$$

It must be noted however, that a constant time delay causes the phase of the reflection coefficient to shift linearly with frequency, as Figure 2.6 shows qualitatively. But as the number of filter coefficients increases, so do the memory requirements and susceptibility to quantization error.

Figure 2.6: Phase shift at different frequencies

From equations 2.41 and 2.42 the filter coefficients needed to mimic an acoustically hard wall ($R = 1$) and pressure release surface ($R = -1$) can also be obtained (see sec. 2.2). For an acoustically hard wall the reflection coefficient has no phase, and thus time delay, and results in setting filter coefficient a_0 to 1. For a pressure release surface the phase ψ is $-\pi$, resulting in a time delay of half the period of the incoming wave. For a non-reflecting outlet the filter coefficients are all zero.

It becomes more difficult if the reflection coefficient is specified as a function of frequency at multiple frequencies. It is assumed that the reflection coefficient R can be approximated as a rational function (ratio of two polynomials) in equation 2.36, these polynomials are then fitted to the given points by means of minimizing the error squared. An example of such a fit, done by Polifke et al, is shown in Fig. 2.7 for a short duct [9]. This process of *complex-curve fitting* is discussed in detail by E.C. Levi in [11].

In the construction of filters based on the transfer function (eq. 2.36) two types of filters can be distinguished:

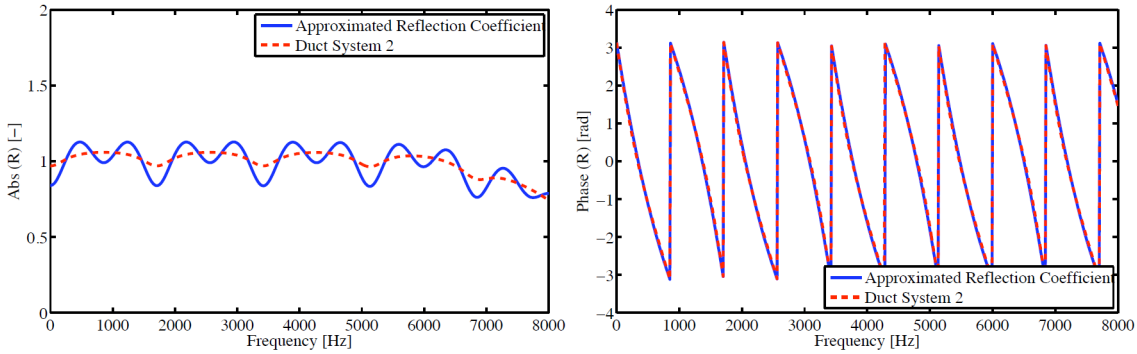


Figure 2.7: Coefficient fit for a short duct [9]

- Infinite Response Filter (IIR): History of both inputs and outputs are considered ($a_i, b_i \neq 0$).
- Finite Response Filter (FIR): Only the history of the inputs are considered ($a_i \neq 0, b_j = 0$).

FIR filters are naturally stable and easier to fit (discussed in section 2.6.3). The IIR filters can be unstable, but need (far) less historical information and are more accurate. The merits of both filters will be investigated for constructing a TD-IBC in chapter 3.

2.6.2 Determination of normalized angular frequency

The argument z of the Z-transform is an important quantity in determining the filter coefficients. The argument z reflects the frequency at which a reflection coefficient R should be enforced in the filter method. The bilateral Z-transform is given by equation 2.43. Where n is an integer, and z a complex number defined by equation 2.44. Because the system must be causal in time, the sum is taken from $n = 0$ to $n = \infty$ in equation 2.43. This is called a unilateral Z-transform.

$$X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n] z^{-n} \tag{2.43}$$

$$z = r \cdot e^{j\omega_n} \tag{2.44}$$

The complex argument ω_n can be seen as a normalized frequency, which runs from 0 to π for the causal discrete time systems in question. The frequency of the sampled wave f is normalized by the sampling frequency f_s ($f_s = \Delta t^{-1}$) (see eq. 2.45). The Nyquist frequency states that the frequencies above 0.5 times the sampling frequency aliases with lower frequencies, and can therefore not be uniquely distinguished. From this fact the upper bound of π (in radians) can be understood, as it corresponds to the maximum frequency that can be uniquely identified on the grid.

$$\omega_n = \frac{f}{f_s} \cdot 2\pi \qquad [^{rad}/_{sample}] \tag{2.45}$$

2.6.3 Stability of filter coefficients

Unfortunately not all combinations of filter coefficients are unconditionally stable. The question of stability is basically a question if all the poles of the transfer function (with which R is approximated) reside within the unit circle in the complex plane (i.e. $|z| \leq 1$). For simple coefficient sets this is often the case, for more complex coefficient sets the problem becomes

more difficult. However, it might not lead to stability problems if the frequencies corresponding to the poles are hardly or not present in the domain. Also, as FIR filters have no poles by definition, they are more stable. Results by Polifke showed that stable results can be obtained via this method with the numerical results within an error margin of around 3% for laminar flow [9].

2.7 Reflection measurement theory

Different methods exist to quantify reflection or impedance of a sample or interface in numerical simulations (but also in experimental set-ups). Among the most used are the Two-Microphone Method (TMM), the Multi-Microphone Method (MMM), and the Standing-Wave Ratio Method (SWR Method). The MMM is the method of choice, because the other two methods have certain disadvantages that make them less suitable for this application. The TMM is simple, but very sensitive to sensor positioning, especially if the microphone separation is near one half wavelength [18][8]. The SWR method is on the other hand very accurate and reliable, but very time-consuming, because a probe with a pressure sensor needs to moved through the tube [18][8]. As *ANSYS CFX* does not support moving monitor points (to measure pressure), and approximating the Standing-Wave Ratio as a function of position might be a cumbersome task, the SWR-Method is at a disadvantage.

The multi-microphone method is discussed briefly in section 2.7.1. All three methods are discussed in more detail in the appendix.

2.7.1 Multi-microphone method

The MMM method calculates the reflection coefficient R (eq. 2.14) by solving a linear system of equations for forward traveling pressure amplitude $\delta\hat{p}^+$ and backward traveling pressure amplitude $\delta\hat{p}^-$, which constitute the definition of R. First the time-series of three or more microphones, a certain distance d from the boundary, must be recorded (see Fig. 2.8 for placement). On these pressure time-series a Fourier transform must be performed to extract the pressure amplitude $\delta\hat{p}$ as a function of frequency at the different locations.

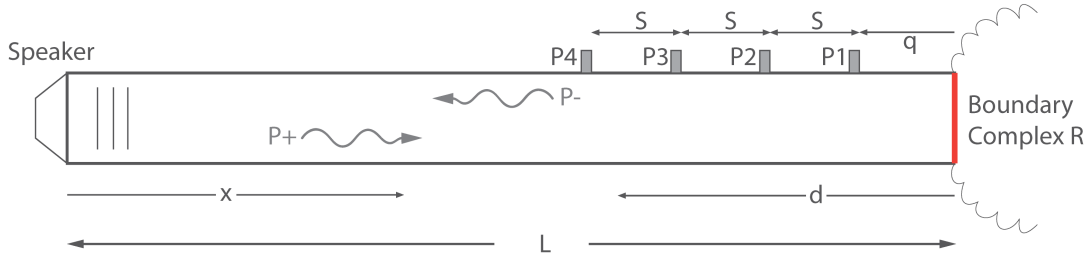


Figure 2.8: Multi Microphone Method in impedance tube

The linear system of equations, that will be constructed later on, is based on the solution of the Helmholtz equation (eq. 2.12). Matters are simplified by introducing a new spatial coordinate d originating from the boundary with the relation $L - d = x$ (see Fig. 2.8), and recognizing that $\delta\hat{p}_+ = A \cdot e^{-ikL}$ and $\delta\hat{p}_- = B(f) \cdot e^{ikL}$ (evaluated at the boundary). This results in the solution of the Helmholtz equation given by equation 2.47.

$$\delta\hat{p}(x, f) = A(f) \cdot e^{-ikL} e^{ikd} + B(f) \cdot e^{ikL} e^{-ikd} \tag{2.46}$$

$$\delta\hat{p}(d_i, f) = \hat{p}_+(f)e^{ikd_i} + \hat{p}_-(f)e^{-ikd_i} \tag{2.47}$$

Using the known pressure amplitude spectra of the microphones an overdetermined linear system of equations of the form $\mathbf{A}\vec{x} = \vec{b}$ can be constructed (eq. 2.48). The known pressure amplitudes $\delta\hat{p}_1$,$\delta\hat{p}_2$,..., $\delta\hat{p}_n$ are in the right-hand side (RHS) vector $\vec{b}$. The coefficient matrix A consists entirely of complex powers of the euler number e. The only unknowns are $\delta\hat{p}_+$ and $\delta\hat{p}_-$ in the $\vec{x}$ vector.

$$\begin{bmatrix} e^{ikd_1} & e^{-ikd_1} \\ e^{ikd_2} & e^{-ikd_2} \\ ... & ... \\ e^{ikd_n} & e^{-ikd_n} \end{bmatrix} \begin{Bmatrix} \hat{p}_+ \\ \hat{p}_- \end{Bmatrix} = \begin{Bmatrix} \hat{p}_1 \\ \hat{p}_2 \\ ... \\ \hat{p}_n \end{Bmatrix} \tag{2.48}$$

The overdetermined system is solved for $\vec{x}$ using matrix inversion. But because $\mathbf{A}$ is not a square matrix, an inverse $\mathbf{A}^{-1}$ of the matrix in the general sense cannot be created. The problem is solved by using a so-called Moor-Penrose pseudo-inverse $\mathbf{A}^{+}$ of $\mathbf{A}$, which is determined using a least-squares method (eq. 2.49).

$$\vec{x} = \mathbf{A}^{+}\vec{b} \tag{2.49}$$

Sensor positioning has an influence on the sensitivity to measurement errors, and is quantified in the *Singularity Factor* (SF). The lower the SF value, the lower the estimation error. Seung-Ho et al showed that the SF is minimal when an equidistant spacing between microphones is used [18]. In this case of equidistant sensor spacing, the optimal ratio of the wave number corresponding to freqeuncy to be measured, over the critical wave number k_{cr}, is 0.5. Using this formula, and the known design frequency, the optimal microphone spacing can be calculated using the formula given in equation 2.50.

$$k_{cr} \cdot \Delta x_{mic} = \pi \cdot (1 - \mathcal{M}^2) \tag{2.50}$$

A more detailed description about the multi-microphone method, Moor-Penrose inverse, attenuation effects, and sensor positioning sensitivity are added to the appendix.

Chapter 3

Method

The TD-IBC must be tested against a well-defined numerical problem. This numerical problem consists of a discretized domain, boundary conditions, initial conditions, fluid models, and other properties such as spatial and temporal integration schemes. Time discretization, space discretization, and boundary conditions are discussed in section 3.1. Fluid models and initial conditions are discussed in section 3.2. The actual implementation of the TD-IBC in *ANSYS CFX* using *FORTRAN* subroutines is discussed in section 3.3. The manner in which the reflection coefficient R is determined is discussed in section 3.4. Lastly an overview and reasoning behind the test cases is discussed in section 3.5, of which the results are discussed in the next chapter, chapter 4.

3.1 Domain model

To eliminate problems not inherent to the implemented TD-IBC, the numerical problem should be defined as simple as possible, but is capable of capturing essential characteristics. To capture the plane waves correctly, and exclude any turbulent, shear, and boundary layer contributions, the domain is modeled one-dimensional. Taking into account the fact that the reflection coefficient must be determined in post-processing, and an optimal configuration would be to use four microphones per wavelength in an equidistant array (see sec. 2.7.1), the domain should be at least one wavelength long. The domain size thus depends on the expected frequencies in the domain, which is predominantly the excitation frequency at the inlet of $f_{in} = 100Hz$, resulting in an expected wavelength of $\lambda_{100} = 3.43m$ (see sec. 3.2). A domain size $L = 4m$ seems a good compromise between accuracy of measurement and domain size (and thus computational effort). The domain size with respect to wavelength is visualized in Figure 3.1.

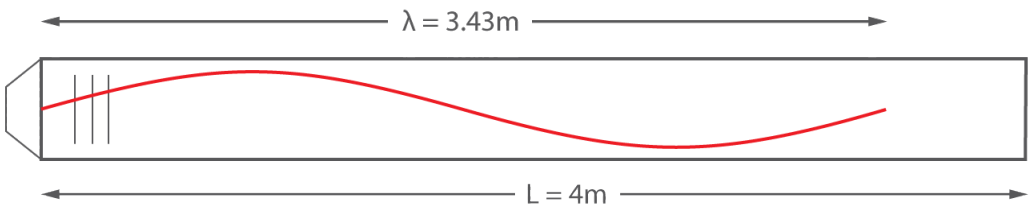


Figure 3.1: Wavelength of excited wave at $f_{in} = 100Hz$ w.r.t. domain size $L = 4m$

The applied boundary conditions are discussed in section 3.1.1, and the space and time discretization is discussed in section 3.1.2.

3.1.1 Boundary conditions

An overview of the applied boundary conditions is given in Figure 3.2. At the inlet a sinusoidally variying velocity of $f_{in} = 100Hz$ is specified. The choice for the excitation frequency is somewhat arbitrary, as spatial and temporal discretization can easily be adjusted to account for different frequencies. However, an excitation freqeuncy of $100Hz$ is also often used in literature by Oosterhuis and Bühler, which aids in the comparison of results [14][3]. The other boundaries at the sides of the domain are *symmetry boundaries*, used to emulate a one-dimensional problem. The solver in *ANSYS CFX* cannot be set to use a one-dimensional set of equations, regardless of domain shape and settings, *ANSYS CFX* always uses a three-dimensional set of equations to solve the problem at hand. The boundary on the right side of the domain is the pressure specified TD-IBC boundary condition.

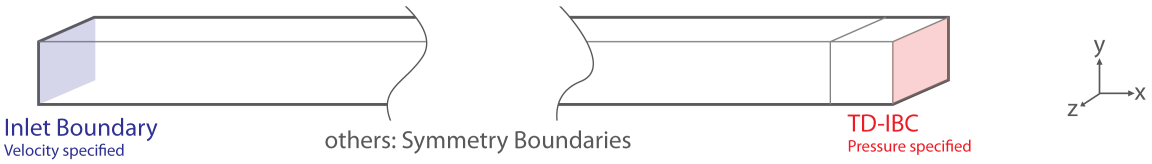


Figure 3.2: Applied boundary conditions in domain model

The sinusoidally varying velocity is specified according to equation 3.1, using Command Expression Language (CEL) in *ANSYS CFX*.[1]

$$u = u_0 + u_a \cdot sin(2\pi ft) \tag{3.1}$$

3.1.2 Space and time discretization

The influence of discretization on the solution of thermoacoustic problems in CFD has been tested by Bühler [3], and is used as a guideline in discretizing the domain. Results showed that 100 elements per wavelength and 100 timesteps per period leads to an error, compared to the analytical solution, smaller than 0.5%. These values are taken as minimum values in determining the element size Δx and timestep Δt.

To determine the exact values of Δx and Δt used in simulations, the Courant-Friedrichs-Lewy number, or simply the *CFL number*, is used. To give an intuitive explanation about what the CFL number actually represents, it can be seen as the ratio between the speed in which information innate to the physical problem travels, over the speed at which information can travel over the numerical grid (see eq. 3.2). If physical information travels faster than the grid can handle, stability and convergence problems can arise. This limit is given by the maximum CFL number CFL_{max}, and depends on the numerical schemes used to solve the governing equations.

For example, if a compact stencil[2] is used to solve the Navier-Stokes equations (or derived fluid dynamics equations), and an explicit first order time marching scheme is used, information is only taken from neighboring points to calculate the next value at the point in question, meaning that information can only travel one element size Δx over one timestep Δt, resulting in a maximum CFL number $CFL_{max} = 1$. Implicit methods are usually unconditionally stable, which means that the schemes used (and thus the CFL number) are not a factor in stability.

[1] See section 3.3.1 for more information on CEL.
[2] A compact stencil is a stencil that includes only directly neighboring nodes

In accuracy however, it is a factor. The solver used by *ANSYS CFX* is implicit, however, the implemented boundary conditions later on are explicit (see sec. 3.3). To rule out any convergence problems due to the CFL number being to high, a maximum CFL number $CFL_{max} = 1$ is chosen.

$$CFL = \frac{u_{phys} \cdot \Delta t}{\Delta x} < CFL_{max} \tag{3.2}$$

Next, the maximum physical speed at which information travels in the governing problem needs to be determined. Mean flow velocity u_0 and the acoustic velocity amplitude $\delta\hat{u}$ will be relatively small compared to the sound speed $c_0 = 346.13^m/_s$ in the test cases defined in section 3.5 (see sec. 3.2). The sound speed c_0 is therefore the limiting physical velocity u_{phys}. Taking an acoustic CFL number $CFL_{acoustic} \leq 1$, and keeping the minimum criteria for the element size and timestep discussed at the beginning of this section in mind, a conforming element size $\Delta x = 4 \cdot 10^{-3} m$ and a timestep $\Delta t = 1 \cdot 10^{-5} s$ is chosen. The minimum and maximum design criteria leave room open to adjust the element size and timestep, therefore rounded numbers are chosen to minimize user error in calculations and derivations, and maximize readability. The resulting discretized domain is visualized in Figure 3.3.

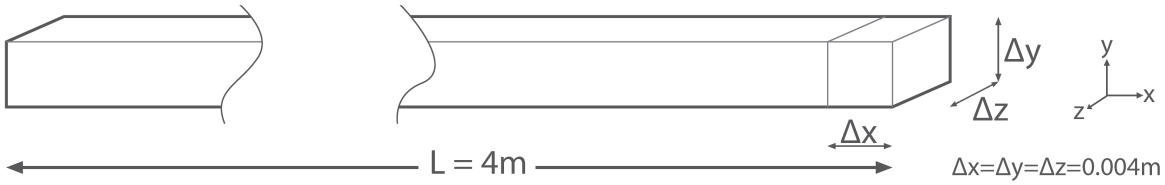


Figure 3.3: Discretization of the domain

For the spatial discretization of the domain, perfectly cubical elements are chosen.[3] All elements of size $\Delta x = 4 \cdot 10^{-3} m$ in each direction fit exactly 1000 times within the domain of $L = 4m$, resulting in perfect mesh quality for different *ICEM* mesh quality checks, such as determinant, volume, angle, and warpage [1]. *ICEM* software is the software used to create the mesh. A timestep of $\Delta t = 1 \cdot 10^{-5} s$ results in 1000 timesteps per wavelength for a frequency of $f_{in} = 100Hz$. The element size $\Delta x = 4 \cdot 10^{-3} m$ results in 857.5 elements per wavelength $\lambda_{100} = 3.43m$. Both well above the set minimum of 100 elements per wavelength set out at the beginning of this section.

[3] A cube is a form of a hexahedron element: a regular hexahedron with all faces square.

The PWM method as well as the TD-IBC need a characteristic wave amplitude f to be determined at a sample plane near the boundary (see sec. 2.5 and 2.6). The sample plane should be far enough from the boundary to prevent feedback caused by the used schemes (stencil), but close enough so that the f determined at the sample plane is very close to the actual f when it hits the boundary (especially amplitude attenuation due to viscous losses). Viscous losses will be neglected in the heat transfer equations because its influence can hardly be measured (see sec. 4.1.1). The sample plane is placed at a distance $x_{pwm} = 0.4m$, or 100 elements, from the boundary, which should be out of reach for any stencil. See Figure 3.4 for placement of the boundary within the domain.

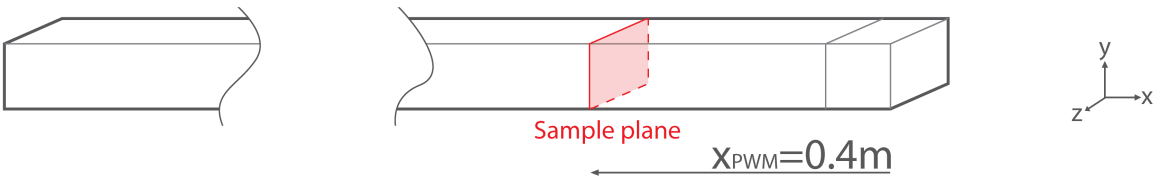


Figure 3.4: Sample plane position within the domain

Preliminary tests showed that increasing time increment Δt, until less than 100 timesteps per period are attained, indeed caused significantly worse results, and even stability issues (i.e. unusable results), as is predicted by Bühler [3]. The hypothesis is that increasing the mesh linearly near the boundary will result in a better reflection coefficient R, as the gradients for pressure and velocity near the boundary might be determined more accurately by *ANSYS CFX*, i.e. the discretization error near the boundary decreases. However, a smaller Δx also increases the CFL number near the boundary, and causes the hexahedron elements to be non-cubical, which might decrease accuracy and cause stability problems as mentioned earlier in this section. See test-cases in section 3.5.4 for the test-cases including linear refinement.

The mesh is linearly refined from a distance of $x_{refine} = 0.4m$ towards the boundary (see Figure 3.5). Instead of 100 elements within this section, now 300 elements are used, totaling 1200 elements for the complete domain. The last element against the boundary has a edge ratio of $1/5$, meaning that the x-length of the element is $\Delta x_{min} = 0.0008m$.

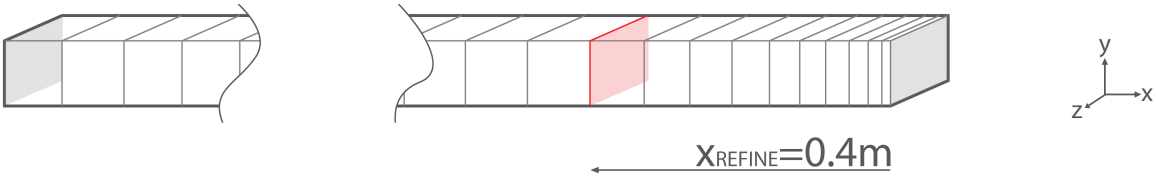


Figure 3.5: Mesh refined near boundary

3.2 Fluid models

The fluid used as a medium in all test cases is air, and its properties vary according to the ideal gas law. No turbulence model is used in the simulations, as it is not a fundamental part in describing acoustic waves, making the flow problem purely laminar.

3.2.1 Initial conditions

The domain is initialized using air at standard pressure and temperature (STP). Unfortunately, STP definitions vary, and to prevent ambiguity, the values of the properties used in initialization are mentioned in Table 3.1.

Table 3.1: Fluid properties Air @ STP

$p_0 = 101325.0Pa$	$\rho_0 = 1.1839^{kg}/_{m^3}$	$c_0 = 346.13^{m}/_{s}$	$T_0 = 298.15K$
$\gamma = 1.4$	$Z_0 = 409.78Rayl$	$C_p = 1007^{J}/_{kg \cdot K}$	$R = 287.058^{J}/_{kg \cdot K}$

The initial velocity is equal to the inlet velocity at initialization (eq. 3.1) to prevent discontinuities at initialization. The inlet velocity at initialization ($t = 0s$) is always equal to the mean velocity u_0 due to a sine function being 0 when the argument is zero. The mean velocity is set per test case, which are defined in section 3.5.

3.2.2 Heat energy equations

There is a distinct difference in isothermal and adiabatic sound speeds. In the isothermal case, heat exchange between the wave and the environment need to take place at moments of rarefaction and compression to keep the temperature constant, which is only accurate when the period of the wave is of the same order or longer than the time it takes to cool the medium. Acoustic waves on earth generally have periods much shorter than it takes to cool the medium, which causes the fluid parcels to compress and expand adiabatically by approximation. In practice this means that the speed of sound is higher in the adiabatic case compared to the isothermal case. Mathematically this results in an adiabatic index $\gamma = 1.4$ in the adiabatic case, and $\gamma = 1.0$ in the isothermal case. The sound speed can be calculated using equation 3.3 (see appendix for derivation).

$$c_0 = \sqrt{\frac{\gamma p_0}{\rho_0}} \tag{3.3}$$

In *ANSYS CFX* calculations can be done both with a heat transfer equation and without (isothermal). Using the heat transfer equation, the acoustic waves will behave adiabatically.

Depending on the heat model that is used, there are consequences for the boundary specification of the temperature. For isothermal calculations the temperature does not vary throughout the domain, and need not at the boundary (i.e. constant temperature boundaries). It is a different story when a heat transfer equation is used. The acoustic waves vary in temperature cyclically through each period. This temperature variation should also apply to the boundary condition to prevent reflections. The temperature can be varied on the boundary by using the adiabatic equation for temperature variation given by equation 3.4, derived from the ideal gas law (eq. 2.4).

$$T_2 = T_1 \left(\frac{P_2}{P_1} \right)^{\frac{\gamma-1}{\gamma}} \tag{3.4}$$

Preliminary tests showed that varying temperature using this adiabatic temperature variation equation (eq. 3.4) resulted in large pressure drift and large reflection coefficients. An explanation could be that even though the acoustic wave behaved adiabatically, in reality always some losses and heat transfer to neighboring fluid parcel occurs. Another explanation could be that something goes wrong in the coupling between boundary values and internally calculated temperatures. Fortunately, the option exists in *ANSYS CFX* to specify an *opening pressure* at initialization, and let the solver do the calculations using conservation laws. The results improved significantly, but does still have some slight problems with pressure drift compared to isothermal calculations (see sec. 4).

Two types of heat transfer equations can be used in *ANSYS CFX* to take into account the varying temperature, "Thermal Energy" and "Total Energy". The thermal energy equations only convects enthalpy h with the flow, while the total energy equations also adds kinetic energy $0.5 \cdot u^2$ (needed for example to take into account stagnation temperatures). The total energy equation also automatically includes the pressure transient term in the set of equations, while for the thermal energy equation it is an option. The total energy heat equation is generally used for high speed flow problems ($\mathcal{M} > 0.3$) [23].

Both heat transfer equations have the options of including *viscous work terms* in *ANSYS CFX*. This option adds the viscous work term to the energy equation, necessary when large viscous shear forces are to be expected. Viscosity is expected to have a very small influence because the domain is small compared to the wavelength and sound speed, and the flow field is laminar (no turbulent viscosity losses). Viscosity causes a decay in amplitude of the acoustic wave over distance traveled. It can also influence the coupling between acoustic pressure δp and acoustic velocity δu, which is assumed in phase in many equations (e.g. in the linearized wave equation, eq. 2.7). The difference is tested in a pragmatic manner by running the same simulation with the viscous work terms both included and excluded, and comparing the solution of the flow field via the reflection coefficient R (see sec. 3.5).

The thermal energy equation also has explicitly the option to include a *pressure transient term* for describing the influence of the time-varying behavior of the pressure on the temperature. In the total energy equation it is automatically included. This pressure transient term is essential for acoustic waves which behave adiabatically (and has a varying temperature). The influence of this term is also tested (see sec. 3.5).

The theoretical amplitude of the resulting pressure wave $\hat{p}$ throughout the domain is different for the isothermal and the thermal energy case because the sound speed differs. The sound speed c_0 can be calculated using equation 3.3, and results in $346.21^m/_s$ and $295.86^m/_s$ for thermal energy and isothermal calculations respectively. The pressure will, as a linear approximation, vary in space as a sine according to equation 3.5.

$$\delta p = \hat{p} \cdot sin(2\pi \frac{x}{\lambda}) \tag{3.5}$$

To verify the calculations in *ANSYS CFX*, it is also important to check if the expected amplitudes of the pressure and velocity gradients match the theoretical values. As the gradients are of crucial importance in the LODI-relations (eq. 2.19). By differentiating eq. 3.5 in x using the chain rule, the resulting expressing for the pressure and velocity gradient can be derived (eq. 3.6 and eq. 3.7).

$$\delta p_x = \hat{p}_x \cdot sin(2\pi \frac{x}{\lambda}) \qquad where \quad \hat{p}_x = \frac{2\pi}{\lambda}\hat{p} \tag{3.6}$$

$$\delta u_x = \hat{u}_x \cdot sin(2\pi \frac{x}{\lambda}) \qquad where \quad \hat{u}_x = \frac{2\pi}{\lambda}\hat{u} \tag{3.7}$$

The properties that pertains to the difference in isothermal and heat transfer equation simulations are summed up in Table 3.2. The wavelength λ is calculated using the well-known formula that relates wavelength, frequency, and sound speed in linear media: $c = f \cdot \lambda$.

Table 3.2: Wave amplitude expressions at the boundaries

	Isothermal	Heat transfer	Unit
f_{in}	100		$[Hz]$
γ	1.0	1.4	$[-]$
c_0	295.86	346.13	$[m/s]$
λ	2.9586	3.4613	$[m]$
u_a	0.5		$[m/s]$
$\hat{p}$	175.1	204.9	$[Pa]$
$\hat{p}_x$	371.86	371.86	$[Pa/m]$
$\hat{u}_x$	1.062	0.91	$[m/s/m]$

3.3 Implementation of a TD-IBC in *ANSYS CFX*

In the first sections some general topics concerning programming in *FORTRAN* for *ANSYS CFX* applied to the non-reflective part (including LRM), the optional PWM part, and the TD-IBC part, is discussed. In section 3.3.7, details specific to the non-reflective part are treated. In section 3.3.8, the same is done for the PWM part, and in section 3.3.9 the TD-IBC implementation is discussed.

3.3.1 Fortran programming in *ANSYS CFX*

There are two methods to create custom behavior in *ANSYS CFX*. The first being a script language called Common Expression Language (CEL). CEL can be used to construct relatively simple mathematical statements which can directly be evaluated, and used as variables, but also certain field quantities and other variables can be retrieved. For example, a vibrating inlet velocity can be specified as a function of time t.

For more complex applications involving solutions to be stored in memory, gradients or time derivatives to be retrieved/calculated from certain locales, output to screen, and complex arithmatic, CEL is not suitable. For these more demanding applications *FORTRAN* subroutines are used. Subroutines work comparable to a normal function, with arguments that can be supplied (both user defined, and fixed solver arguments), and return values that are returned.[4] The custom built *FORTRAN* subroutines are programmed and compiled before the solver runs, and the solver calls them during execution. Compiled subroutines are platform specific, which means that a subroutine compiled for a Linux environment will only run in a Linux environment [23]. Two types of *FORTRAN* subroutines can be distinguished in *ANSYS CFX*:

[4]See *ANSYS CFX* documentation for more information on *FORTRAN* subroutines [23].

1. User (Defined) CEL function: Function that can be called using standard CEL rules. Solver determines when and how often the subroutine is called.

2. Junction Box Routines. Called at certain points during solver execution.

Both type of subroutines are used in the developed boundary conditions. A total of three subroutines are used to construct both boundary conditions (NRBC and TD-IBC), two of which are *junction box* routines, and one a *user CEL* routine. The *user CEL* subroutine is used to return the pressure, and is automatically called by the solver during its solution process. The two *junction box* subroutines are called only at certain stages within the solution process, of which both positions are given in Figure 3.6. The *User Input junction box* is only called once, at the start of the simulation, while the *End of Timestep junction box* is called at the end of every timestep when the solution is converged.

The *PLinit* subroutine at the *junction box* location *User Input* creates data areas and directories, retrieves user parameters, and checks if some important prerequisites for the code to work are met. The *user CEL* function *Pset* sets the pressure at the current locale. To calculate the returned pressure (to the solver), the subroutine retrieves gradient values for pressure and velocity, historical information from memory, and does time integration various quantities. The junction box routine *PTloop* at the junction box location *End of Timestep* calculates the pressure to be set the next timestep and reorders memory, as well as retrieving area averaged quantities to calculate characteristic wave f.

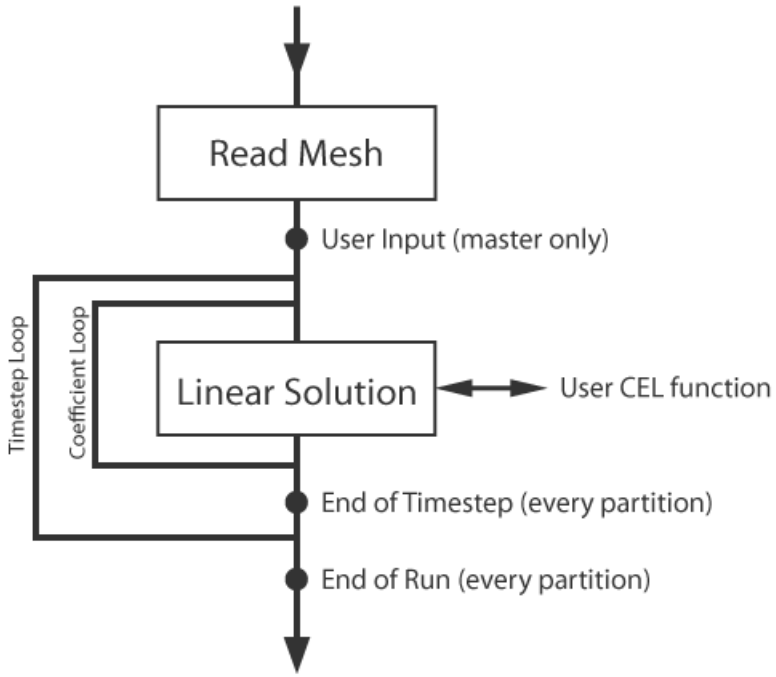


Figure 3.6: *Junction box* and *user CEL* locations of subroutines used.

Note that the graph in Figure 3.6 is not a depiction of all of the available junction box routines in *ANSYS CFX*, which far exceeds the shown, but merely shows junction box locations relevant to the programmed code. See *ANSYS CFX* docomumentation for all *junction box* locations [23].

3.3.2 Parallelization

Often multiple processors are used at once to solve a numerical problem that places stringent demands on computational power within a reasonable time-frame. If the developed boundary conditions are to be used in such a setting in the future, it is important that they are compatible with running on multiple processors. This is however not as straightforward as it might seem, especially memory allocation might create problems when constructing the subroutines in *FORTRAN*.

ANSYS CFX runs on the Single-Program-Multiple-Data (SPMD) model for distributing memory. Before the solver starts with the solution routine, the computational domain is split into many regions. Each region, called locale, is assigned to a different (partition) process and is handled by a certain processing core. One process is called master, and the rest of the processes are called slaves. This distinction between master and slave is important in memory management, because the user-data memory is not shared among processes (see sec. 3.3.4). Regardless of the type of partitioning that is chosen in *ANSYS CFX*, both Message Passing Interface (CHameleon) (MPI(CH)) and Parallel Virtual Machine (PVM) do not result in a shared memory model for user-data.[5]

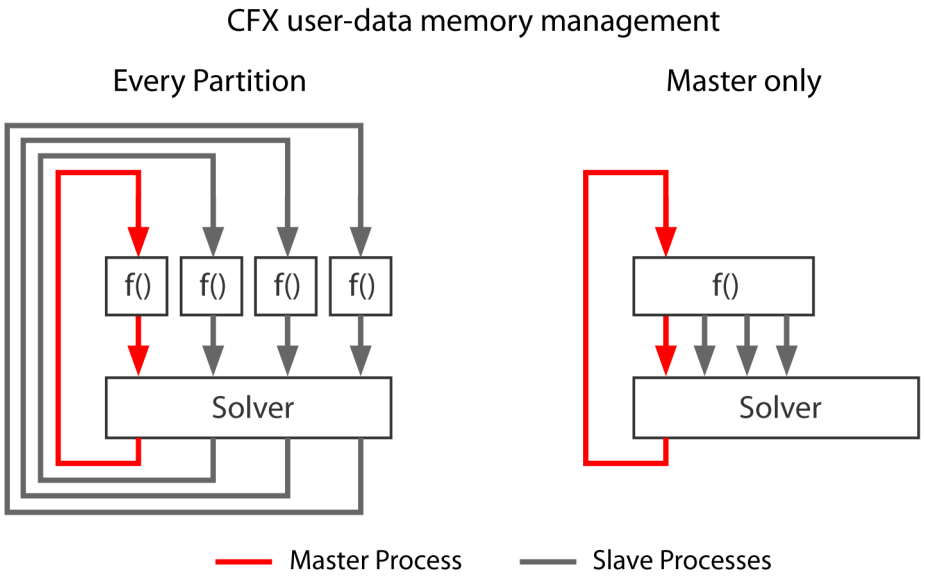


Figure 3.7: Difference "every partition" and "master only" memory handling

Based on the *junction box* location, or user CEL routine, at which the subroutine is called, the solver handles memory management differently. Distinction is made between "Every Partition" and "Master only" memory management, which is visualized in Figure 3.7. "Every partition" means that subroutine called from a partition has its own memory assigned space throughout the simulation. From this subroutine variables set on other partitions cannot be read or changed, and vice versa. "Master only" means that when the subroutine is called, the user-data memory area of the master process overwrites the user-data memory of slave processes. Any change in the user-data memory will only hold until such a "master only" subroutine is called. As can be seen from Figure 3.6, the *PLinit* subroutine at *junction box* location *User Input* is of type "Master only". The *PLinit* subroutine is thus only called once from the master process in a parallel environment, and written data in memory is automatically copied to the slave processes at the end of the routine. the *PTloop* subroutine at *junction box* location *End of Timestep* is of type "Every partition", and is called on every partition once every timestep.

[5]The difference between parallel models are discussed in the*ANSYS CFX* solver modeling guide [23]

The *user CEL* function is always of "Every partition" type. However, this type of memory management does not need to be a problem, but should be taken into account when programming. The subroutine *PTloop* called at *junction box* location *End of Timestep* determines for each partition if any data is stored in its user-data area, and does the calculation for this partition.

3.3.3 Solver data structure

ANSYS CFX structures data and calls *User CEL* subroutines in a way that is not fully documented by the *ANSYS CFX* documentation [23]. *ANSYS CFX* also distinguishes internal solver names from user names. User names often have descriptive names such as "Outlet" for a boundary condition, internal solver names do not. User names can be specified in *ANSYS CFX PRE*, whereas internal solver names are fully controlled by the solver. For example, the user name for a boundary condition could be "Outlet", then the corresponding solver name is "BCPn", which is an abbreviation of *Boundary Condition Patch*, and where n is an integer referring to a boundary condition number. An overview of the different types is given in Table 3.3, where '-' under heading *User* signifies that is only used by the solver and cannot be set in any way in *ANSYS CFX PRE*.

Table 3.3: Ansys CFX model structure, *n*:identifying (natural) number

User-name	Name	Solver-name	Format
Domain	"BODY"	Zone	ZN**n**
Phase	"Fluid 1"	Phase	FL**n**
Boundary Condition	"Outlet"	Boundary Condition Patch	BCP**n**
-	-	Interior Element Group	IELG**n**
-	-	Boundary Element Group	BELG**n**
-	-	Integration Point	IP**n**
-	-	Boundary Integration Point	BIP**n**
-	-	Element Centre	CENTRE**n**
-	-	Boundary Centre	BCENTRE**n**

A boundary condition patch is subdivided multiple in boundary element groups, which comprises of element faces that coincide with the boundary. These element faces each have one face centre (BCENTRE), and, depending on the type of element, multiple boundary integration points (BIP). In the domain model (see sec. 3.1) cubical elements are used with quadrilateral faces. The quadrilateral faces that coincide with the boundary have one BCENTRE and four BIPs, of which the placement is shown in Figure 3.8.

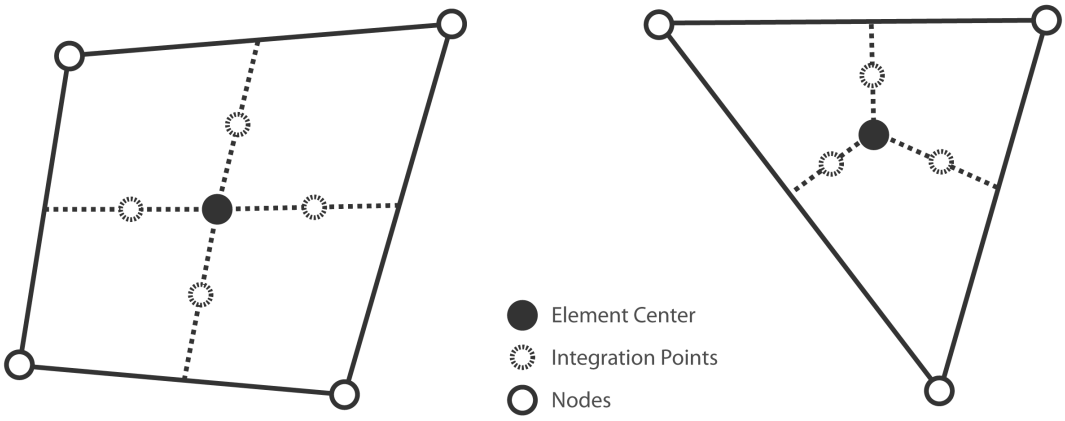


Figure 3.8: Integration points and Element center, left: quadrilateral element, right: triangular element

The *Pset* routine is only called from these BIP and BCENTRE entities from within a boundary element group (BELGn), other entities are not used directly in the developed subroutines. The velocity and pressure gradient are needed to construct the amplitude variation $\mathcal{L}_5$ based on the LODI relations (see sec. 2.3.2). The built-in *ANSYS CFX* boundary condition uses BIPs for retrieval of the pressure gradient, and BCENTREs for retrieval of the velocity gradient, for reasons that are not known publicly [19].

3.3.4 Memory precision

Initial tests with the boundary conditions showed a large amount of pressure drift. An example of the mentioned pressure drift is given by Figure 3.9. Before the excited wave reaches the boundary after $0.012s$, already $40Pa$ pressure drift is observed. After two periods this has accumulated to over $100Pa$ pressure drift. Loss in precision due to multiplication of single precision numbers several orders of magnitude apart seemed to be the cause. The solution is to use double precision numbers. Both the solver needs to be set to double precision, as well as the *FORTRAN* subroutines needs to be coded and compiled as such.

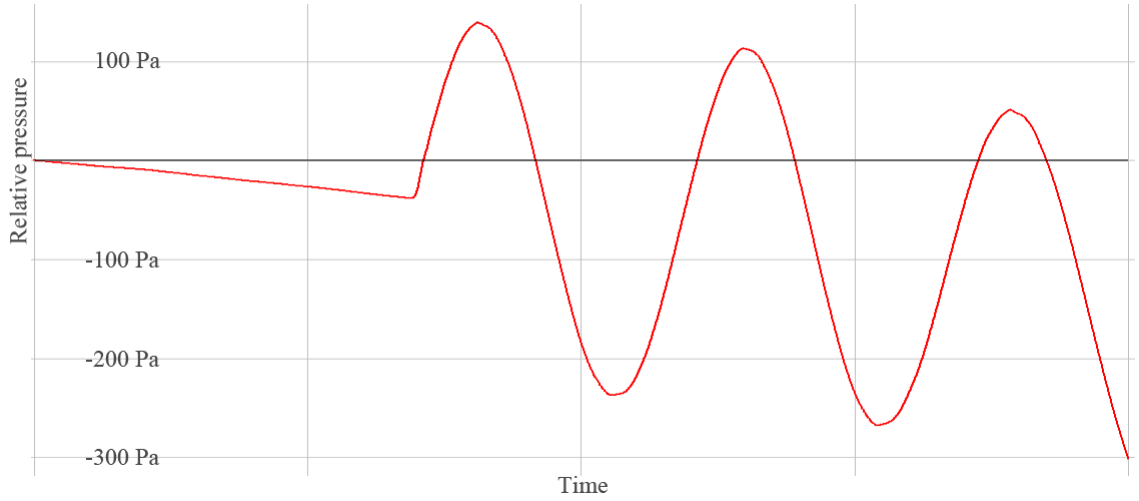


Figure 3.9: Example of pressure drift ($K = 10^1/s$, $\Delta t = 1 \cdot 10^{-5}$)

Depending on the value of the decimal number to be stored, single precision numbers give 6 to 9 significant decimal digits precision, while double precision numbers give 15 to 17 significant decimal digits precision. When multiplying or adding numbers orders of magnitude apart, the error in the resulting number can be substantial. This is often the case when multiplying a time derivative with a very small time increment to retrieve the value at the next time step. Or when adding a small pressure increment onto the current pressure. These type of mathematical operations occur in the *FORTRAN* subroutines written. This results in an error that adds up every timestep, and can cause the pressure to drift significantly, or even become unstable.

More details about single and double precision numbers are added to the appendix.

3.3.5 Memory Management System

To make things easier, *ANSYS* created a memory structure that can be accessed using a "directory" type of hierarchical structure called Memory Management System (MMS). This keeps data ordered and easy to manipulate. However, on a low level all data of a certain type (e.g. integer, real, etc) is placed on a stack and managed by the memory management of the host system. The global stacks on which the pressure information is stored can be read and written to using two ways, the first by calling a so-called *peek* or *poke* routine with the MMS path as argument, the second by directly accessing the stacks. A *peek* routine retrieves values from the stack, while a *poke* routine stores values on the stack. However, for double precision floating point numbers the *peek* and *poke* routines are not defined, and not in any of the includes of *ANSYS CFX*. The stacks need to be directly addressed to access and store values.

To alter data on the stacks, first a pointer to the data area in question needs to be retrieved. A pointer is an integer data type memory address to the first value of the data structure it points to. By using a MMS function called *LOCDAT*, and the MMS path as argument, a pointer can be retrieved. Multi dimensional arrays are stored as a one-dimensional vector in memory, with all memory positions reserved in one consecutive data area. For a 2D matrix A of size $V \times W$, each cell $A(I, J)$ can be accessed on stack DZ using pointer pPOINT (see eq. 3.8). This is done in a row for row fashion. For example, for a 4×3 matrix A cell $A(3, 2)$ can be accessed on the stack via eq. 3.9. Pointers are, by convention, written starting with a small p.

$$A(I, J) = DZ(pPOINT + W * (I - 1) + (J - 1)) \tag{3.8}$$

$$A(3, 2) = DZ(pPOINT + 3 * (3 - 1) + (2 - 1)) \tag{3.9}$$

The complete directory structure used to store variables and arrays is shown in Figure 3.10. The *USER* data area is used to store *user parameters* and values returned by various *utility routines* defined by *ANSYS CFX*. *User parameters* are variables that are read in from a text-file at initialization, and stored in memory. For the developed subroutines the *user parameters* are used to import values such as the relaxation factor K, and the target pressure at infinity p_∞. *Utility routines* are built-in routines from *ANSYS CFX*, for example to get area integrated quantities of field variables. The *USER_ DATA* data area is used to store all variables that are specific to the programmed code.

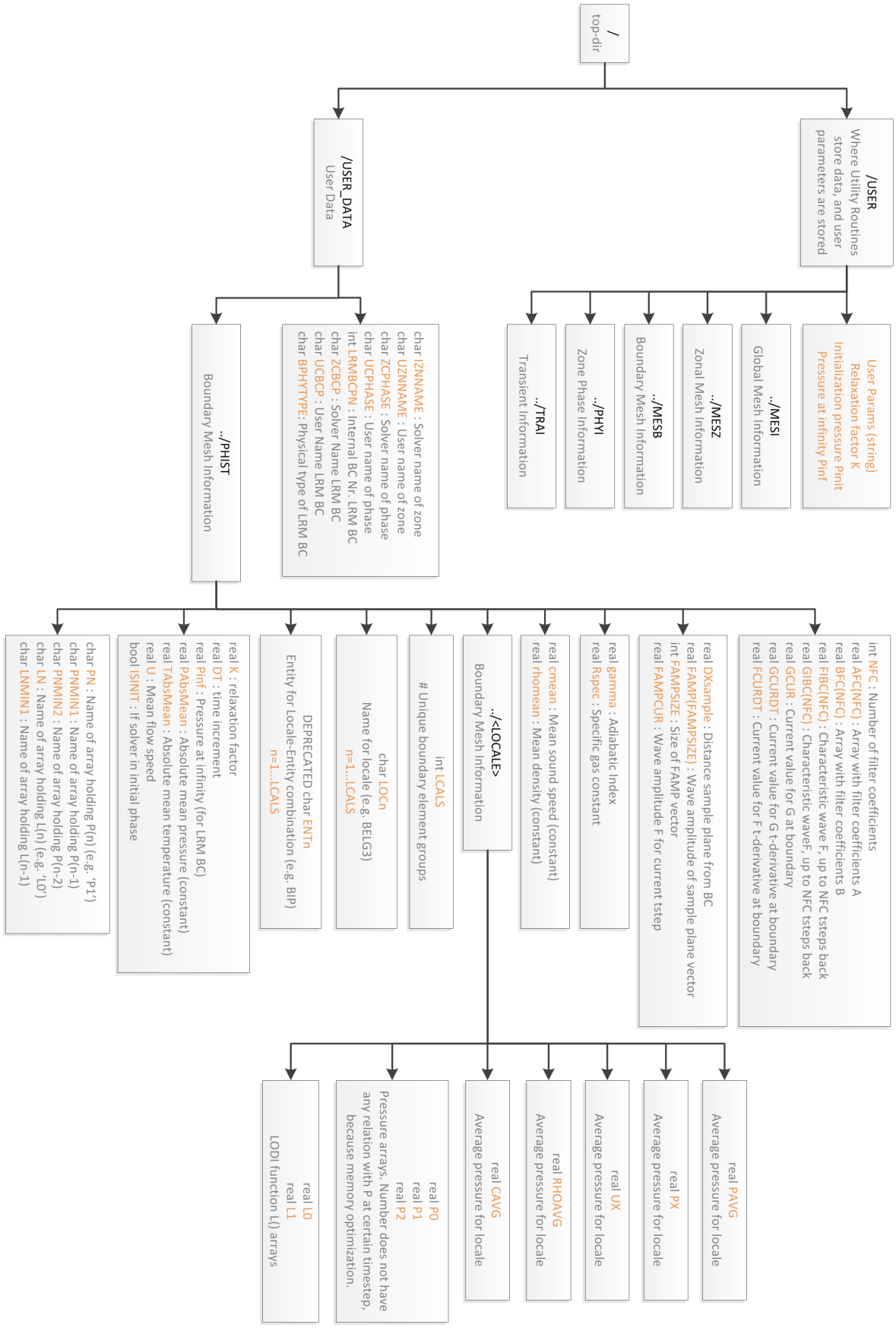
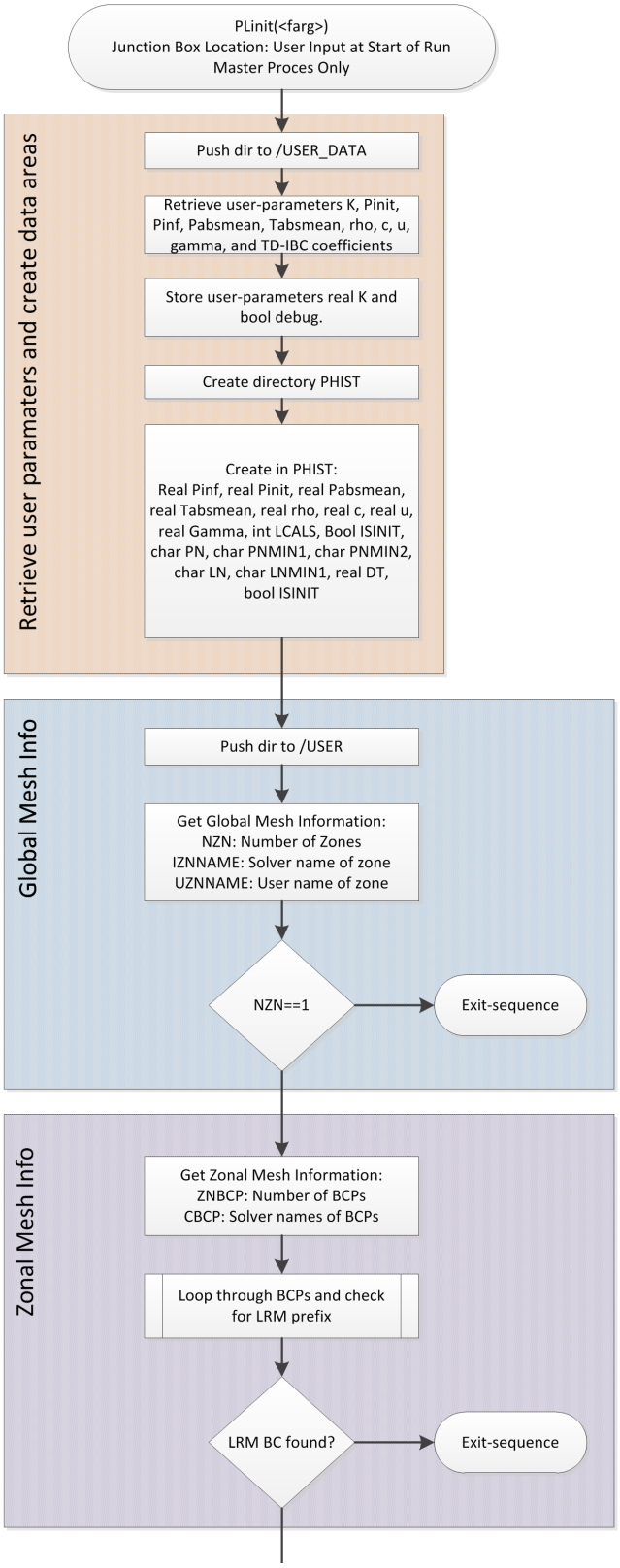


Figure 3.10: Memory structure used in subroutines

3.3.6 Initialization routine

The *PLinit* routine, of which the flowchart is shown in Figure 3.11, is equal for both the non-relflective (with or without PWM) and the TD-IBC implementation, except for a few added *user parameters* in the case of the TD-IBC. In the first section of the code the *user parameters* are retrieved and stored in the MMS. Also other important variables and directories are created. The second section retrieves global mesh information, such as the number and names of zones in the problem definition. These are needed for certain *utility routines* in the other two developed subroutines *PTloop* and *Pset*. If there is more than one zone, the exit-sequence is called. More than one zone means that multiple domains are defined, which is not supported. The exit-sequence stops the code, and signals to the solver that it needs to stop the simulation. In the zonal mesh info section the number and names of BCPs are retrieved, and checked for the "LRM" prefix needed to mark which boundary is designated to function as a non-reflective or TD-IBC boundary. If no boundary is found, the exit-sequence is called. The LRM BCP mesh info section checks which physical type ("OUTLET", "WALL", etc) the designated LRM-boundary has. Only "INLET", "OUTLET", and "OPENING" are supported, or else the exit-sequence is called. The next section is the Zonal Phase Info section that checks if only one phase is present, as multi-phase flow is not supported in the code. If multi-phase flow is present, the exit-sequence is called. The last section stores information retrieved regarding the defined problem, such as domain names, BCPs, and Phase names.


PLinit(<farg>)
Junction Box Location: User Input at Start of Run
Master Proces Only
Retrieve user paramaters and create data areas
Push dir to /USER_DATA
Retrieve user-parameters K, Pinit, Pinf, Pabsmean, Tabsmean, rho, c, u, gamma, and TD-IBC coefficients
Store user-parameters real K and bool debug.
Create directory PHIST
Create in PHIST:
Real Pinf, real Pinit, real Pabsmean, real Tabsmean, real rho, real c, real u, real Gamma, int LCALS, Bool ISINIT, char PN, char PNMIN1, char PNMIN2, char LN, char LNMIN1, real DT, bool ISINIT
Global Mesh Info
Push dir to /USER
Get Global Mesh Information:
NZN: Number of Zones
IZNNAME: Solver name of zone
UZNNAME: User name of zone
NZN==1
Exit-sequence
Zonal Mesh Info
Get Zonal Mesh Information:
ZNBCP: Number of BCPs
CBCP: Solver names of BCPs
Loop through BCPs and check for LRM prefix
LRM BC found?
Exit-sequence

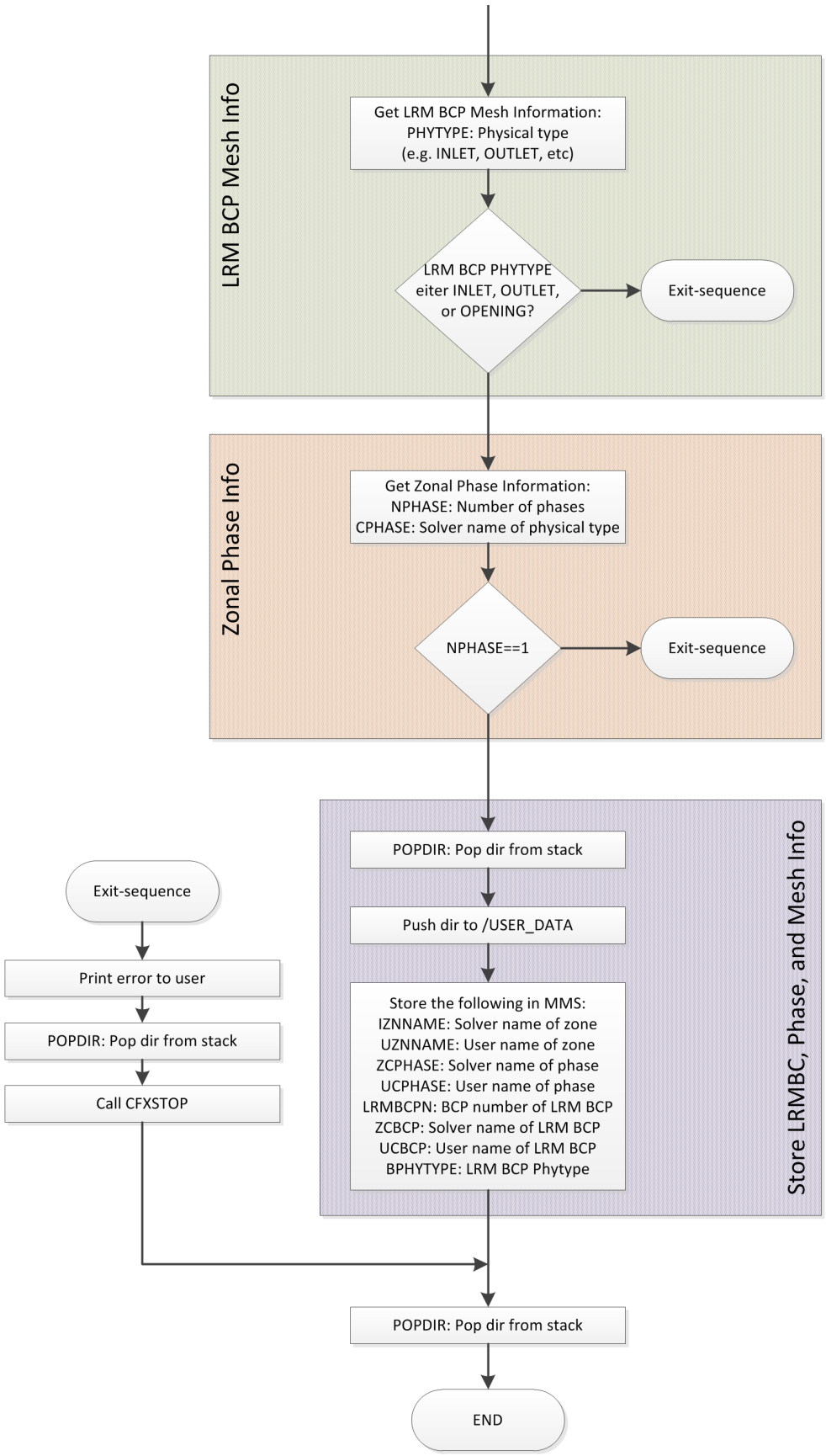


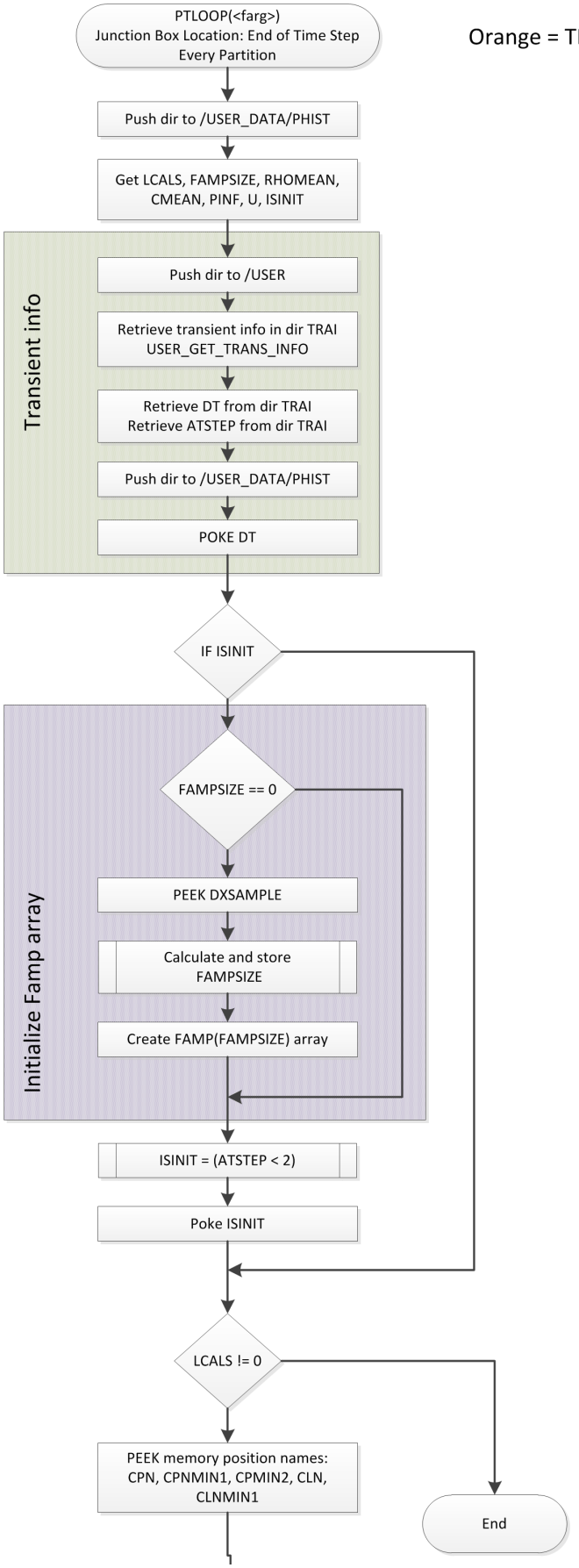
Figure 3.11: Simplified PLinit subroutine flowchart

3.3.7 Non-reflective part

The non-reflective part uses the LODI method to construct amplitude variation $\mathcal{L}_5$, which is distinct from the TD-IBC part that uses the characteristic wave f retrieved from a sample plane near the boundary (see sec. 2.3.2). The *PTloop* and *Pset* subroutines are more distinct for both the non-reflective and TD-IBC boundary conditions. The flowchart representing the *PTloop* routine is given by Figure 3.12. The orange part is the TD-IBC specific part, the rest is equal for both boundary conditions. The flowchart in Figure 3.13 is the *Pset* subroutine for the non-reflective boundary condition. The LRM and PWM methods are both implemented in the non-reflective boundary condition, but can be used or neglected based on the *user parameters* given. By specifying a relaxation factor $K = 0^1/_s$, the LRM method is effectively not used. The inclusion of the PWM term is controlled by the PWM flag in the *user parameters*, which can be set to 0 (off) or 1 (on).

3.3.7.0.1 *PTloop* subroutine description The *PTloop* subroutine described by the flowchart in Figure 3.12 is called at the end of each timestep on every partition. First, transient data such as the timestep and time increment Δt are retrieved and stored, then it is checked if the subroutine is in its initial phase (meaning first timestep in the simulation). If the subroutine is in initial phase, an array that holds data concerning the characteristic wave f sampled on the sample plan is constructed. How this array is constructed is discussed in section 3.3.8.2. Then the *ISINIT* boolean flag is set to false, so the arrays in question are not constructed again within this partition. As data is stored in memory according to the SPMD model (each partition isolated), it is checked (using *LCALS*) if the current partition is tied to the BCP in question (and thus holds any relevant data). If it is not tied to the LRM BCP, the routine exists and the solver continues. Next a memory optimization routine is performed, of which the details are explained in section 3.3.7.2. In the "Calculate new FAMP" section, the characteristic wave f is calculated using area average pressure and velocity at a sample plane near the boundary. (see sec. 3.3.8.1 for more details). The last section is TD-IBC specific, and is explained in section 3.3.9.

3.3.7.0.2 *Pset* subroutine description The non-reflective version of subroutine *Pset* is more distinct from the TD-IBC version than for the *PTloop* subroutine, because of the different manner in which amplitude variation $\mathcal{L}_5$ is determined. The flowchart of the non-reflective version of the *Pset* subroutine is shown in Figure 3.13. First locale and entity information is gathered (see sec. 3.3.3), and certain other important quantities such as the relaxation factor K and the time increment Δt are retrieved. If the solver is in its initial phase, arrays containing historical information about the gradients and pressure are constructed, and given initial values. If the subroutine is not in initial phase, then the gradients are retrieved from *utility routine user_getvar*, the amplitude variations $\mathcal{L}_1$ and $\mathcal{L}_5$ calculated using the LODI relations (see sec. 2.3.2 and 2.4), and the new pressure for the current timestep is determined. Lastly this new pressure is returned to the solver to be set at the boundary.


PTLOOP(<farg>)
Junction Box Location: End of Time Step
Every Partition
Orange = TD-IBC specific
Push dir to /USER_DATA/PHIST
Get LCALS, FAMPSIZE, RHOMEAN, CMEAN, PINF, U, ISINIT
Transient info
Push dir to /USER
Retrieve transient info in dir TRAI USER_GET_TRANS_INFO
Retrieve DT from dir TRAI
Retrieve ATSTEP from dir TRAI
Push dir to /USER_DATA/PHIST
POKE DT
IF ISINIT
Initialize Famp array
FAMPSIZE == 0
PEEK DXSAMPLE
Calculate and store FAMPSIZE
Create FAMP(FAMPSIZE) array
ISINIT = (ATSTEP < 2)
Poke ISINIT
LCALS != 0
PEEK memory position names: CPN, CPNMIN1, CPMIN2, CLN, CLNMIN1
End

.

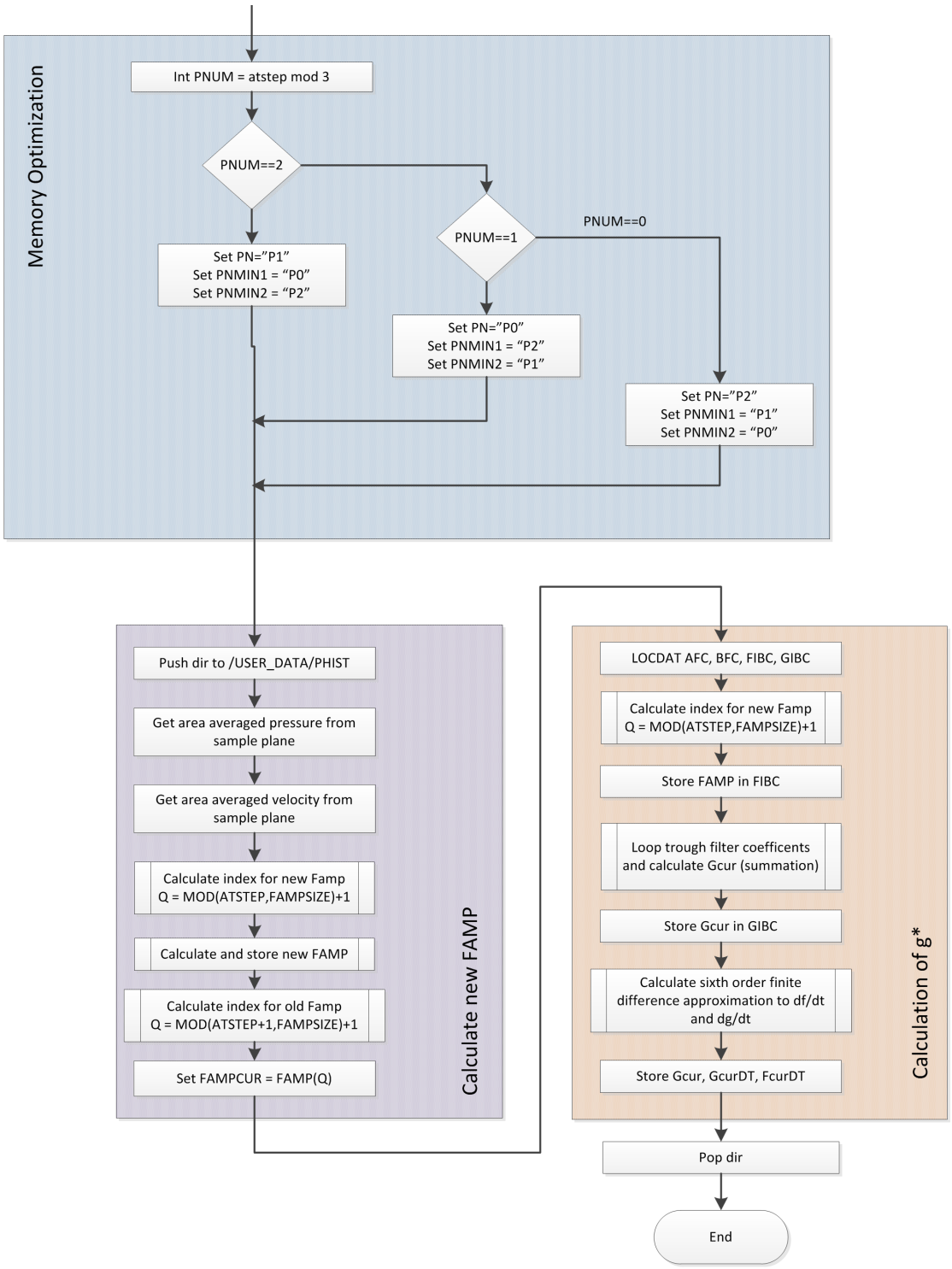


Figure 3.12: Simplified PTloop subroutine flowchart

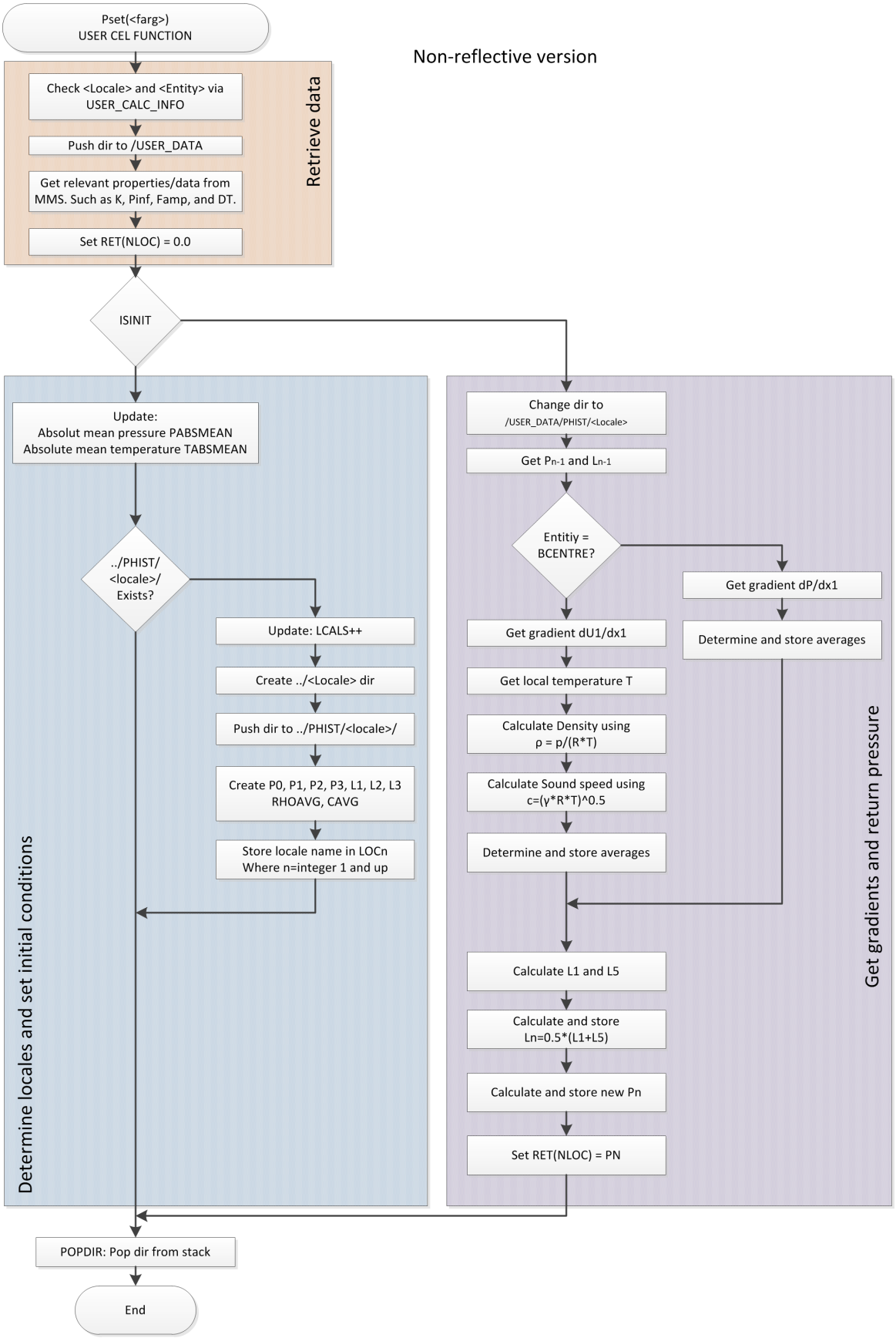


Figure 3.13: Simplified Pset subroutine flowchart (NRBC version)

3.3.7.1 Time integration

As only temporal derivatives of the pressure are given by the LODI relations (see eq. 3.10), the pressure derivative must first be integrated using a numerical time integration scheme to determine a pressure value to be set at the next timestep.

$$\frac{\partial p}{\partial t} = L \qquad where \quad L = -\frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1) \tag{3.10}$$

$$p^{n+1} = p^n + \int_t^{t+\Delta t} L \quad dt \tag{3.11}$$

Different time integration schemes can be used to approximate the integral in (3.11), the simplest being explicit forward (eq. 3.12) and implicit backward Euler (eq. 3.13).

$$p^{n+1} = p^n + \Delta t \cdot L(n) \tag{3.12}$$

$$p^{n+1} = p^n + \Delta t \cdot L(n+1) \tag{3.13}$$

The Euler method is only first order in time, and the accuracy and stability might benefit from higher order schemes. Well-known integration schemes such as Runge-Kutta will not work, because one can use a Euler-Forward predictor to predict $p(n+\frac{1}{2})$, but is not able to evaluate L at future timesteps, because L does not depend on t and p explicitly. However, for more accuracy (lower truncation error), one can use higher order backward (finite) differences to approximate the time derivative, and a predictor of L at $n+\frac{1}{2}$.

$$p^{n+1} = \frac{4}{3}p^n - \frac{1}{3}p^{n-1} + \frac{2}{3}\Delta t \cdot L^*(n+\frac{1}{2}) + \mathcal{O}(\Delta t^2) \tag{3.14}$$

$$p^{n+1} = \frac{18}{11}p^n - \frac{9}{11}p^{n-1} + \frac{2}{11}p^{n-2} + \frac{6}{11}\Delta t \cdot L^*(n+\frac{1}{2}) + \mathcal{O}(\Delta t^3) \tag{3.15}$$

The time integration schemes given by equation 3.14 and 3.15 are second order and third order respectively. These higher order integration schemes also require more historical information, and thus higher memory requirements. The $L^*(n+\frac{1}{2})$ term is a predictor for the LODI function at $n+\frac{1}{2}$, created by using either a (explicit) Lagrange polynomial extrapolation (see eq. 3.16), or by using an implicit predictor by taking the average left and right of $n+\frac{1}{2}$ (see eq. 3.17).

$$L^*(n+\frac{1}{2}) = \frac{3}{2}L(n) - \frac{1}{2}L(n-1) \tag{3.16}$$

$$L^*(n+\frac{1}{2}) = \frac{1}{2}(L(n) + L(n+1)) \tag{3.17}$$

Using the implicit predictor for $L^*(n+\frac{1}{2})$ (see eq. 3.17) results in a *Crank-Nicolson* scheme. The caveat is how to evaluate L at $n+1$, which is not known yet at the beginning of the timestep. By using a predictor for $L(n+1)$ at the beginning of the timestep and first coefficient loop iteration (for example, constant extrapolation: $L(n) = L(n+1)$), the loop can be started and at the second loop the value for $L(n+1)$ can be calculated on the boundary. This is a semi-implicit scheme which uses the latest information available.

3.3.7.2 Memory optimization

As can be seen in section 3.3.7.1, more historical information is needed at the boundary for higher order time integration. For a third order approximation of the pressure derivative, pressure data is needed at n, $n-1$, and $n-2$, and for function L it is $L(n)$ and $L(n-1)$. This will most probably causes no big increase in memory requirements, because the number of nodes at the boundary increases by an order of 2, while the number of nodes in the interior increases by an order of 3. And when the domain is very small, memory requirements are usually not an issue. Not only memory requirements increases, but also memory operations if the data is just shifted over each timestep loop. The number of memory operations can be reduced by using a modulo operation (see Fig. 3.14).

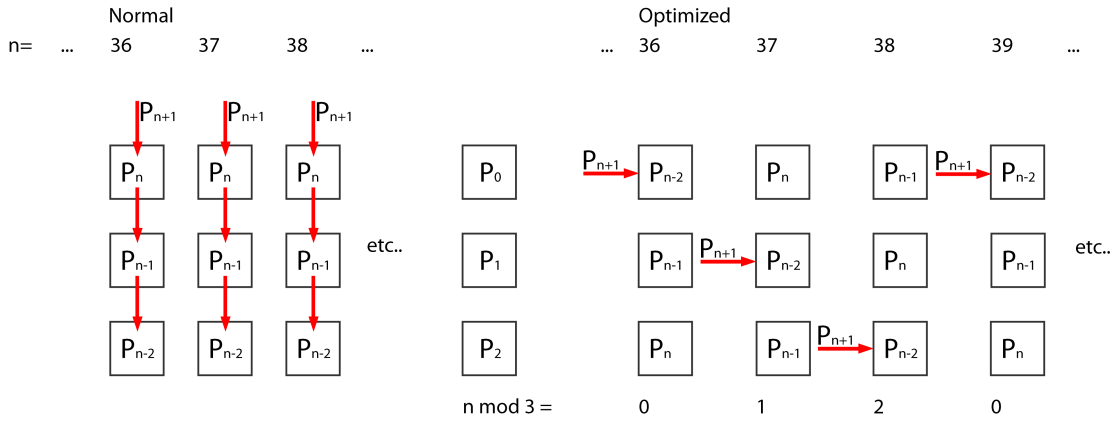


Figure 3.14: Optimized memory storage

At each timestep a new pressure needs to be calculated, but the pressure at $n-1$ in memory is not $n-1$ anymore (as was the previous timestep), so the value must be moved to $n-2$, and $n-2$ to $n-3$, and so on. For a third order scheme three memory operations must be performed per node to store the new value. Each arrow in the Figure (3.14) represents an operation. By using a new way of managing the memory locations, this can be reduced to just one. By taking the modulo of 3 (three locations) for timestep n, the correct memory location (0, 1, or 2) for the storage of the new pressure can be determined at the beginning of each timestep.

3.3.8 Plane-wave masking

The plane-wave masking method is used to mask the characteristic wave f from the LRM term (eq. 2.25), in this way deviations from the target pressure p_∞ resulting from acoustic contributions are not corrected, and thus do not cause reflective behavior (see sec. 2.5). To construct characteristic wave f, the area averaged pressure and velocity must be retrieved at a sample plane near the non-reflective boundary. How these quantities are retrieved is explained in section 3.3.8.1. Next, the calculated incoming characteristic wave f (see eq. 2.31) must be stored for at least the time delay until the wave hits the boundary, which is elaborated upon in section 3.3.8.2.

3.3.8.1 Retrieval of pressure and velocity from sample plane

To evaluate the characteristic wave f, the area averaged pressure $\langle p \rangle$ and velocity $\langle u \rangle$ on a sample plane near the boundary are needed. The *utility routine user_get_gvar* can retrieve area averaged field quantities on any user defined location. User locations can include 2D regions

(e.g. boundaries) and 3D regions (e.g. domains). However, one cannot just create an arbitrary 2D region within the mesh. The mesh needs a face to work on, and this cannot be an internal element face. The approach is to create two blocks in a mesh editor (e.g. ICEM), and create a named part from the interface between them (see Fig. 3.15). This interface (shown in red) is used as the sample plane. In *ANSYS CFX* a 1:1 general connection interface needs to be constructed, which automatically creates two boundaries. These are now two locations (on exactly the same location) from which the area averaged field quantities can be retrieved. Also the mean density and sound speed are needed, but these remain constant during the simulation and are evaluated at initialization.

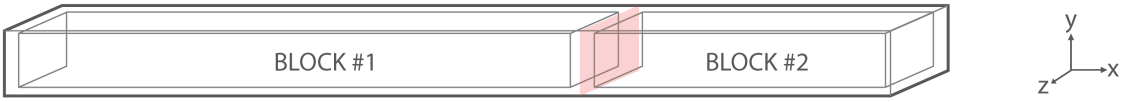


Figure 3.15: Plane-wave masking sample plane construction using blocks in *ICEM*

3.3.8.2 Storage of integrated quantities

The evaluated characteristic wave f at the sample plane must be stored in memory until the wave reaches the boundary, where it is used in the specification of $\mathcal{L}_1$ (see eq. 2.25). The sample plane is positioned at a distance x_{pwm} from the boundary (see Fig. 3.15). The corresponding time delay $t_{pwmdelay}$ is calculated using eq. 3.18.

$$t_{pwmdelay} = \frac{x_{pwm}}{c_0} \tag{3.18}$$

The characteristic wave f can only be retrieved at discrete time intervals, and is stored in a vector of length n_f, where n_f is calculated by dividing the time delay $t_{pwmdelay}$ by the time increment Δt, and adding 1 (see eq. 3.19). If one would choose a fixed position in the vector for the newest and oldest value (the oldest being the one that should be retrieved to calculate $\mathcal{L}_1$), all data must shift one position every timestep, resulting in n_f read and write operations. By using a modulo operation the index of the newest and oldest data shifts one cell (see eq. 3.20). This results in just one read and write operation per timestep (see Fig. 3.16).

$$n_f = lower\left(\frac{t_{pwmdelay}}{\Delta t} + 1\right) \tag{3.19}$$

$$Q_1 = MOD(t_{step}, n_f) + 1 \qquad and \qquad Q_2 = MOD(t_{step} + 1, n_f) + 1 \tag{3.20}$$

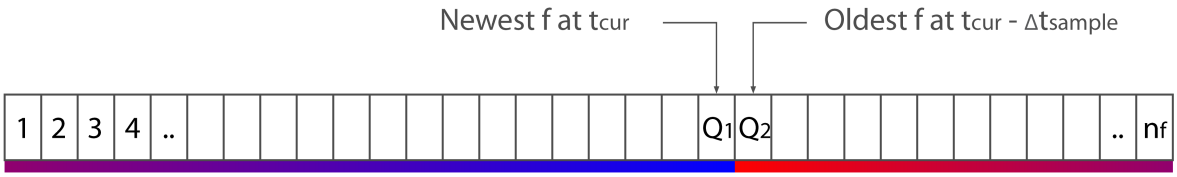


Figure 3.16: Plane-Wave Masking memory storage

3.3.9 Time-domain impedance boundary condition part

The TD-IBC part is constructed by adding the temporal derivative of an external perturbation g^* to the amplitude variation $\mathcal{L}_5$, which is determined through filter-methods and historical information about f and g^* (see sec. 2.6). How the filter coefficients are determined is discussed in section 3.3.9.1. Next, an alternate formulation for $\mathcal{L}_1$ and $\mathcal{L}_5$ is discussed in section 3.3.9.2, and in section 3.3.9.3) the storage of quantities and the calculation of the first time derivative of g^* and f is discussed. In the last section (sec. 3.3.9.4) the TD-IBC specific *user CEL routine Pset* is explained.

3.3.9.1 Calculation of filter coefficients

MATLAB includes a function that calculates the filter coefficients (see sec. 2.6.1) based on the method by E.C. Levy [11]. This function is called *invfreqz*, and returns the coefficients of the numerator and denominator of the rational expression for R in equation 2.36. The function takes the complex reflection R at different normalized frequencies ω_n, and the order of the numerator and denominator as arguments. A script is written that takes the frequency range f, reflection coefficient R, and the timestep Δt as input, and calculates the order n and normalized frequency ω_n as input for the *invfreqz* function (see sec. 3.4). The script is based on the assumption that the complex reflection coefficient stays constant with the given frequency range. Based on this, the order can be calculated by means of the largest time delay needed (see eq. 2.41 and 2.42), which belong to the lowest frequency.

To assess if the filter coefficients will be stable, a pole-zero map is used (see Fig. 3.17 for an example pole-zero map). Any pole (marked: $\times$) outside the unit circle in the complex plane might cause unstable behavior. Preliminary simulations indeed show that filter coefficient sets with poles outside the unit-circle indeed become unstable. Iterative runs, and slight adjusting of the reflection coefficient and number of coefficients, might result in better positioning of the poles, and is discussed in section 3.5.5.

3.3.9.2 Altered LODI formulation

The locally reacting nature within the definition of $\mathcal{L}_5$, as given by equation 2.20 gives rise to problems in combination with the modified expression (with external perturbation) for $\mathcal{L}_1$ as given by equation 2.32. Because the $\frac{\partial g^*}{\partial t}$ term in the modified $\mathcal{L}_1$ expression changes the pressure and velocity gradients such that the incoming f wave cannot be uniquely and correctly distinguished anymore. The problem can be overcome by giving up some properties of the LODI relations. The characteristic wave f is no longer determined implicitly by boundary gradients, but taken from the sample plane and offset in time towards the boundary. By doing this it is assumed that between the sample plane and the boundary, the characteristic wave f is a pure 1D plane wave, does not exhibit dispersion behavior, has a constant sound speed, and does not decay in amplitude. The new expression for $\mathcal{L}_5$ is given by equation 3.21.

$$\mathcal{L}_5 = -\frac{\partial f}{\partial t} \tag{3.21}$$

3.3.9.3 Storage and time differentials

The forward travelling wave f is sampled at a sample plane near the boundary. As plane-wave masking is also used for the TD-IBC implementation using filter methods, the same time delayed value of f can be used. The value of f when it reaches the boundary is retrieved from

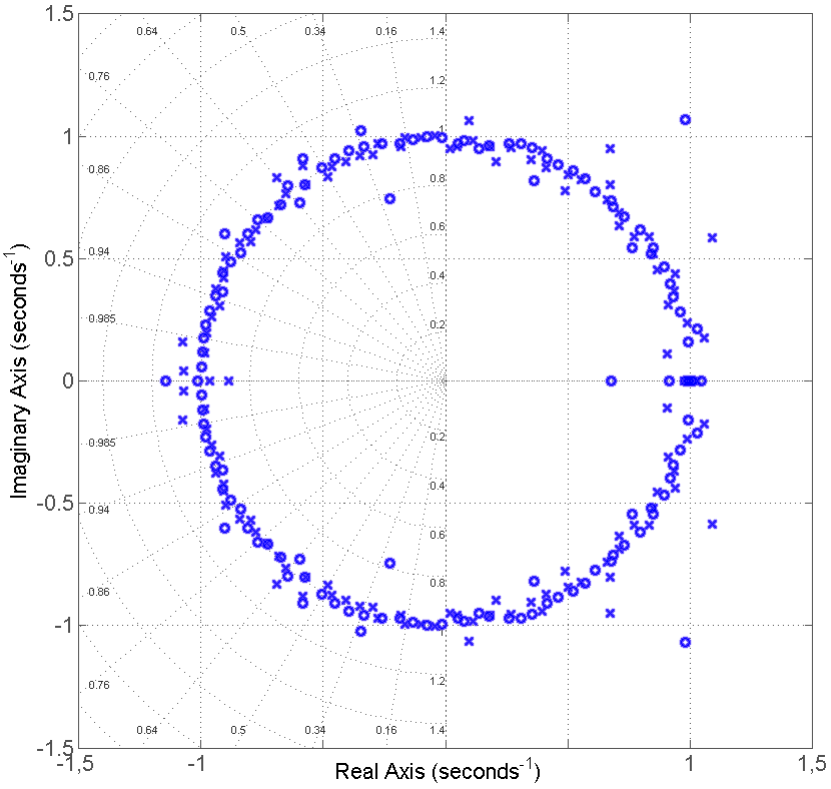


Figure 3.17: Pole-Zero map for $R = 0.5 + 0.5i$, o:zero, x:pole

the plane wave masking array, and stored in the TD-IBC array for f (see Fig. 3.18). Also g^* needs to be stored, but is calculated directly at the boundary using equation 2.40. The size of both arrays for f and g^* are equal to the number of filter coefficients n used.

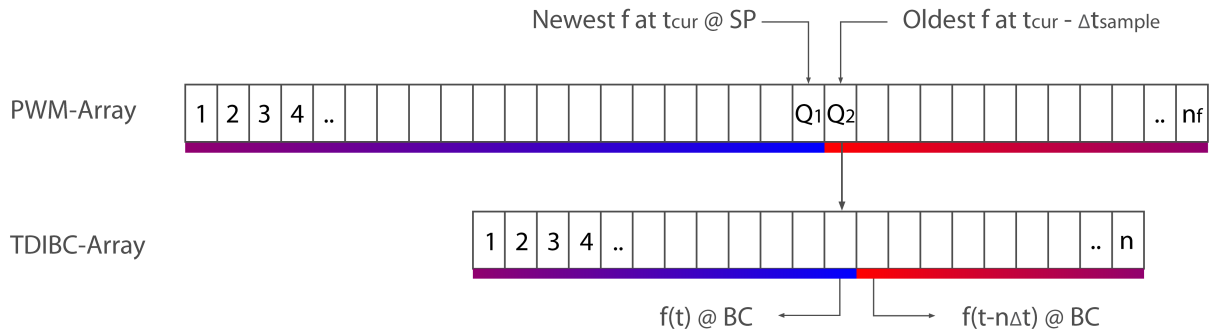


Figure 3.18: TD-IBC Masking memory storage

As can be seen from the expressions for $\mathcal{L}_1$ and $\mathcal{L}_5$ (see eq. 2.32 and 3.21), the time derivatives of both f and g are needed. These are approximated using one-sided sixth order finite difference expressions (see eq. 3.22). Which demands historical information of six timesteps back.

$$\frac{\partial f}{\partial t} = \frac{1}{\Delta t}\left(\frac{49}{20}f_t - 6f_{t-\Delta t} + \frac{15}{2}f_{t-2\Delta t} - \frac{20}{3}f_{t-3\Delta t} + \frac{15}{4}f_{t-4\Delta t} - \frac{6}{5}f_{t-5\Delta t} + \frac{1}{6}f_{t-6\Delta t}\right) \quad (3.22)$$

3.3.9.4 User CEL routine *Pset* for TD-IBC

The *Pset* subroutine for the TD-IBC differs from the NRBC version mainly in the manner in which the amplitude variation $\mathcal{L}_5$ is determined (see sec. 3.3.9.2). The flowchart of the TD-IBC version of the *Pset* subroutine is given in Figure 3.19. In the purple section of the flowchart the forward and backward traveling characteristics f and g are retrieved (which were calculated after each timestep in the *PTloop* subroutine using area averaged integrated quantities). From these quantities the temporal derivatives are determined, which are used in the definition of $\mathcal{L}_1$ and $\mathcal{L}_5$. The new pressure of the current timestep can be calculated using equation 3.10 and 3.11, which is subsequently returned to the solver.

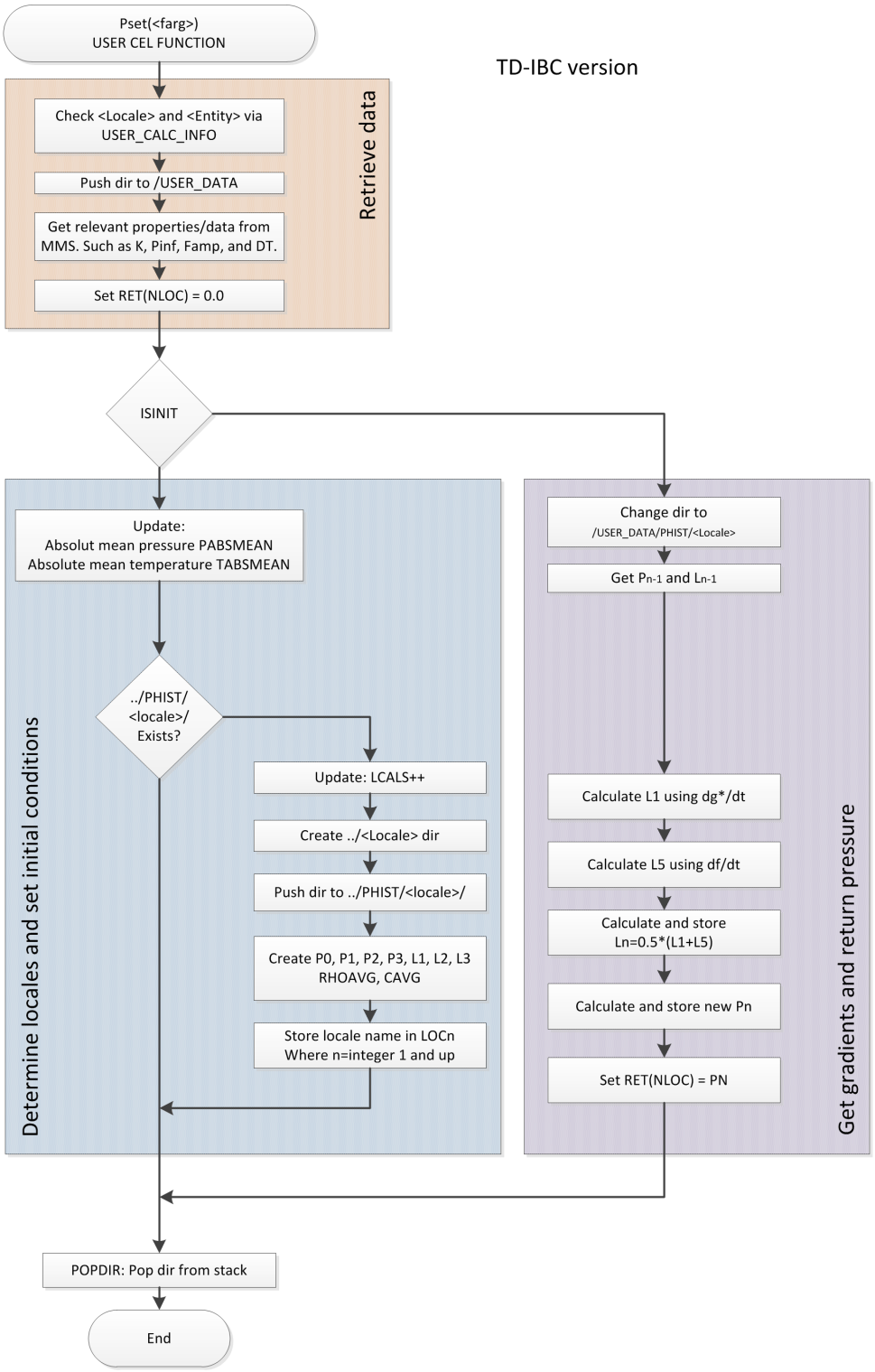


Figure 3.19: Simplified Pset subroutine flowchart (TD-IBC version)

3.4 Reflection measurement

The multi-microphone method discussed in section 2.7.1 is programmed in *Matlab* using object oriented programming instead of classing programming. This is done to make the code more general, easier to understand, use, and extend by others. A *class* called *MultiMicMethod* is written that stores all the data, and harbors all functionality. The class is structured in such a way that other users can easily extend the functionality by adding methods to the class, or use class inheritance and keep the original class intact.

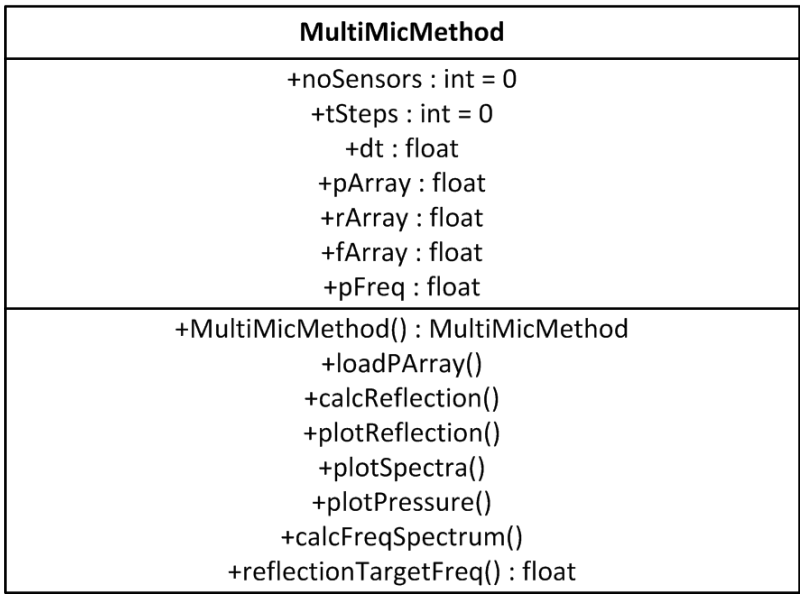


Figure 3.20: UML diagram of MultiMicMethod Class (incomplete, see appendix for full UML)

The most important properties and methods are shown in the Unified Modeling Language (UML) diagram in Figure 3.20. This is only a short list of important properties and methods, the full UML diagram is added to the appendix. The work-flow in short, first an MultiMicMethod object needs to be created, and then the pressure time-series can be imported from *ANSYS CFX* using the *loadPArray* function. The frequency spectrum for the pressure time series for each microphone must be created using the *calcFreqSpectrum* function before one can calculate the reflection coefficient R as a function of frequency with the *calcReflection* function. With the function *reflectionTargetFreq* one can obtain the reflection coefficient nearest to the given target frequency (as argument). Be advised that one has determined the positioning of the pressure sensors such as to minimize the error in the Multi-Microphone Method for a given frequency (see sec. 2.7.1). Further away from this "design" frequency the error in determining the reflection coefficient R will increase. The *plotReflection* function plots the reflection coefficient as a function of frequency.

Losses due to turbulence and viscosity can also be taken into account using the *MultiMicMethod* class, but is not used in determining the reflection coefficient R in all the defined test cases in section 3.5. The inclusion of viscosity in the heat transfer equations (see sec. 3.2.2) resulted in a negligible difference in the non-viscous reflection coefficient and solution (see sec. 4.1.1), therefore only the non-reflective version is used. More detail about taking into account losses in the multi-microphone method is added to the appendix.

3.4.1 Measurement in test-cases

In the test-cases defined in section 3.4.1 the reflection coefficient is measured using four equidistant microphones per wavelength, as this is seen optimal according to MMM theory (see sec. 2.7.1). The wavelength varies depending on whether the simulation is performed isothermal, or the heat equation is included (see sec. 3.2.2). The inclusion of a heat transfer equation is imperative in thermoacoustics as the change in temperature is what drives thermoacoustic devices. Therefore this wavelength of $\lambda = 3.472m$ is chosen as a basis to calculate the microphone spacing Δx_{mic}, which results in $\Delta x_{mic} = 0.0825m$ (see Table 3.4 for absolute position of microphones, coordinate x and d as shown in Figure 2.8). The pressure time-series of the microphones are exported from *ANSYS CFX* to comma-separated values (CSV) format. These timeseries are then used as input for the *Matlab* class *MultiMicMethod* which calculated the reflection coefficient R based on the MMM method.

Table 3.4: Microphones used for determining reflection

Microphone	P_1	P_2	P_3	P_4
x position	1.196	2.064	2.932	3.8
d position	2.804	1.936	1.068	0.2

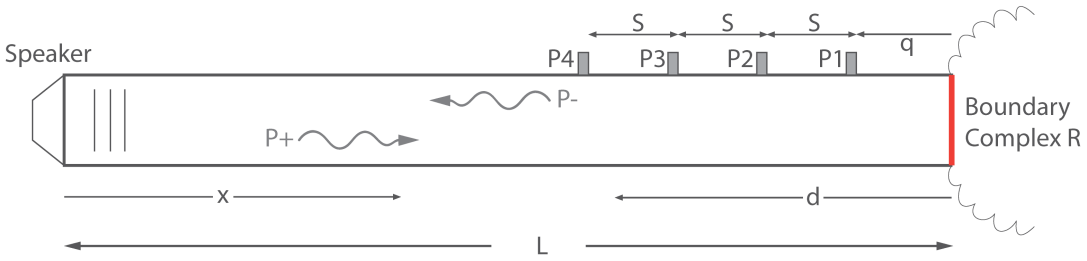


Figure 3.21: Microphone placement within domain

3.5 Test cases

Several test cases are formulated to test the performance and stability in different situations. First general solver settings and convergence criteria are discussed in section 3.5.1.Then the reasoning behind the simulation time is discussed in section 3.5.2, and the relaxation factor K used in the LRM term is discussed in section 3.5.3. Distinction is made between the non-reflective and TD-IBC test cases. Not only the phase and absolute value of the reflection coefficient can explicitly be set in the TD-IBC boundary, but also the way in which the non-reflective character is being established is different (see sec. 3.3.9.2). The non-reflective boundary test-cases are discussed in section 3.5.4, and the TD-IBC test-cases are discussed in section 3.5.5.

3.5.1 Solver settings and convergence criteria

For all simulations the same solver settings and convergence criteria are used. Because the acoustic CFL number used is on the save side (less than one) and a very basic domain and models are used, a high resolution advection scheme is used instead of an upwind scheme. This minimizes numerical dissipation and maximize accuracy. The transient scheme used is second order backward Euler with a timestep initiliazation being extrapolated from the previous timestep. The residual is set to $max\ 1 \cdot 10^{-6}$.

3.5.2 Simulation time

On the basis of the length of the domain and the characteristic sound speed, the wavefront is calculated to arrive at the boundary after $\frac{4}{343,2} = 0.011655s$. To see a standing wave appearing in the tube the calculation time should be at least twice this time. To be able to exclude transient effects from influencing the reflection coefficient calculated using the multi-microphone method, the simulation time should be such that at least a few periods in a periodic steady state is attained. Preliminary simulations showed that some simulations need at least ten periods to attain a periodic steady state (due to pressure drift). On the basis of this observation, a total simulation time of $t = 0.2s$, and a sampled range from $t = 0.015s$ to 0.2 is chosen, which results in 15.87 periods that are sampled.

3.5.3 Relaxation factor

The relaxation factor K influences the reflection coefficient R. Theoretically this relation is given by equation 2.23 in section 2.4. The reflection coefficient R of the boundary conditions are tested against K, and compared to the theoretical value of R as a function of K. It is expected that the measured reflection coefficient is near the theoretical reflection coefficient, and behaves in the same way. The relaxation factor K is varied logarithmically between $K = 1s^{-1}$ and $10^5 s^{-1}$. In *ANSYS CFX* the coupling parameter σ is used instead of K, but can be converted using equation 2.24. The optimal values 0.27 and 0.58, as discussed in section 2.4, have the corresponding relaxation factors of $K = 23.436^1/_s$ and $K = 50.344^1/_s$ respectively. These values for K are rounded to $K = 23^1/_s$ and $K = 50^1/_s$, and used across all other test-cases as defined in section 3.5.4 and 3.5.5. All values for relaxation factor K, and corresponding coupling parameters σ, are summed up in Table 3.5.

Table 3.5: K values and corresponding σ values for a mean flow $u_0 = 0$

K	10^0	10^1	23	50	10^2	$2 \cdot 10^2$	$5 \cdot 10^2$	10^3	10^4	10^5
σ	0.0115	0.1152	0.2650	0.5760	1.152	2.304	5.760	11.52	115.2	1152

3.5.4 Non-reflective boundary test cases

The test-cases defined in this section are used to compare three NRBCs: the built-in *ANSYS CFX* NRBC, the custom NRBC based on the NSCBC method including the LRM term, and the custom NRBC including both LRM and PWM terms. Three important parameters are varied among the test-cases, which are summed up in Table 3.6. Calculation of all possible combinations would result in 120 test cases per boundary, and a grand total of 360 for all three boundaries. To decrease the number of test-cases, only certain combinations are tested, which are elaborated upon in this section. The influence of mean flow u_0 on the non-reflective boundaries is tested for optimal relaxation factors $K = 23 [1/s]$ and $K = 50 [1/s]$ only. Main figures of merit during simulations are the reflection coefficient R and the pressure drift P_{drift}.

Table 3.6: Possible combinations test cases

Heat model	Isothermal, Thermal energy, Total energy	
Relaxation factor K	10^0, 10^1, 23, 50, 10^2, $2 \cdot 10^2$, $5 \cdot 10^2$, 10^3, 10^4, 10^5	$[1/s]$
u_0	0, 0.5, 2.5, 5	$[1/s]$

Before the standard test cases are performed, first the influence of different time integration schemes and the type of heat transfer equation are analyzed. The time integration schemes (see sec. 3.3.7.1) are tested using the total energy heat equation without the viscous work term, and a relaxation factor of $K = 50 [1/s]$, and are summed up in Table 3.7.

Table 3.7: Time integration test cases

	Integration order	L^* predictor
Influence of time integration	1^{st}	$L^* = L^n$
	2^{nd}	$L^* = L^n$
	3^{rd}	$L^* = L^n$
Influence of L^* predictor	1^{st}	$L^* = L^{n+1}$ (implicit)
	1^{st}	$L^* = \frac{1}{2}(L^n + L^{n+1})$

The influence of the heat transfer equation, and their options, are tested. The test cases are summed up in Table 3.8. The heat transfer test cases are only performed for the built-in *ANSYS CFX* NRBC and the custom NRBC. A relaxation factor of $K = 50 [1/s]$ is used.

Table 3.8: Heat transfer test cases, $K = 50 [1/s]$

Heat transfer equations	Pressure transient term	Viscous terms
Thermal energy	Off	Off
	Off	On
	On	Off
	On	On
Total energy	N/A	Off
	N/A	On

The best performing time integration scheme is determined to be second order with an

explicit Lodi function: $L^* = L^n$ (see sec. 4.1.2 for results and analysis). The best performing heat transfer equation is the total energy equation without viscous terms (see sec. 4.1.1 for results and analysis). These options are used in all other simulations, unless specified otherwise.

3.5.4.1 Smooth sine construction

The wavefront of a sine impinging on the boundary shows a discontinuity in its gradient, which can be seen in Figure 3.22. This causes an overshoot and wiggly behavior in the numerical solution of the gradient (see sec. 4.1.7). A hypothesis is that this initiates drift. To test this, test-cases are constructed with a sine that is smoothed at the front.

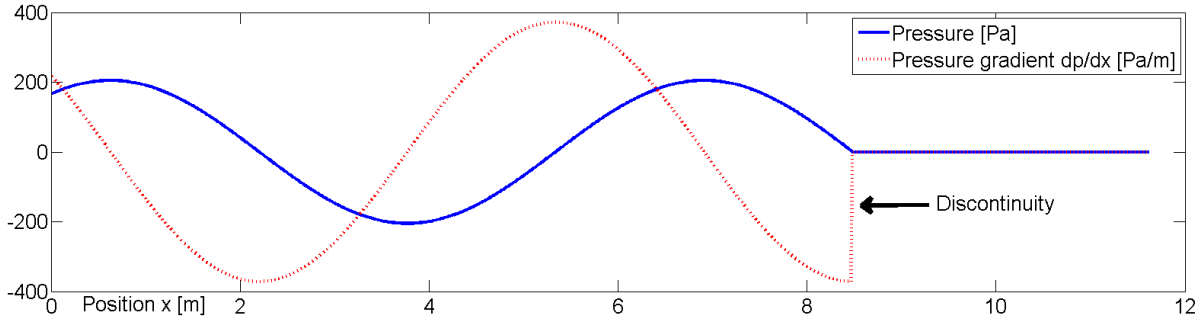


Figure 3.22: Normal sine and gradient

The sine is smoothed by multiplying a polynomial $f(x)$ for the first quarter of the wave. To prevent introducing yet another discontinuity, the polynomial should match certain criteria. First, the product of the functions should equal zero at the onset of the wave, and be equal to the sine amplitude at a quarter wave-length (at maximum amplitude). This corresponds with $f(0) = 0$ and $f(1) = 1$ (normalized 1 for 0.5π). Also, to prevent a discontinuity, the gradient of the product should be zero at both the wave-front and at maximum amplitude, which results in $f'(0) = 0$ and $f'(1) = 0$. Given these four constraints, the polynomial should be of third order to have a fully determined problem. After solving the system of equations, the smoothing function f and its derivative are found, and given by equation 3.23.

$$f(x) = 3x^2 - 2x^3 \qquad f'(x) = 6x - 6x^2 \tag{3.23}$$

The resulting sine and corresponding gradient are given by Figure 3.23. One can observe a small "kink" in the gradient, the gradient is therefore not completely smooth and causes a discontinuity in the second spatial derivative. This might cause some inaccuracies, but should be far less severe than with an normal sine. The second spatial derivatives are not used in the construction of the boundary conditions, therefore no problems are expected.

The smooth sine is added to definition file in *ANSYS CFX PRE* using the CEL statement in equation 3.24.

$$u = u_0 + [(3t_{norm}^2 - 2t_{norm}^3) \cdot step(1 - t_{norm}) + step(t_{norm} - 1)] \cdot u_a \cdot sin(2\pi ft) \tag{3.24}$$

The *step* function is zero when its argument is smaller than zero, and one when it is larger than zero. t_{norm} is the normalized time, from beginning to end of smoothing: $t_{norm} = {}^{t}/_{t_{end}}$.

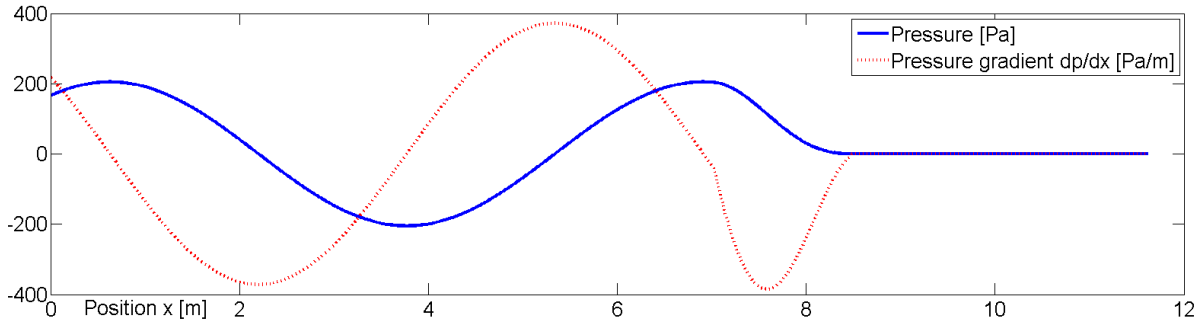


Figure 3.23: Smooth sine and gradient

The influence of the smooth sine is tested for all three NRBCs (*ANSYS CFX* NRBC, custom NRBC, and custom NRBC+PWM) using the ten relaxation factors mentioned at the start of the section.

3.5.4.2 Linear refinement near boundary

The influence of the linearly refined mesh, as discussed in section 3.1.2, is tested for all three NRBCs (*ANSYS CFX* NRBC, custom NRBC, and custom NRBC+PWM) using the ten relaxation factors mentioned at the start of the section. This results in thirty simulations to be run.

3.5.4.3 Influence of partitioning

Last, the influence of multiple partitions on the performance and stability of all three NRBCs is tested. Both serial, 2 partitions, and 4 partitions are tested. Using a relaxation factor of $K = 50s^{-1}$ and the total energy equation.

3.5.5 Time-domain impedance boundary condition test cases

A lot of different test cases can be thought of for a TD-IBC because it can basically acoustically mimic any type of domain behind the boundary. The test-cases defined in Table 3.9 are designed to cover different relevant types of reflective behavior.

Table 3.9: TD-IBC test cases

Nr.	Test case	R	Frequencies [Hz]
1	Acoustically hard wall	1	All
2	Pressure release surface	-1	100
3	Non-reflecting outlet	0	All
4	Complex R at fixed frequency	$0.5 + 0.5i$	100
5	Complex R over range frequencies	$0.5 + 0.5i$	100-400
6	Levine-Schwinger non-ideal open end	$R(f)$	100-400

All test-cases are tested against theory. Only the non-reflective test-case can be tested against the NRBC boundary conditions. The pressure release surface TD-IBC can be tested against a constant pressure outlet in *ANSYS CFX*, and an acoustically hard wall TD-IBC can be tested against a wall in *ANSYS CFX*. The reflection coefficient R of the test-cases is also determined using the Multi-Microphone method (see sec. 2.7.1), because it also captures phase

information.

3.5.5.1 Acoustically hard wall and pressure release surface

The acoustically hard wall and pressure release surface have relatively simple coefficient sets, only one non-zero coefficient for both a and b. The acoustically hard wall causes pressure doubling at the wall, and a reflection coefficient that is real and positive ($R = 1$). This results in two non-zero coefficients; $a_0 = 1$ and $b_0 = 1$. For the pressure release surface the reflection coefficient is $R = -1$. The phase of R is π, which results in a time delay of half a period. At a frequency of $f = 100Hz$ this is 0.005 seconds. Using equation 2.42 and $\Delta t = 10^{-5}$, n is determined to be 500. The non-zero filter coefficients are $a_{500} = 1$ and $b_0 = 1$.

3.5.5.2 Non-reflecting outlet

The non-reflecting outlet in question is different from the *ANSYS CFX* and custom NRBC based on the NSCBC formalism, as $\mathcal{L}_5$ is specified using the characteristic wave f determined at the sample plane (see eq. 3.21) instead of the gradients at the boundary. The corresponding filter coefficients are relatively simple, b_0 must be equal to 1.

3.5.5.3 Complex reflection at fixed and varying frequency

A complex reflection of $R = 0.5 + 0.5i$ is tested with filter coefficients derived from one fixed frequency, and fitted along different frequencies using the method by E.C. Levy [11]. The complex reflection coefficient of $R = 0.5 + 0.5i$ represents an absolute reflection of $0.5 \cdot \sqrt{2}$, and a phase of 45°. This is done to mimic a partially reflective outlet with a phase change somewhere between an acoustically hard wall and a pressure release surface. Using equation 2.41 and 2.42, n is determined to be 125, which results in two filter coefficients: $a_{125} = 0.5 \cdot \sqrt{2}$ and $b_0 = 1$, the rest being zero.

The second case which is fitted among different frequencies has a more complex coefficient set. The reflection coefficient is fitted from 100 Hz to 400 Hz. At least 125 filter coefficients are needed. However, calculating a stable IIR filter (also called minimum-phase filters) for this many filter-coefficients proved to be a difficult task. Even a damped Gauss newton iteration method (see [13]) does not result in all poles to reside within the unit circle. Varying parameters, such as the order (number of filter coefficients) and frequency, does not result in a stable filter. It seems that the *invfreqz* function in Matlab cannot find stable solutions for a high order, higher than 10, transfer function. The complex reflection coefficient R is nonetheless fitted very accurately. See Figure 3.24 for fitted R and pole-zero plot using 303 coefficients.

Filter coefficient sets with only a few poles slightly outside the unit-circle, for frequencies far from the excitation frequency of $f = 100Hz$, have been obtained. These filter coefficient sets also proved to be unstable. The hypothesis is, that although the dominant frequency is $f = 100Hz$, practically all frequencies that can be represented on the grid are excited, however small. An extremely small, but non-zero, amplitude corresponding to a freqeuncy of a (unstable) pole slightly outside unit-circle causes the amplitude to grow each timestep, until the amplitudes becomes extreme, and the solver crashes. When looking at the Fourier transform of the pressure time-series (see Fig. 3.25), it becomes clear that indeed the amplitude of practically all frequencies are excited, albeit usually extremely small compared to the dominant excitation frequency of $f = 100$. These frequencies can be excited by numerous sources, such as solver and boundary condition characteristics, the finite precision of numbers stored in memory, and

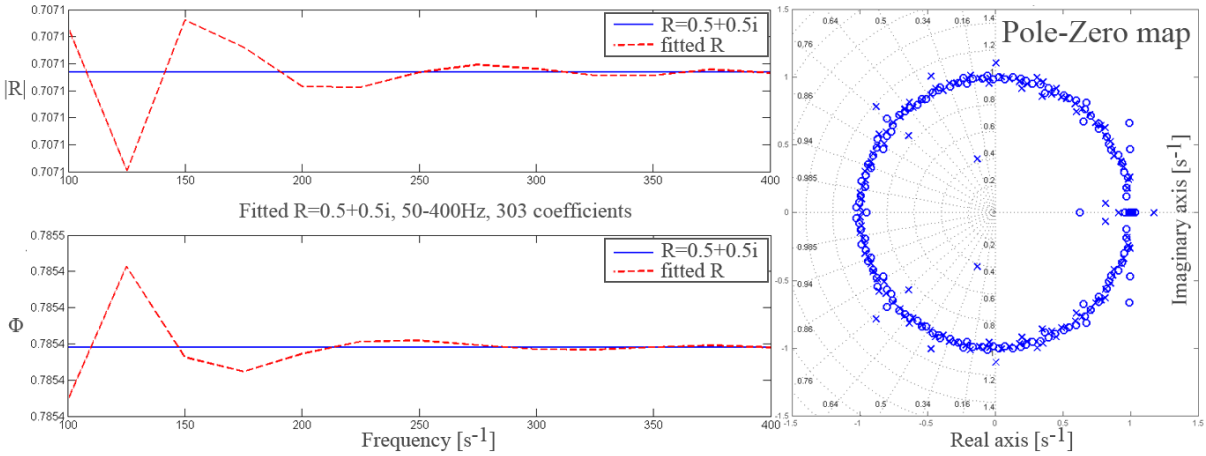


Figure 3.24: Unstable IIR configuration for $R = 0.5 + 0.5i$, x:pole, o:zero

the non-smooth sine as its wavefront arrives at the boundary.

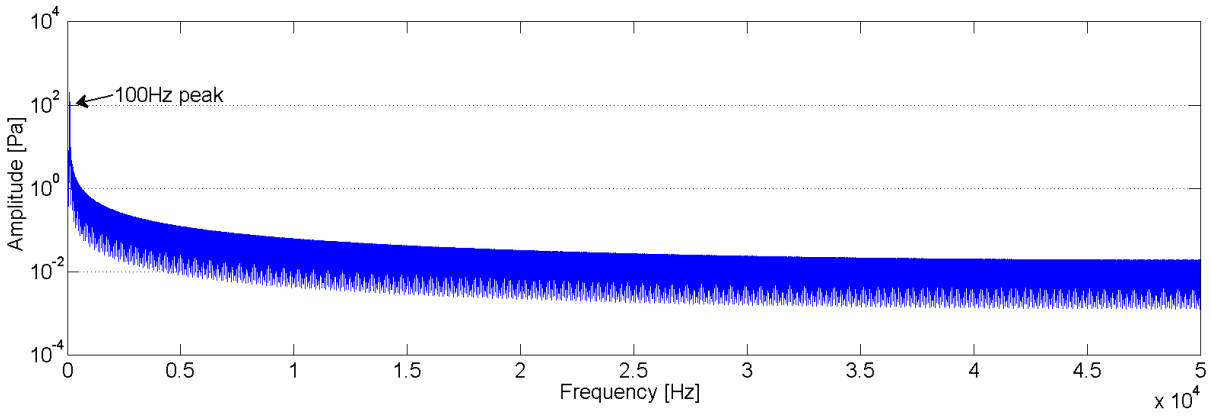


Figure 3.25: Spectra for pressure sensor P_1, the custom NRBC, $K = 50s^{-1}$, Total energy, $u_0 = 0^m/_s$.

An obvious alternative would be to use a FIR filter which has no poles by definition. After simulation this also seems to result in instabilities. This time not because of poles outside the unit circle, but probably because of the large coefficients which can cause quantization error [6]. Calculated coefficients are order $\mathcal{O}(10^5)$, while for IIR filters the coefficients are generally of order $\mathcal{O}(10^0)$.

By reducing the order of the denominator to 11, and reducing order of numerator to 302, a stable system in the sense of no poles outside the unit circle can be retrieved (see Fig. 3.26). However, even this system proves to be unstable during simulations. This might also have to do with quantification error, as coefficients are generally of order $\mathcal{O}(10^1)$. Another cause might be that the low denominator order might not include enough historical information about $g*$. Another problem with this coefficient set, although not a stability problem, is that the complex reflection coefficient R is not fitted very accurately.

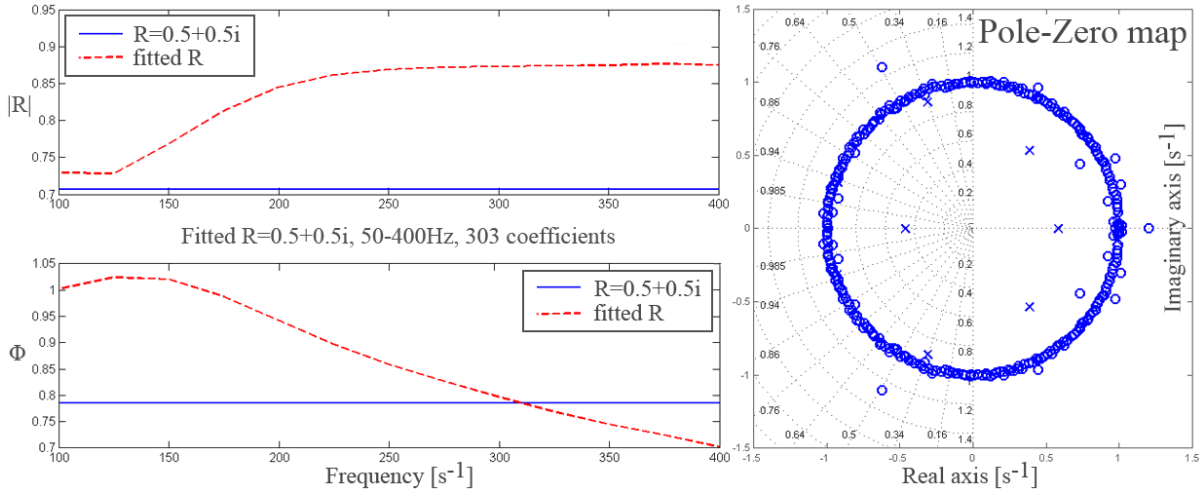


Figure 3.26: "Stable" FIR configuration for $R = 0.5 + 0.5i$, x:pole, o:zero

3.5.5.4 Levine-Schwinger non-ideal open end

Levine constructed a set of equations to analytically calculate the complex reflection coefficient of an unflanged open pipe [10]. These equations are quite complicated, and therefore a rational equation from Norris is used to approximate the Levine-Schwinger equations [2]. The equations for the absolute reflection $|R|$ and phase angle ϕ are given by equations 3.25 and 3.26 respectively.

$$|R| = \frac{1 + 0.2 \cdot (ka) - 0.084 \cdot (ka)^2}{1 + 0.2 \cdot (ka) + 0.416 \cdot (ka)^2} \tag{3.25}$$

$$\phi = \pi - 2\pi a \cdot \frac{0.6133 + 0.027 \cdot (ka)^2}{1 + 0.19 \cdot (ka)^2} \tag{3.26}$$

But also here, as in the complex reflection case, no stable filter coefficient sets are found after varying various parameters.

Chapter 4

Results

The results are separated into two sections, one for the results of the non-reflective boundary conditions, and the other for the results of the Time-Domain Impedance Boundary Condition (TD-IBC). In the first section the custom Non-Reflective Boundary Condition (NRBC) with and without Plane-Wave Masking (PWM) is compared to the *ANSYS CFX* NRBC and theory. In the second section the TD-IBC is compared to theory, and for certain complex reflection coefficients they are compared to other boundary conditions. Those boundary conditions include a wall and a constant pressure outlet, which are fully reflecting. Also a non-reflective TD-IBC is tested, as its determination of $\mathcal{L}_5$ differs from the other NRBCs (see sec. 2.3.2), and is compared to the custom NRBC, the PWM NRBC, and the *ANSYS CFX* NRBC.

The phase angle ϕ of the measured complex reflection coefficient is not analyzed (but it is registered in the appendix) for the non-reflective boundary conditions. The goal of non-reflective boundary conditions, as the name implies, is to be non-reflective, this means that the phasor of the complex reflection coefficient R is generally small, and is therefore prone to measurement error of the phase angle. It is also less relevant, as it is not the goal of a non-reflective boundary condition to influence the phase angle. In contrary to the non-reflective boundary conditions, the phase-angle ϕ is fundamental for TD-IBCs, and is therefore taken into account in this section.

The main idea of this chapter is to characterize the different boundary conditions, and to asses their performance. Several unexpected trends, odd behavior, and lacking performance in certain areas were observed in various simulations. Not all of the observations can be explained with absolute certainty, as research would be time-consuming and uncertain. For those observations only hypothesis are postulated based on theory on other observations.

The results in the following sections are mainly presented by graphs, as showing all the figures of merit for all simulations would not aid in the readability, given the number of simulations that were run. The numbers behind all the graphs in the results chapter are added to the appendix.

4.1 Non-reflective boundary conditions

First the influence of different simulation parameters such as the heat transfer equation and time integration scheme is tested in section 4.1.1 and 4.1.2, of which the best option is used in subsequent test cases. The influence of the relaxation factor K on reflection coefficient R is tested in section 4.1.3. Other test cases include the influence on the solution of a mean flow u_0 (sec. 4.1.4), partitioning (sec. 4.1.5), analysis of the influence of discontinuous gradients (sec. 4.1.7), and linear refinement near the boundary (sec. 4.1.6).

4.1.1 Heat transfer equation

The influence of the heat transfer equation options in *ANSYS CFX* is tested for both the custom NRBC as well as the *ANSYS CFX* NRBC. The results are presented in Figure 4.1 for the reflection coefficient R, and in Figure 4.2 for the CPU time. The thermal energy equation is abbreviated as "Thermal E", and the total energy equation as "Total E". The acronyms PT stands for Pressure Transient term on, and VT stands for Viscous work Terms on. Note that the total energy equation already includes pressure transient behavior on temperature. Explanation about these terms and equations can be found in section 3.2.2.

What immediately stands out from the reflection coefficient graph (Fig. 4.1) are the high reflection values when the pressure transient term is not included in the set of equations, as was expected in section 3.2.2. Although the *ANSYS CFX* NRBC performs a lot better in this case, the reflection coefficient R is still very high with $R = 22.8\%$ compared to the cases in which the pressure transient term is included. The CPU time also approximately doubles compared to the other cases (see Fig. 4.2). While the reflection coefficient R is be lower for the *ANSYS CFX* NRBC, the CPU time is higher. The average amount of coeffient loop iterations are approximately 11.5 for both the *ANSYS CFX* NRBC as well as the custom NRBC in this case. The cause remains unknown, but this does not matter as simulations are concerned, as these parameters (i.e. the exclusion of the pressure transient term) perform abysmally for both the *ANSYS CFX* NRBC and the custom NRBC, and thus will not be used. As expected (see sec. 3.2.2), the pressure transient term is essential for correct function of both NRBCs.

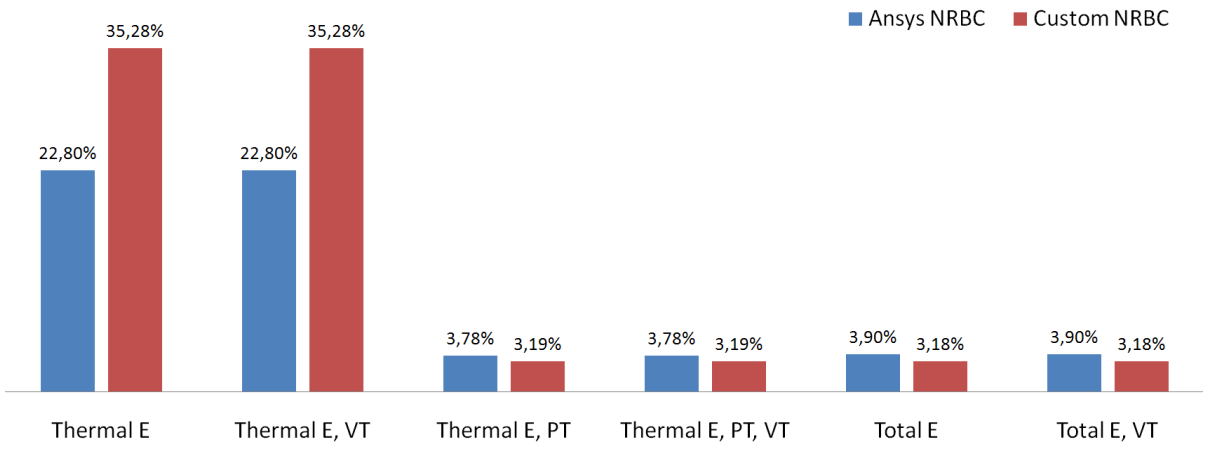


Figure 4.1: Influence of heat transfer equations on the reflection coefficient for the *ANSYS CFX* NRBC and custom NRBC, $K = 50s^{-1}$

The reflection coefficient is between $R = 3.18\%$ and $R = 3.9\%$ for the other four cases tested. A slight, but almost negligible, difference in favor of the custom NRBC is observed compared to the *ANSYS CFX* NRBC. The differences in CPU time between the custom NRBC and the *ANSYS CFX* NRBC are negligible. The inclusion of the viscous work terms do not create any difference, which is to be expected because hardly any attenuation will occur for such short domains with the used simulation parameters.

The difference in the reflection coefficient between thermal energy equation and total energy equation is very small. The total energy equation is a more complete equation, but actually takes approximately an hour shorter to run than the thermal energy simulations. For both the *ANSYS CFX* NRBC and the custom NRBC the average number of coefficient loops is approximately three for the total energy simulations, while it is approximately five for the thermal

energy simalations. The total energy simulations thus converges faster. The main difference between the thermal energy and total energy equation is the inclusion of the kinetic energy term and pressure transient term in the total energy equation (see sec. 3.2.2). It might be that the pressure transient term is added as an extra equation in the set of equations when the thermal energy equation is used, while in the total energy equation it is included in the equations, and therefore takes more time to converge. Based on the results, -a shorter simulation time and comparable reflection-, the choice is made to use the total energy equation excluding the viscous work term over the thermal energy equation including the pressure transient term.

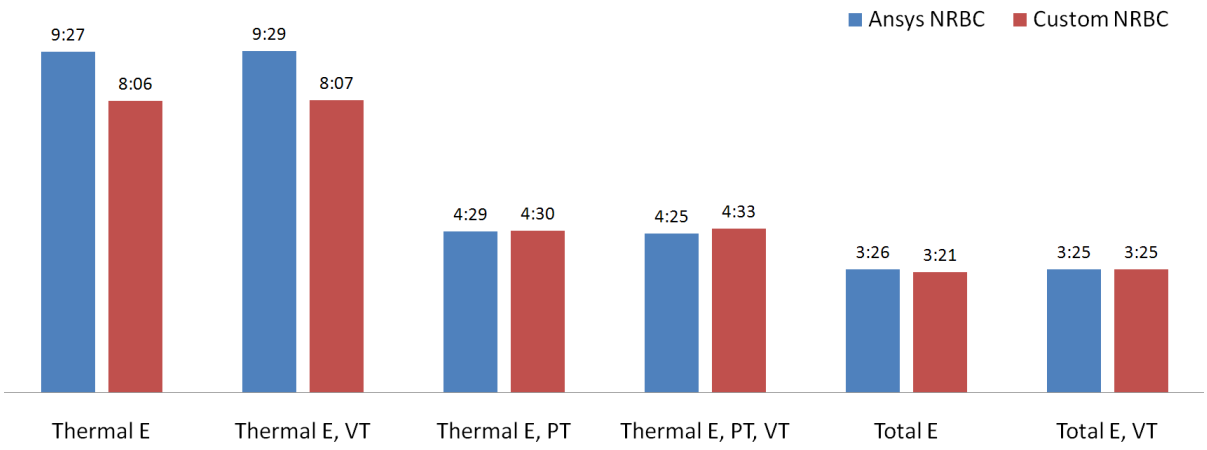


Figure 4.2: Influence of heat transfer equations on CPU time $[hours : min]$, $K = 50s^{-1}$

4.1.2 Time integration

The influence of the time-integration schemes for the custom NRBC are tested, as it only applies to the custom NRBC and one does not have such level of control over the *ANSYS CFX* NRBC. The five cases tested are shown in Figure 4.3. It is observed that a third order approximation to the time derivative does not result in a lower reflection coefficient over the second order approximation. The most noteworthy observation however, is that against expectations both the implicit as well as the half-way predictor showed a higher reflection coefficient R compared to the explicit first order expression. This might have something to do with the manner in which the solver constructs and calculates its conservation equations, but is outside the scope of this research.

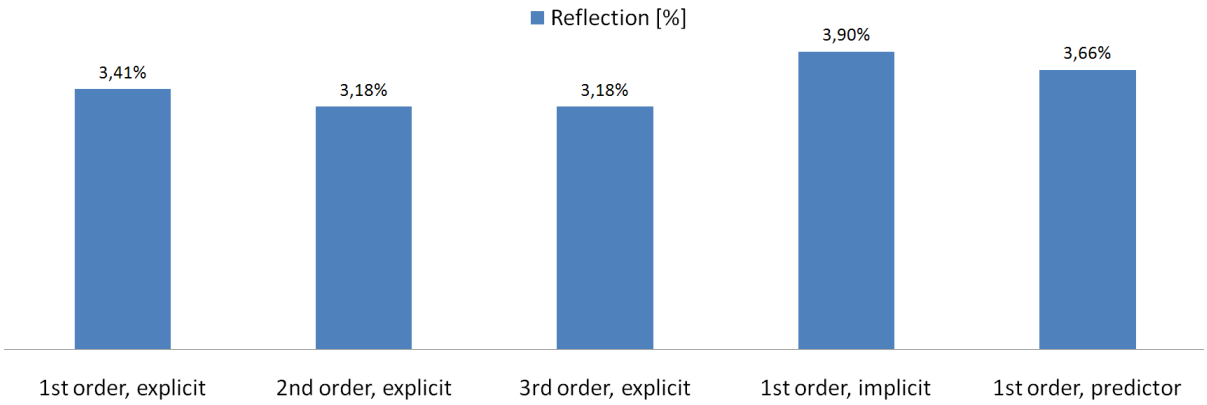


Figure 4.3: Influence of the time integration scheme on reflection for the custom NRBC, $K = 50s^{-1}$

4.1.3 Relaxation factor

The reflection of the *ANSYS CFX* NRBC, and the custom NRBC, both including and excluding the PWM term, are tested using various relaxation factors K, and compared to the analytical solution. Both the isothermal and total energy equation are used because of distinct properties discussed in section 3.2.2. A lower relaxation factor K should result in a lower reflection coefficient R, but is also more prone to pressure drifting (see sec. 2.4).

4.1.3.1 Reflection coefficient

The results for the *ANSYS CFX* NRBC are shown in Figure 4.4. The black line is the analytical solution. For the total energy simulations (red dots) the results are practically equal to the analytical prediction, and has a minimum reflection coefficient of $R = 0.12\%$ for a relaxation factor $K = 1s^{-1}$. More peculiar is the large reflection coefficient in the order of 8% for isothermal simulations (blue dots) at lower relaxation factors, while one would expect values in the same order as the total energy equation. The custom NRBC does not show this behavior (see Fig. 4.5) for isothermal calculations. The problem might possibly reside in the wrong assumption of the sound speed for isothermal calculations by *ANSYS CFX*. The returned sound speed, by either monitor points within the domain as well as via a *Fortran utility routine* called *user_getvar*, is equal to the sound speed in which adiabatic expansion can occur. This is physically incorrect, as the sound speed should be equal to the isothermal sound speed, which is $1.4^{-0.5}$ lower (see sec. 3.2.2). It seems that an adiabatic index of $\gamma = 1.4$ is taken to calculate the sound speed from the primitive field quantities, while this should be $\gamma = 1.0$. For both isothermal and total energy equations no pressure drift is observed, even at a relaxation factor $K = 1s^{-1}$.

The CPU time is approximately 3:25 hour for the total energy simulations, and 8:30 hour for the isothermal simulations. The custom NRBC, both including and excluding the PWM term, shows the same order of CPU time, and thus does not seem boundary condition dependent. The total energy simulations need on average three coefficient loops per timestep to meet the convergence criteria, while the isothermal simulations need twelve coefficient loops per timestep. The isothermal simulations are harder to solve, which might be caused by the heat transfer needed to maintain a constant temperature (see sec. 3.2.2).

In Figure 4.5 the results for the custom NRBC are shown. The reflection coefficient is consequently slightly below the theoretical value for relaxation factors between $K = 23s^{-1}$ and $K = 1000s^{-1}$, but follows the same general trend nonetheless. Both the isothermal and total

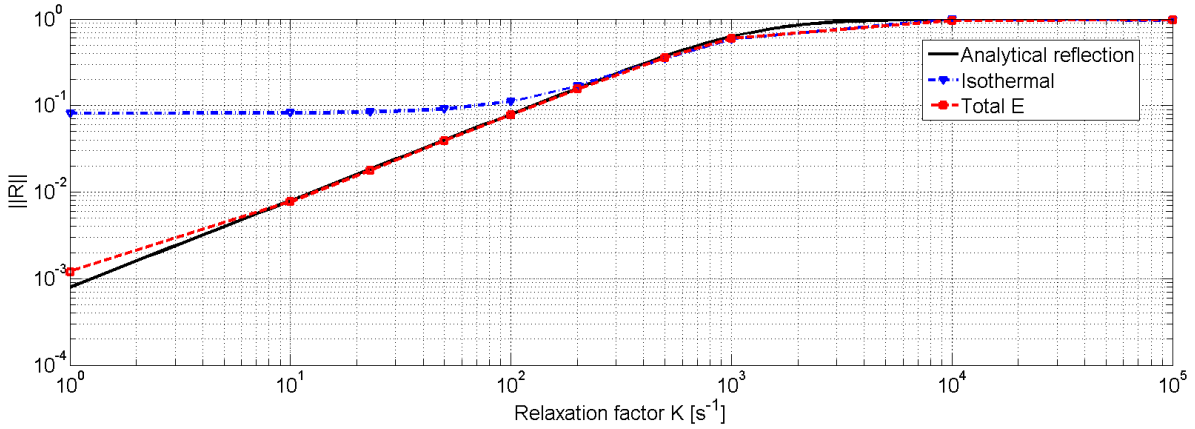


Figure 4.4: Absolute reflection coefficient vs. relaxation factor, for the *ANSYS CFX* NRBC

energy simulations bottom out at $R = 1.31\%$, with a minimum of $R = 1.156\%$ for the total energy simulation using a relaxation factor $K = 10s^{-1}$. The *ANSYS CFX* NRBC does not show this bottoming out for the total energy simulations. It seems that something else is limiting the accuracy of the boundary condition, which could be caused by the discretization error in the returned pressure and velocity gradients near the boundary, that are determined using one-sided difference approximations. Another cause could be that the used (fluid) properties are not accurate. The bottom at around $R = 1.3\%$ is low nonetheless, so for practical application this is probably acceptable.

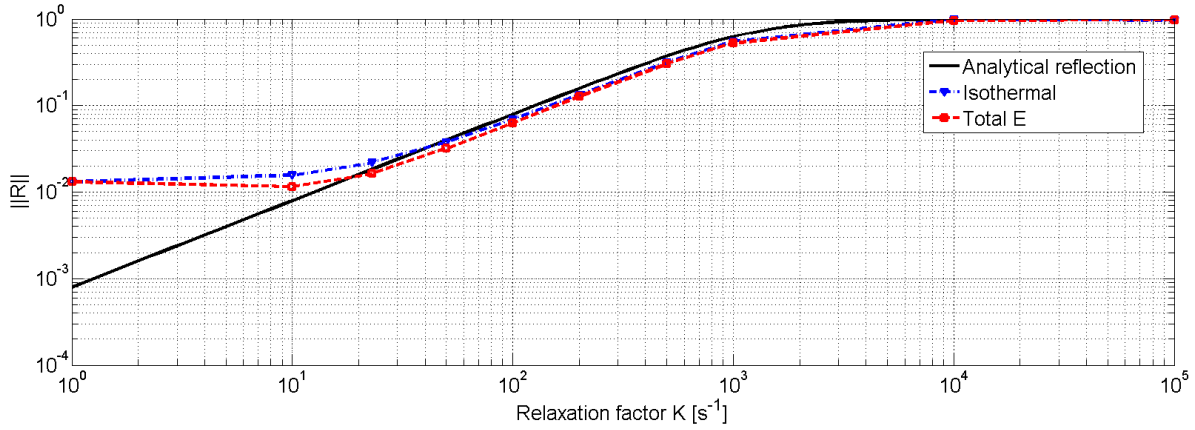


Figure 4.5: Absolute reflection coefficient vs. relaxation factor, for the custom NRBC

The results for the custom NRBC including the PWM term are shown in Figure 4.6. The reflection coefficient, for both the isothermal and the total energy simulations, remain fairly constant at around $R \approx 1.2\%$ throughout most of the relaxation factors, with a minimum of $R = 0.07\%$ for a relaxation factor of $K = 1000s^{-1}$. There is no relation with the analytically determined values for the reflection coefficient R as a function of the relaxation factor K, which can be expected because in the PWM method the pressure difference due to acoustics is "masked" from the LRM term, while the LRM term without PWM is the basis of the analytically determined equation for R as function of K (eq. 2.23).

The minimum around $K = 10^4 s^{-1}$ looks impressive due to the logarithmic scale, but the reflection coefficient is actually roughly only 1.1% lower.

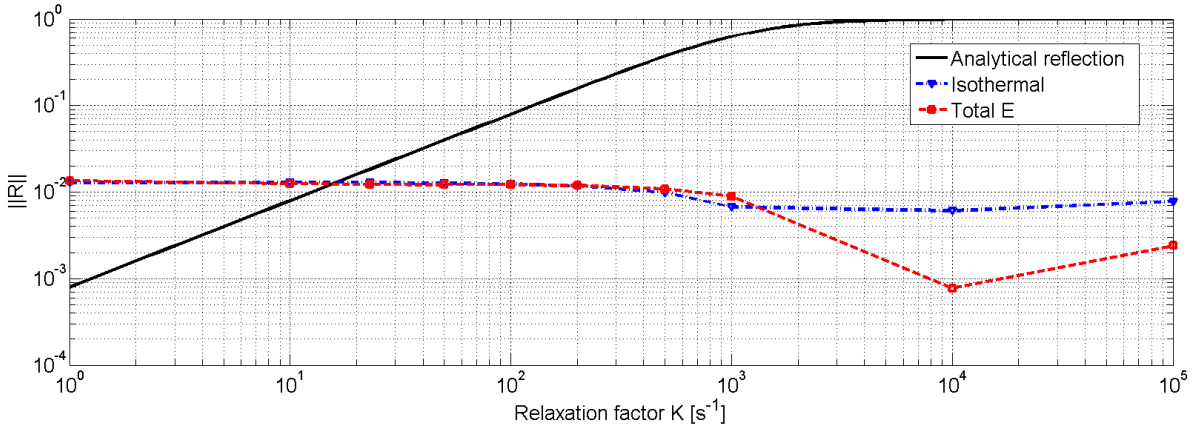


Figure 4.6: Absolute reflection coefficient vs. relaxation factor, for the custom NRBC+PWM

4.1.3.2 Pressure drift

The custom NRBC, with and without PWM, showed pressure drift for certain relaxation factors, the *ANSYS CFX* NRBC on the other hand, did not. Two figures of merit are introduced to characterize this behavior, a maximum pressure drift $P_{drift-max}$ and a stabilized pressure drift $P_{drift-sta}$, as shown in Figure 4.7. The pressure drift for the total energy equation with a relaxation factor of $K = 50s^{-1}$ is shown in Figure 4.7. The pressure time-series of microphone P_2 is plotted to show the pressure drift (see sec. 3.4.1 for the placement of the microphone).

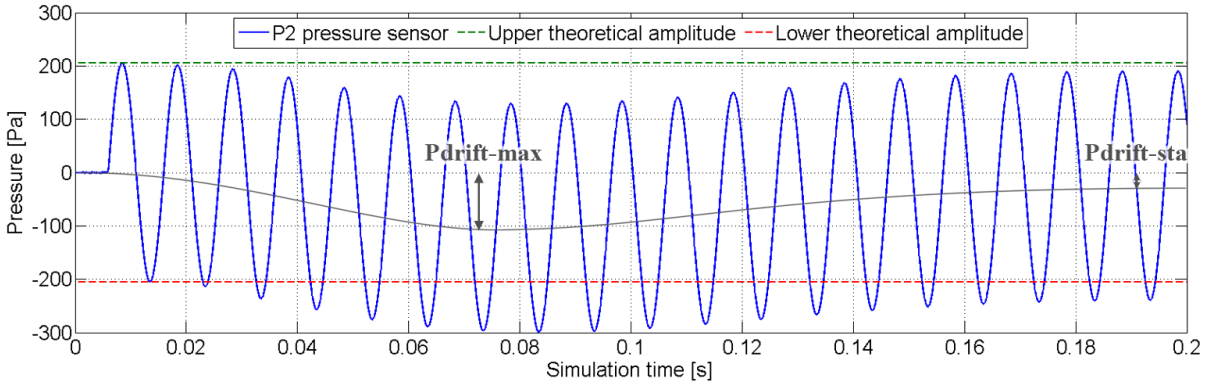


Figure 4.7: Pressure drift figures of merit. Example pressure is P_2 (see sec. 3.4) for the custom NRBC, total energy equation, and $K = 50s^{-1}$

For comparison of different simulations a pressure drift versus relaxation factor K graph is constructed, and shown in Figure 4.8. Certain values are missing, which means that they cannot be determined, for example when the pressure did not stabilize within the simulation time.

Comparing the isothermal and total energy simulations, it becomes apparent that the isothermal simulations show much less pressure drift than the total energy simulations. For the total energy simulations the trend between the pressure drift and relaxation factor is clear, the pressure stabilizes with increasing relaxation factor K. This trend is expected, as it is the reason the LRM term is introduced (see sec. 2.4). For the isothermal equations the trend is less clear and remains fairly constant throughout the relaxation factors, eventually becoming negligible (lower than $1Pa$) for relaxation factors higher than $K = 500s^{-1}$. The stabilized pressure drift is also close, or equal, to the maximum pressure drift for the isothermal simulations, meaning that no or hardly any overshoot occurs. It is a different story for the total energy simulations, which does show significant overshoot, especially at lower relaxation factors. The

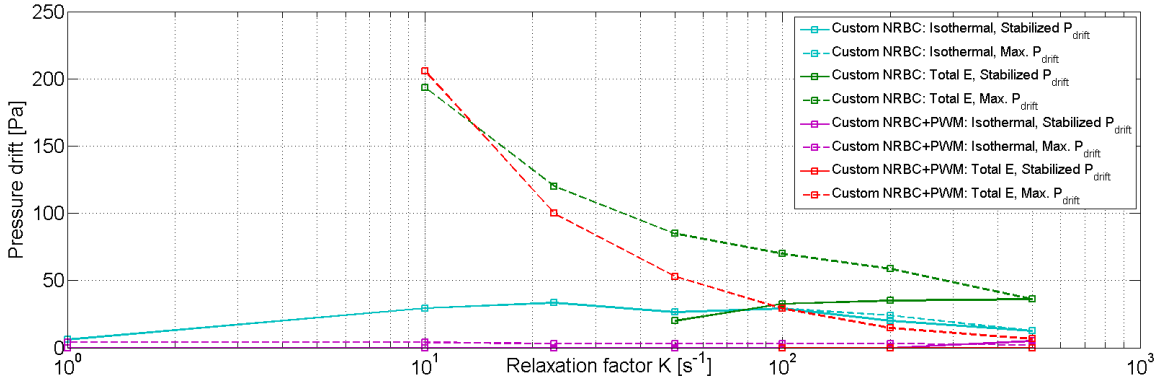


Figure 4.8: Pressure drift vs. relaxtion factor K for the custom NRBC and NRBC+PWM

total energy equations seems to stabilize within the simulation time of $0.02s$ only for relaxation factors $K = 50s^{-1}$ or higher.

Using the total energy heat equation, the custom NRBC including the PWM term performs better and stabilizes sooner than the custom NRBC. But still considerable maximum pressure drift is observed, see Figure 4.8. When the isothermal simulations are run in combination with the PWM NRBC however, practically no pressure drift is observed. The small pressure drift that is observed is in the order of $2Pa$ to $4Pa$, but all stabilize to $0Pa$ within the simulation time.

The mean pressure of certain simulations seems to asymptotically converge to a value other than the expected value of $0Pa$ relative pressure, while a relative pressure of $0Pa$ is expected as the LRM term always "forces" the solution to go to this value. An additional simulation is run to test if the observed non-zero asymptote continuous after $t = 0.2s$. This simulation is five times longer than the standard simulation time, $t_{sim} = 1.0s$ instead of $t_{sim} = 0.2s$. Apart from the changed simulation time, the same properties are used as for the simulation shown in Figure 4.7. The solution of the P_2 microphone at $x = 2.064m$ is plotted in Figure 4.9. It can be seen that the observed mean pressure asymptote did continue after $t = 0.2s$. This odd behavior seems to be caused by another restoring "force" (other than the LRM term) with its equilibrium position elsewhere. It could be that certain properties such as the sound speed or density are not accurate enough, or that the explicit time integration scheme causes this error.

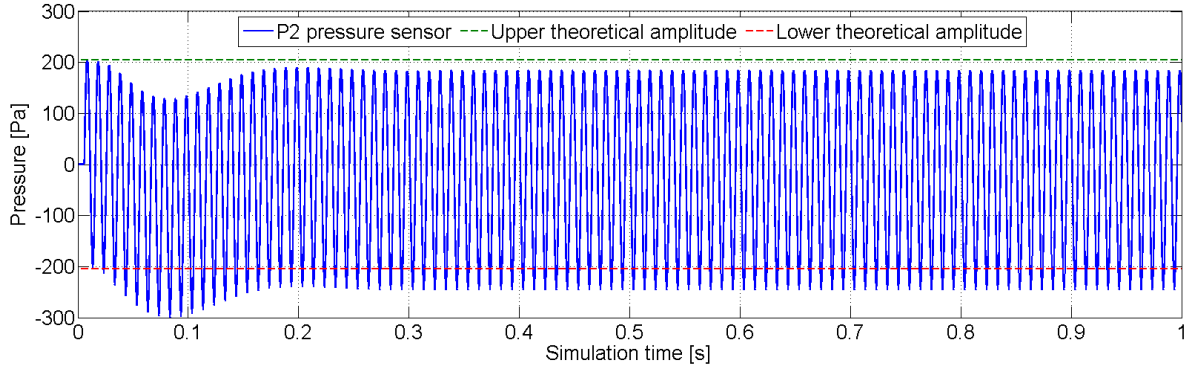


Figure 4.9: Pressure drift for a simulation of 1 second long. Pressure sensor P_2 at $x = 2.064m$ (see sec. 3.4). Custom NRBC, total energy equation, and $K = 50s^{-1}$

Although the exact reason behind the observed pressure drift is unclear, there seems to be a correlation between pressure drift and the varying temperature in combination with the

LRM term. The pressure drift decreases when isothermal simulations are run, which suggests the temperature variation is related to pressure drift. When subsequently the PWM term is included in the custom NRBC to mask the incoming characteristic pressure perturbation from the LRM term, the pressure drift basically vanishes. It could be that the restoring force of the LRM method reacting to the acoustic pressure perturbation in combination with varying temperature at the boundary is causing the pressure and/or velocity gradients to be incorrectly returned, and create a "tendency" (or force) for either a positive or negative relative mean pressure to be attained, which constitutes pressure drift. See section 4.1.3.3 for a more in-depth analysis of the varying temperature and temperature trends on pressure drift.

Another explanation could be that pressure drift of this kind is expected because the problem is not "well-posed" enough, and that the *ANSYS CFX* NRBC has a built-in mechanism that prevents such ill-posedness.

Based on the results of the reflection coefficient and pressure drift, the conclusion could be drawn that isothermal simulations using the custom NRBC inlcuding PWM performs the best, and should therefore be used. But this conclusion is only valid within this idealized numerical problem. Isothermal simulations would not produce useful results when simulating a stack in a thermoacoustic device, as it is the temperature gradient that induces sound waves to form, and thus a heat transfer equation must be used. The custom NRBC including PWM might be influenced negatively in various situations, such as when very viscous fluids are used, with significant turbulence, or when the sampled wave is not traveling perpendicular from the sample plane towards the boundary. In these situations the sampled wave will not be the same as the actual wave at the boundary. But even when this does happen, it is not expected to cause major problems as it is only used to mask the acoustic contribution to the pressure from the LRM term. Only when the phase difference is off by more than a quarter wavelength, problems can be expected, as it will give positive feedback to the LRM restoring term.

4.1.3.3 Temperature trends

One of the possible causes of the previously mentioned problems of pressure drift is the varying temperature. Because *ANSYS CFX* itself determines the temperature at the boundary using its conservation equations (via the option *opening temperature* in *ANSYS CFX*), one does not have direct control over the temperature that influences the solution. The temperature evolution in time for all three NRBCs, using the total energy equations and a relaxation factor of $K = 50s^{-1}$, is plotted in Figure 4.10. The temperature does not vary in isothermal simulations by definition, therefore it is useful to look at isothermal simulations.

All three NRBCs seem to converge to a mean temperature that is $\Delta T = 0.175K$ higher than the expected $T = 298.15K$. The *ANSYS CFX* NRBC converges the fastest to this higher mean temperature, the custom NRBC comes in second, while the custom NRBC including PWM lags the most. Maybe some very slight attenuation of the acoustic waves takes place, of which the energy changes its form to heat by virtue of energy conservation. However, no definitive explanation can be given about the cause of this phenomenon, but it might influence, or even instigates, pressure drift.

The increased temperature also influences the density and sound speed in the domain, albeit slightly. The density decreases with temperature (ideal gas law: $\rho = p/RT$), but the sound speed increases (sound speed, ideal gas: $c = \sqrt{\gamma RT}$). In the definition of the amplitude variation $\mathcal{L}_5$ (eq. 2.20), which is used in the custom NRBC and custom NRBC+PWM to determine the pressure at the next timestep, both the mean density ρ_0 and the small signal sound speed c_0

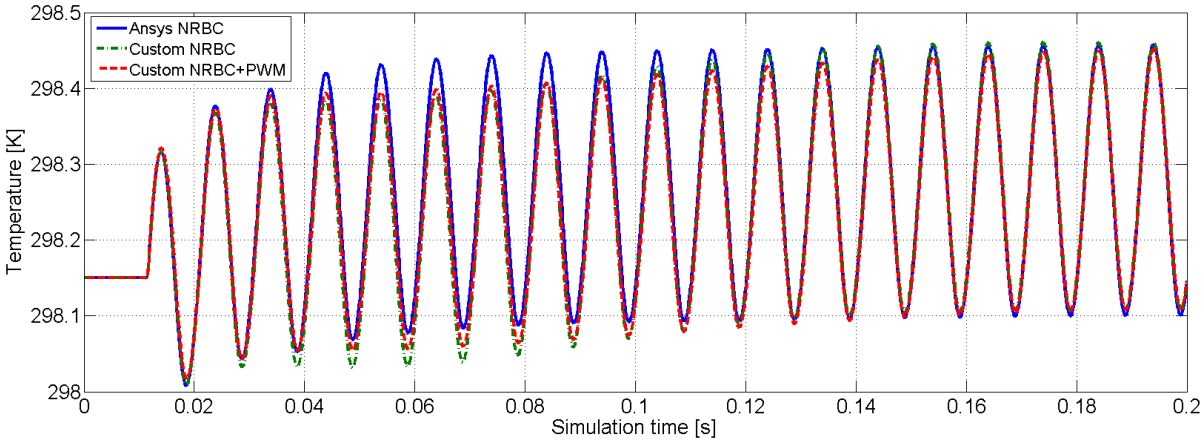


Figure 4.10: Temperature vs. time for the *ANSYS CFX* NRBC, custom NRBC, and custom NRBC+PWM. Total energy, $K = 50s^{-1}$.

are used as factors in the equation. These quantities are determined at initialization of the simulation, using the pressure and temperature at initialization. The hypothesis is that that due to a slightly changed mean density ρ_0 and small signal sound speed c_0, an error is introduced at each timestep. This error builds-up and causes (more) pressure drift. The fact that isothermal simulations show much less pressure drift underlines this hypothesis. An algorithm that updates the mean density and small signal sound speed could potentially decrease this error from building up, and thus preventing pressure drift.

4.1.4 Mean velocity

The influence of a mean flow on the reflection and pressure drift is tested for the *ANSYS CFX* NRBC and the custom NRBC, the last both including and excluding the PWM term. Both the isothermal and total energy equation are used, for a relaxation factor $K = 23s^{-1}$ and $K = 50s^{-1}$ (these relaxation factors are considered optimal, see sec. 3.5.3).

4.1.4.1 Reflection coefficient

The reflection coefficient R of the isothermal simulations is shown in Figure 4.11, from which it can be seen that for the custom NRBC and PWM boundaries, the reflection coefficient R increases significantly with mean velocity. The *ANSYS CFX* NRBC performs abysmally for all mean flows tested, which can be expected from the isothermal results discussed in section 4.1.3, although a slightly decreasing trend can be observed. A possible explanation could be that increasing the mean flow actually makes up for the wrongly calculated sound speed by shear coincidence. With additional mean flow the sound wave travels faster than the isothermal sound speed $c_0 = 295.86^m/_s$ towards the boundary, which is closer to the adiabatic sound speed $c_0 = 346.13^m/_s$ that seems to be incorrectly used by *ANSYS CFX* in the LODI-relations for isothermal calculations (see sec. 4.1.3). When linearly extrapolating the reflection coefficient to a mean velocity of $50^m/_s$, which is approximately the difference between the isothermal and adiabatic sound speed, the reflection coefficient is $R \approx 0.8\%$, which is in the same order as the *ANSYS CFX* NRBC using the total energy equation and without mean flow. To test this hypothesis two additional test cases were run, one with a mean flow of $u = 25^m/_s$, and one with a mean flow of $u = 50^m/_s$. The decreasing reflection coefficient trend indeed continued, with a reflection coefficient of $R = 5\%$ for the $u = 25^m/_s$ simulation, and $R = 0.49\%$ for the $u = 50^m/_s$

simulation. This observation supports the hypothesis that the sound speed is indeed incorrectly calculated by *ANSYS CFX* for isothermal simulations, and the mean flow makes up for it.

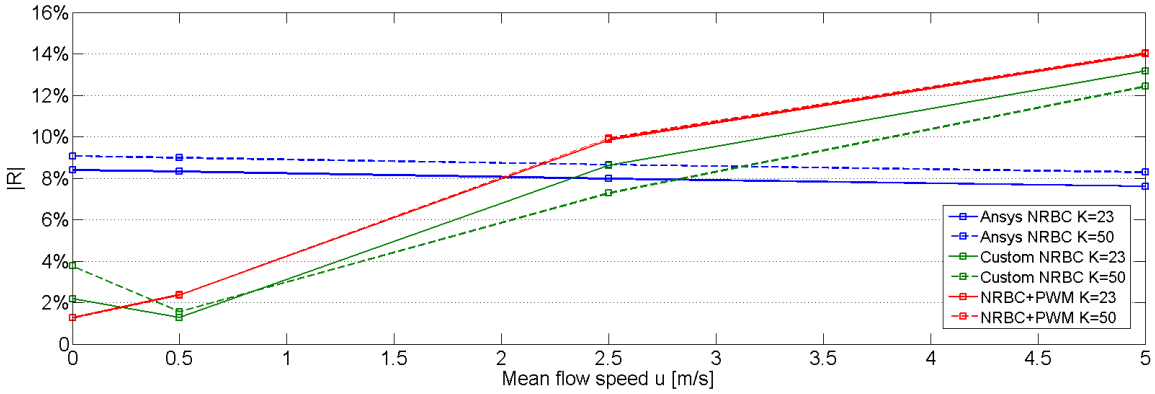


Figure 4.11: Influence of mean flow on the reflection coefficient for isothermal simulations

In Figure 4.12 the reflection coefficient R as a function of mean flow u is shown for the total energy simulations. The custom NRBC and PWM boundary conditions show a comparable trend and values with respect to the isothermal case. The reflection coefficient R of the *ANSYS CFX* NRBC increases only slightly with mean flow, outperforming the custom NRBC and PWM boundaries when mean flow is introduced.

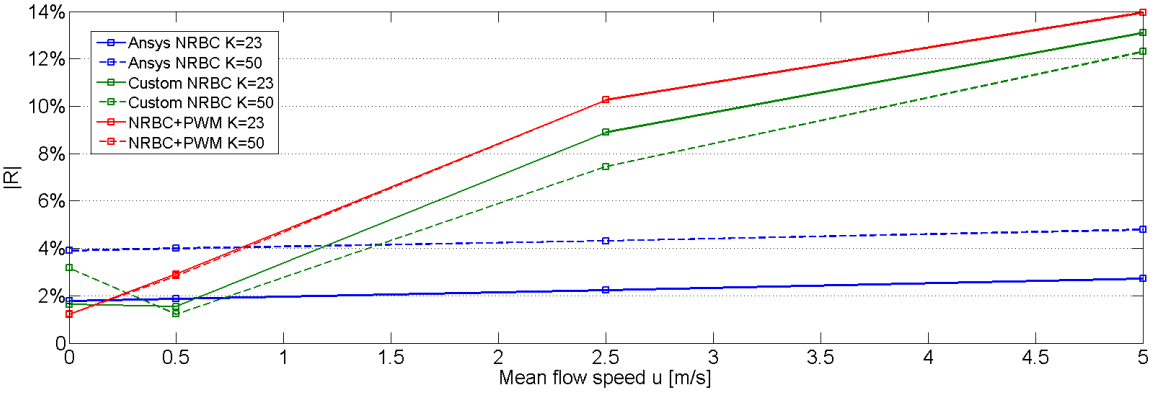


Figure 4.12: Influence of mean flow on the reflection coefficient for total energy simulations

The custom NRBC increases at approximately the same rate as the custom NRBC including PWM, but the reflection coefficient of the custom NRBC exhibits an unexpected minimum at $u = 0.5^m/_s$, resulting in the custom NRBC including PWM to have the highest reflection coefficient for simulations with $u = 2.5^m/_s$ and $u = 5.0^m/_s$ mean flow. Possibly an error is introduced in the custom NRBC method that inadvertently is corrected by a small positive mean flow, of which a slightly inaccurate sound speed seems a plausible cause.

The reason behind the significantly sharper increase of the reflection coefficient of the custom NRBC and PWM boundary conditions compared to the *ANSYS CFX* NRBC is unclear. The sound speed is adjusted for the mean drift in the governing equations of the NRBCs via $c = c_0 + u_0$. It might be the case that higher mean flow velocities influence the retrieved pressure and velocity gradients in an unfavorable manner. The "dip" being only present in the custom NRBC, and not the PWM boundary condition, suggests that the sharp increase is not caused by the LRM term, as the pressure difference due to acoustics is masked. On the other hand, the minimum at $u = 0.5^m/_s$ might be caused by the LRM method, as it is not present in the PWM boundary condition.

The custom NRBC including PWM showed a very small difference between $K = 23s^{-1}$ and $K = 50s^{-1}$, which is expected because the LRM term is effectively masked. See the results of section 4.1.3 for more details.

4.1.4.2 Pressure drift

Not only the reflection coefficient increases with increasing mean flow, also the pressure drift is influenced considerably. For comparison of simulations between simulations with and without mean flow, pressure drift graphs similar to Figure 4.8 are created. The pressure drift without mean flow is subtracted from the pressure drift at mean flow using the following formula: $\Delta|P| = |P_{drift-flow}| - |P_{drift-u=0}|$. A positive value means that it performs worse compared to the pressure drift without mean flow.

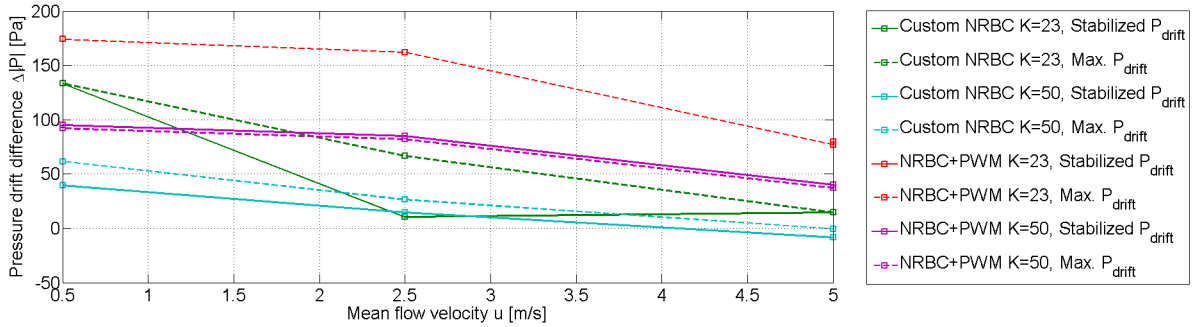


Figure 4.13: Pressure drift vs. mean flow velocity, isothermal simulations, $K = 23s^{-1}$ and $K = 50s^{-1}$, $\Delta|P| = |P_{drift-flow}| - |P_{drift-u=0}|$

The pressure drift difference for the isothermal simulations is shown in Figure 4.13. Practically all values are positive, which means that the pressure drift is larger. The trend is decreasing with mean flow velocity, in contrary to what is observed with the reflection coefficient. The cause might be sought in the propagation of pressure information from the boundary inward. As the reflection coefficient increases with mean flow velocity, so does propagation of pressure information from the boundary to the interior, and thus causing a decrease in pressure drift (i.e. the problem is more "well-posed").

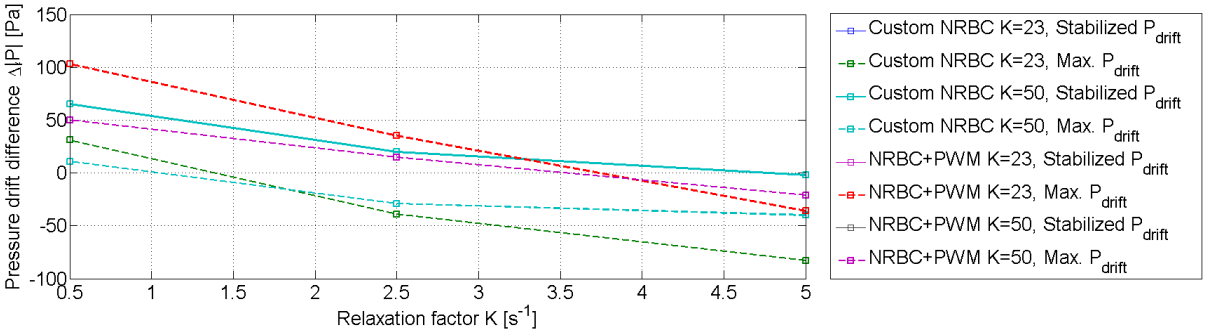


Figure 4.14: Pressure drift vs. mean flow velocity, total energy simulations, $K = 23$ and $K = 50$, $|\Delta P| = |P_{drift-flow}| - |P_{drift-u=0}|$

In Figure 4.14 the pressure drift difference for the total energy simulations are shown. The difference is smaller, and even becomes negative for higher mean flow velocities. The pressure drift for isothermal simulations without mean flow are relatively low compared to the total

energy simulations (see sec. 4.1.3). The higher pressure drift difference for the isothermal simualations therefore work out to be quite similar to the total energy simulations. The conclusion that can be drawn based on these observations is that the isothermal simulations are more sensitive to mean velocity compared to a simulation without mean velocity, but performs very similar in absolute pressure drift.

4.1.5 Partitioning

Partitioning is often used in computational fluid dynamics to speed up simulations by enabling the numerical problem to be divided into different sections (partitions), of which each can run on a single processor core. This enables multiple processor cores to be assigned to solving the numerical problem, instead of just a single processing core. The influence of partitioning is tested for the *ANSYS CFX* NRBC, and the custom NRBC, both including and excluding the PWM term. The simulations are run serial, on two partitions, and on four partitions. Both the reflection coefficient and the simulation time is measured. For the test case, with the given discretization and fluid properties set out in section 3, no difference in the reflection coefficient is observed between simulations. The simulation time did show a difference between simulations, of which the speed-up is shown in Figure **??**. Using two partitions speeds up the simulation about 1.6 times, while using four partitions is actually slower than using two partitions, with a speed-up of 1.38. Assembling and disassembling partitions after each timestep will cost extra simulation time, for such a small domain of 1000 elements this will probably take more effort than the gain in processing power [23]. Although the benefit of partitioning on the speed-up of simulations is slim, the custom NRBC can handle multiple partitions without any performance hit or stability problems.

4.1.6 Linear refinement near boundary

Preliminary simulations showed an influence of the spatial discretization on the accuracy of the returned gradients for pressure and velocity by *ANSYS CFX*. As incorrectly returned gradients by *ANSYS CFX* could be the cause of pressure drift, the influence of increasing the mesh resolution near the NRBC on the reflection coefficient and pressure drift is studied.

4.1.6.1 Difference in reflection coefficient

Using the linearly refined mesh near the NRBC discussed in section 3.1.2, and the total energy equation as the heat transfer equation, the difference in reflection coefficient is determined. As the difference in reflection coefficient R is generally small, it will not be clearly visible in the logarithmic plots such as 4.4. Therefore the difference is plotted in Figure 4.15. The difference is defined as $|\Delta R| = |R_{linref}| - |R|$. A negative value means the linearly refined mesh performs better than the homogeneous mesh, and vice versa.

From the plot (Fig. 4.15) it can be seen that the *ANSYS CFX* NRBC profits slightly from the refined mesh. The custom NRBC benefits from the refinement between $K = 23^1/_s$ and $K = 500^1/_s$, while the custom NRBC including the PWM term is negatively influenced by the refined mesh. It could be that the gradual refinement influences sound speed in the linearly defined section of the domain because mesh size is not equal for each element, which causes the arrival of the characteristic wave f from the sample plane to the boundary to be incorrectly approximated, and therefore influencing the custom NRBC including PWM negatively.

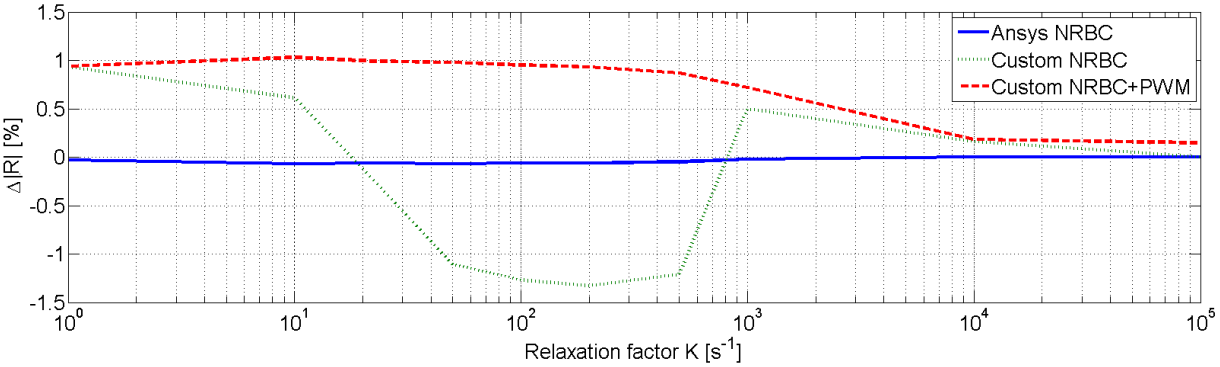


Figure 4.15: Difference in reflection coefficient between the homogeneous mesh and the linearly refined mesh

Another potential source of increased reflection is the increased acoustic CFL number. For the normal mesh, the CFL number is $CFL_{norm} = 0.865325$, while the CFL number is five times higher near the boundary for the linearly refined mesh: $CFL_{ref} = 4.326625$. This is because the mesh size Δx is five times smaller near the boundary, while the timestep Δt has not changed. The increased CFL number might cause decreased accuracy and stability near the boundary, especially in an explicit time marching scheme, such as is used in the custom NRBC. Decreasing the timestep Δt might solve this problem, at the cost of simulation time. See section 3.1.2 for more information about the CFL number.

A simulation run with a relaxation factor of $K = 50s^{-1}$, total energy, and using the custom NRBC, but with a timestep of $\Delta t = 2 \cdot 10^{-6}$, showed a slight decrease in the reflection coefficient from $R = 2.07\%$ to $R = 1.8\%$. The pressure drift was hardly influenced by the increased temporal resolution, in the order of a few Pascal.

4.1.6.2 Pressure drift

In Figure 4.16 the pressure drift difference is shown as a function of the relaxation factor K. The pressure drift difference ΔP is defined in section 4.1.4. While the reflection coefficient in the previous section showed mixed results, the pressure drift has significantly improved over the homogeneous mesh in section 4.1.3, for most relaxation factors. The only exception is the custom NRBC including PWM for a relaxation factor higher than $K = 1000s^{-1}$.

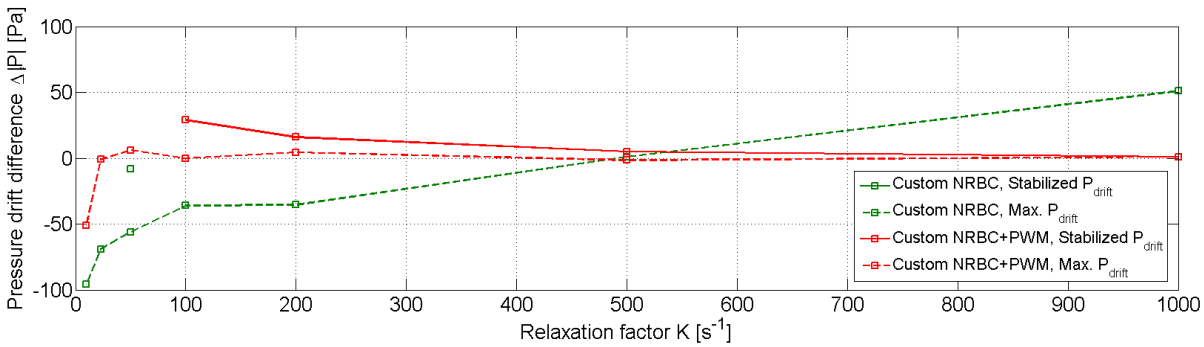


Figure 4.16: Difference in pressure drift between the homogeneous mesh and the linearly refined mesh

The decreased pressure drift is probably due to a decreased discretization error of the one-

sided finite difference approximation to the first spatial derivative of the pressure and velocity at the boundary. In conclusion, the increased mesh resolution near the non-reflective boundary is only significantly helpful for the custom NRBC. The custom NRBC including PWM and the *ANSYS CFX* NRBC are hardly influenced, or even negatively influenced.

4.1.7 Sharp gradients in pressure and velocity

The wavefront of the sine has an infinitely sharp jump in the theoretical solution of the gradient, from $0^{Pa}/_{m}$ to $371.86^{Pa}/_{m}$ (see sec. 3.2.2). Such infinitely sharp jumps are often not accurately represented by numerical methods, as such jumps cause numerical wiggles in the solution and are usually mitigated by flux limiters in CFD [21]. The LODI-relations (see eq. 2.20) use the pressure and velocity gradients to construct the amplitude variation $\mathcal{L}_5$. Thus potential problems can be expected when the wave arrives at the boundary. The pressure gradients are plotted at the sample plane location ($\Delta d = 0.4m$ from the boundary) and at the designated NRBC in Figure 4.17. The "jump" slightly after $0.01s$ is "smoothed", overshoots by approximately 19.6%, and exhibits numerical wiggles that damp out after approximately $0.002s$. The small "bumps" near zero can be explained by the governing sine being at its maximum amplitude, and therefore more prone to error and noise in the gradients.

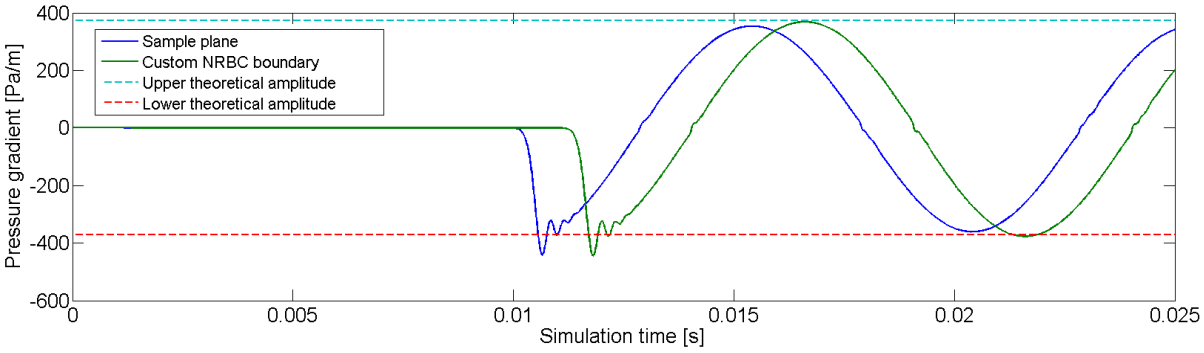


Figure 4.17: Pressure gradient for the normal sine velocity excitation, at the sample plane and custom NRBC boundary

The hypothesis is that these jumps might initiate pressure drift (see sec. 4.1.3), as it seems that pressure drift starts when the traveling wave hits the boundary. This is tested using a smooth transition at the wave-front, as defined in section 3.5.4. The resulting pressure gradients at the sample plane location and custom NRBC boundary is shown in Figure 4.18, from which it can be seen that the transition is much smoother and continuous. A wiggle at around $t = 0.014s$ is observed, which might be due to the theoretical gradient not being smooth (but continuous) at that point (see sec. 3.5.4). The second derivative is discontinuous, which apparently has an influence on the solution, but the wiggly behavior is far less pronounced than for the normal sine.

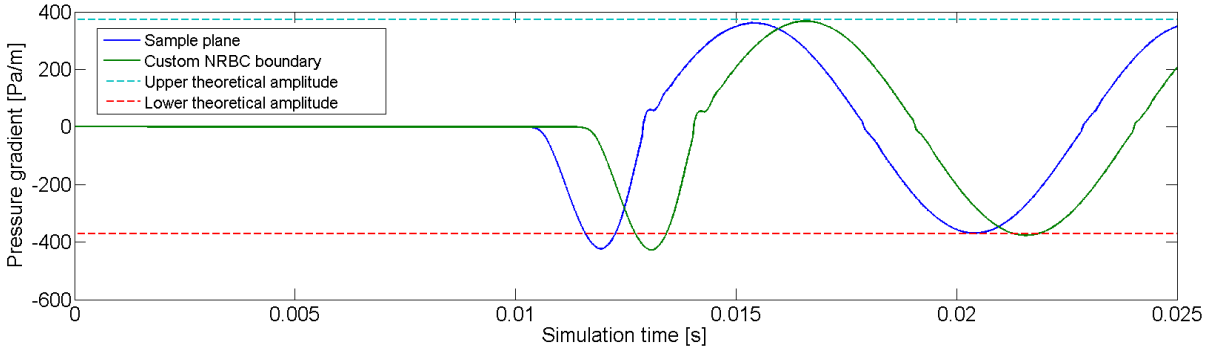


Figure 4.18: Pressure gradient for the smooth sine velocity excitation, at the sample plane and custom NRBC boundary

4.1.7.1 Difference in reflection coefficient

Using the normal sine results, and the total energy equation as the heat transfer equation, the difference in reflection coefficient is determined. A similar definition for the reflection coefficient difference is used as in section 4.1.6, in this case: $\Delta|R| = |R_{smooth}| - |R_{normal}|$. Where a negative difference means that the smooth sine performs better than the normal sine.

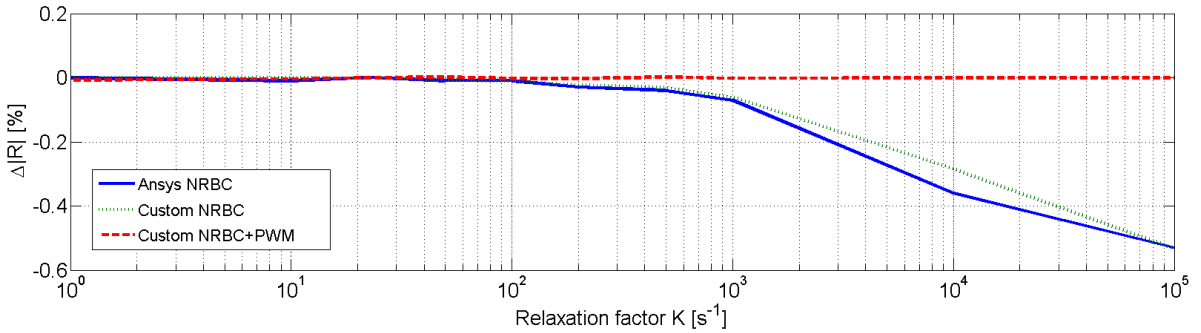


Figure 4.19: Difference in reflection coefficient between the smooth sine and the normal sine, $\Delta|R| = |R_{smooth}| - |R_{normal}|$

The plot in Figure 4.19 shows the reflection coefficient difference $\Delta|R|$ as a function of the relaxation factor K, for all tested NRBC boundaries. The smooth sine shows lower reflections for higher relaxation factors for both the *ANSYS CFX* NRBC and the Custom NRBC. The PWM NRBC shows a different story, and hardly profits from a smooth sine. For the high relaxation factors the reflection is already close to 1, which reduces the practical usability of a smooth sine.

4.1.7.2 Pressure drift

In Figure 4.20 the pressure drift difference is shown as a function of the relaxation factor K. The pressure drift difference ΔP is defined in section 4.1.4. For the PWM NRBC the pressure drift hardly differs from the normal sine case, while the custom NRBC is definitely influenced. For relaxation factors of $K = 23$ and $K = 50$ the maximum pressure drift difference is negative, meaning that the pressure overshoots less. For very high relaxation factors, $K > 500$, the custom NRBC performs worse for the smooth sine. This contradicts the hypothesis that pressure drift is caused by "discontinuity" introduced by the normal sine. The reason behind this observation is not clear, but in practice one would not use such high relaxation factors

with the custom NRBC because it is almost fully reflecting, and therefore not encounter such pressure drift.

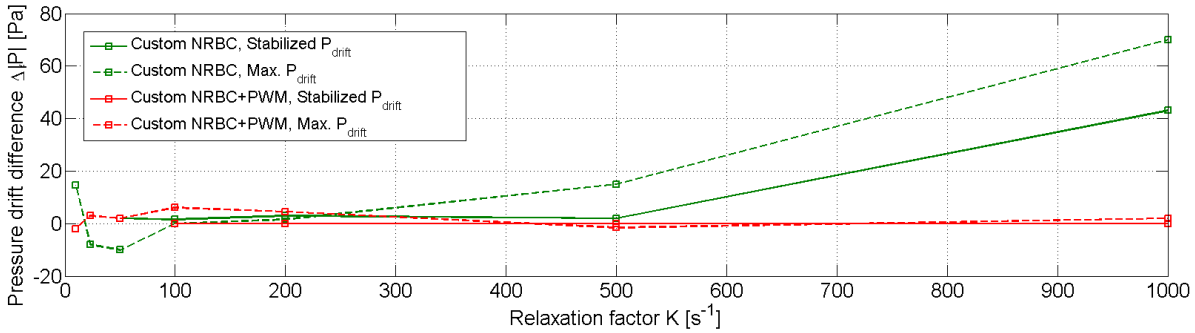


Figure 4.20: Difference in pressure drift of the smooth sine compared to the normal sine, $K = 50$, Total energy, $\Delta|P| = |P_{drift-flow}| - |P_{drift-u=0}|$

In conclusion, it does not seem very helpful to use a smooth sine to prevent drifting or decrease the reflection coefficient.

4.2 Time-domain impedance boundary condition

It is difficult to compare the TD-IBC directly to the other NRBC's discussed and tested in this paper, because a TD-IBC can influence the phase angle ϕ of the complex reflection coefficient R. However, three cases are directly comparable. The non-reflective TD-IBC to the *ANSYS CFX* NRBC and the custom NRBC both including and excluding PWM, the acoustically hard wall TD-IBC to a hard wall in *ANSYS CFX*, and the pressure release surface with a constant pressure outlet in *ANSYS CFX*. Furthermore a complex reflection coefficient of $R = 0.5 + 0.5i$ is tested against theory.

The TD-IBC test cases are all tested using a relaxation factor $K = 50s^{-1}$. The relaxation factor of $K = 50s^{-1}$ is seen as an optimal value for the custom NRBC (see sec. 2.4), and therefore also used throughout all previous test cases, which makes comparison more easily done. Other values for the relaxation factor could also have been chosen, as the TD-IBC uses PWM that causes the reflection coefficient R to be basically "decoupled" from the relaxation factor K (see sec. 4.1.3). The influence of the relaxation factor K is thus expected to be small.

Two figures of merit are introduced to quantify the performance of the test cases, which are the normalized absolute difference with theory $\Delta|\bar{R}|$ (eq. 4.1), and the phase difference $\Delta\phi$ between the measured and theoretical reflection coefficient (eq. 4.2).

$$\Delta|\bar{R}| = \frac{|R_{measured}| - |R_{theory}|}{|R_{theory}|} \tag{4.1}$$

$$\Delta\phi = \phi_{measured} - \phi_{theory} \tag{4.2}$$

For all TD-IBC simulations no pressure drift is observed, irrespective of the relaxation factor K used, which can probably be attributed to the manner in which the amplitude variation $\mathcal{L}_5$ is determined (see sec. 3.3.9.2). This method does not rely on the pressure and velocity gradients on the boundary, but uses an expression for characteristic wave f determined at a sample plane near the boundary. This supports the hypothesis that the pressure drift in the custom NRBC and PWM NRBC originates in the determination of the one-sided pressure and

velocity gradients near the boundary.

4.2.1 Acoustically hard wall and pressure release surface

The theoretical reflection coefficients of both a hard wall and a pressure release surface are compared to the measured reflection coefficient of aforementioned boundaries in Table 4.1. Theoretically both boundary conditions have an absolute reflection of $|R| = 1$, but the phase of the acoustically hard wall is $\phi = 0$ rad while the pressure release surface has a phase of $\phi = \pm\pi$ rad.

Table 4.1: Reflection coefficient results for acoustically hard and pressure release surfaces

Boundary condition	$\lvert R\rvert$	ϕ [rad]	$\Delta\lvert\bar{R}\rvert$	$\Delta\phi$ [rad]
Total Energy Calculations				
Acoustically hard wall TD-IBC	1.0069	-0.0818	0.69%	−0.0818
Wall *ANSYS CFX*	0.9998	-0.1036	−0.02%	−0.1036
Pressure release surface TD-IBC	0.9675	3.1057	−3.25%	−0.0359
Constant pressure outlet *ANSYS CFX*	0.9775	-3.1020	−2.25%	0.0396
Isothermal Calculations				
Acoustically hard wall TD-IBC	1.0114	0.0617	1.14%	0.0617
Wall *ANSYS CFX*	1.0151	0.0433	1.51%	0.0433
Pressure release surface TD-IBC	0.8779	3.0893	−12.21%	−0.0523
Constant pressure outlet *ANSYS CFX*	0.9661	3.0585	−3.39%	−0.0831

The measured complex reflection coefficient for the TD-IBC boundaries are close to the theoretical values, for both absolute reflection coefficient R as well as the phase ϕ. The "hard" (Dirichlet) constant pressure and constant velocity (hard wall) boundaries in *ANSYS CFX* show errors in the same order of magnitude. The error for the TD-IBC is small and comparable to constant pressure and velocity boundaries. Pressure drifting is not an issue with constant pressure and constant velocity boundaries.

4.2.2 Non-reflecting outlet

The reflection of the non-reflective TD-IBC is compared to the *ANSYS CFX* NRBC, custom NRBC, and custom NRBC including PWM for both a value of $K = 50s^{-1}$, and for the best reflection values in Table 4.2.

Table 4.2: Absolute reflection coefficient $|R|$ for different non-reflecting outlets

	ANSYS CFX NRBC	Custom NRBC	Custom NRBC+PWM	TD-IBC
Total E	3.90%/0.12%	3.18%/1.16%	1.21%/0.08%	1.00%
Isothermal	9.07%/8.16%	3.76%/1.31%	1.26%/0.61%	0.45%
$K = 50s^{-1}$ / K_{best}				

The nonreflective TD-IBC shows a lower reflection coefficient compared to the other NRBCs for a reflection coefficient of $K = 50s^{-1}$. Taking the best values of the different models, the TD-IBC is generally within the same order of error.

4.2.3 Complex reflection

The complex reflection of $R = 0.5 + 0.5i$ is fitted to one frequency ($f = 100Hz$), as the designed complex curve fitted filter coefficient sets of high order proved unstable (see sec. 3.5.5). Theoretically the absolute reflection coefficient is $|R| = 0.5\sqrt{2}$, and the phase angle is $\phi = 0.25\pi$ rad.

Table 4.3: Reflection results for complex $R = 0.5 + 0.5i$

Heat transfer	f [Hz]	$\lvert R\rvert$	ϕ [rad]	$\Delta\lvert\bar{R}\rvert$	$\Delta\phi$ [rad]
Total Energy	100	0.7146	0.7520	0.75%	−0.0333
Isothermal	100	0.6994	0.8010	0.77%	0.0156

The results are given in Table 4.3. The measured complex reflection comes very close to the designed complex reflection coefficient. The isothermal simulation showed a slightly smaller error than the total energy equations. In conclusion, it is indeed possible to implement an accurate and stable TD-IBC for a single frequency.

Chapter 5

Conclusions and Recommendations

The Time-Domain Impedance Boundary Condition (TD-IBC) based on theory set out by Polifke et al [9][6] has been implemented successfully for a numerical problem where a single frequency is dominant. No stability problems were observed, and pressure drift was absent for all TD-IBC simulations. Different boundaries have been simulated, among which are the acoustically hard wall, the pressure release surface, a non-reflecting outlet, and a complex reflection of $R = 0.5 + 0.5i$. The last mentioned boundary showed a small difference in the absolute reflection coefficient with theory of $|\Delta\bar{R}| = 0.75\%$, and a phase angle difference with theory of $\Delta\phi = -0.0333$ rad. In the other simulations the absolute reflection coefficient differed with theory ranging from $|\Delta\bar{R}| = 0.69\%$ for an acoustically hard wall using the total energy equation to $|\Delta\bar{R}| = 12.21\%$ for a pressure release surface in an isothermal simulation.

If the impedance or complex reflection coefficient is defined at multiple frequencies, finding the correct filter coefficients to simulate the complex reflection coefficient can become a cumbersome and daunting task. No stable filter configurations have been found for broadband reflection coefficients. Thermoacoustic phenomena in gas-turbines are examples were a broadband reflection coefficient could be used. If the TD-IBC is to be used in broadband applications in the future, more research on finding stable filter coefficients must be done. Fortunately, thermoacoustic devices usually have one dominant operating frequency, and other frequencies are often negligible.

The non-reflective part of the TD-IBC, called the custom Non-Reflective Boundary Condition (NRBC), can work independently from the TD-IBC. In addition, a Plane-Wave Masking (PWM) term is added to the custom NRBC to improve low-frequency behavior, as is proposed by Polifke [16]. These two boundary conditions were tested against theory, and a built-in NRBC by *ANSYS CFX*. For all three NRBCs a low reflection coefficient is attainable in problems without mean flow, but all have distinct peculiarities. The *ANSYS CFX* NRBC showed abysmal performance regarding the reflection coefficient in isothermal simulations, which could be traced back (using mean flow simulations) to an incorrectly assumed isothermal sound speed by *ANSYS CFX*. The custom NRBC seems to "bottom out" at lower relaxation factors to $R \approx 1.3\%$, while the *ANSYS CFX* NRBC shows a minimum of $R = 0.12\%$, and the PWM NRBC even has a mimimum of $R = 0.07\%$.

Another observation is that the custom NRBC and the custom NRBC+PWM both showed considerable pressure drift, especially at lower relaxation factors, while the *ANSYS CFX* NRBC never showed any significant pressure drift. The pressure drift seems to decrease either when the PWM term is used, or when the simulation is run isothermal. When both parameters are set, the pressure drift is reduced to practically zero. The fact that the isothermal simulations show much less pressure drift suggests that the (varying) temperature is a contributor to the pressure drift.

The mean temperature also increases by $\Delta T = 0.175K$ in total energy simulations, which alter the mean density ρ_0 and small signal sound speed c_0, that are determined at initialization. These quantities, that remain constant during the simulation, are used as factors in the definition of the amplitude variation $\mathcal{L}_5$. $\mathcal{L}_5$ is used in the custom NRBC, both including and excluding PWM, to determine the pressure at the next time-step via time-integration of the LODI relations. The slight error introduced in the mean density and small signal sound speed might cause a small error in the temporal derivative of the pressure in the LODI relations, which, after time-integration, builds up every time-step and induces pressure drift. Three other explanations of pressure drift could be that the *ANSYS CFX* NRBC has a built-in "mechanism" to prevent pressure drift, uses an algorithm to update the mean density and small signal sound speed, or uses a higher order finite difference expression at the boundaries to construct the pressure and velocity gradients. The gradients that are used by the custom NRBC are returned by the *ANSYS CFX* solver, and determined using a one-side finite difference expression.

Based on the aforementioned explanations on pressure drift, it is recommended to run simulations with a built-in algorithm to update the mean density and small signal sound speed during simulation. This might potentially reduce pressure drift.

When simulations are run using a non-zero mean flow velocity, the custom NRBC and PWM NRBC show a sharp increase in the reflection coefficient, while the *ANSYS CFX* NRBC showed much less sensitivity to mean flow velocity. A possible explanation could be that the returned pressure and/or velocity gradients are negatively influenced by the mean flow. Mean flow is, however, less important for thermoacoustic applications.

Two optimizations were investigated to potentially positively influence the reflection coefficient and pressure drift. The first being a linear increase in mesh resolution towards the NRBC in question, the other being a smoothed sine in the excitation velocity with a continuous first spatial derivative. The linear refined mesh showed mixed results regarding the reflection coefficient, which might be related to the increased CFL number due to the smaller mesh size Δx. The pressure drift on the other hand decreased, especially at lower relaxation factors. For the smooth sine simulations the reflection coefficient slightly decreased for higher relaxation factors for both the custom NRBC and the *ANSYS CFX* NRBC, while the PWM NRBC was hardly influenced. The pressure drift on the other hand was, unexpectedly, increased by the smooth sine. This contradicts the postulated hypothesis that the pressure drift is instigated by the discontinuity in the pressure and velocity gradient at the wave front.

A non-reflective version of the TD-IBC is also tested, which differs from the other NRBCs in the expression for the outgoing amplitude variation $\mathcal{L}_5$ (see sec. 2.3.2). This non-reflective TD-IBC showed a reflection coefficient of $R = 1\%$ for the total energy equation, and unlike the custom NRBC and PWM NRBC, does not show any pressure drift. For all NRBCs, including the non-reflective TD-IBC, a reflection coefficient in the order of $R = 1\%$ to $R = 3\%$ is attainable using the correct parameters.

All boundary conditions have been tested using a one-dimensional problem with purely plane waves. What has not been done is extensive analysis of two-dimensional and three-dimensional domains, only the correct working of the NRBCs and TD-IBC have been verified for these meshes. The boundary conditions have not been tested against other, more complex, numerical problems. It is therefore recommended that the boundary conditions are tested in other, more demanding, numerical problems to test their performance and robustness.

Other CFD software could use better approximations to the pressure and velocity gradients at the boundary, and differ in other fundamental aspects relevant to the application of the

developed boundary conditions. Implementing the TD-IBC in other CFD packages, and assessing its stability and performance, is therefore recommended. Also, *ANSYS FLUENT* uses the *C++* programming language, which is generally considered more user friendly compared to the antiquated *FORTRAN77* language *ANSYS CFX* uses.

Although the TD-IBC is only extensively tested in a very idealized numerical problem, the results seem very promising for application in thermoacoustic CFD simulations in which a single frequency is dominant. In this regard, subsequent testing in more demanding and complex simulations to characterize the behavior of the TD-IBC in such settings is recommended.

Bibliography

[1] ANSYS ICEM User Manual. pages 724–746, 2009.

[2] I.C. Sheng A.N. Norris. Acoustic Radiation from a Circular Pipe with an Infinite Flange. *Journal of Sound and Vibration*, 135:85–93, 1989.

[3] S Bühler and T.H. van der Meer. Numerical thermoacoustic experiment for verification of commercial CFD code. 2009.

[4] K. Fung and J. Hongbin. Impedance and Its Time-Domain Extensions. *AAIA*, 38(1), 2000.

[5] K.Y. Fung and J. Hongbin. Time-domain Impedance Boundary Conditions for Computational Acoustics and Aeroacoustics. *International Journal of Computational Fluid Dynamics*, 18(6):503–511, August 2004.

[6] A. Huber, P. Romann, and W. Polifke. Filter-Based Time-Domain Impedance Boundary Conditions for CFD Applications. *Proceedings of ASME Turbo Expo*, 2008.

[7] P. in 't Panhuis. *Mathematical Aspects of Thermoacoustics*. 2009.

[8] M.G. Jones and T.L. Parrot. Evaluation of a Multi-Point Method for Determining Acoustic Impedance. *Mechanical Systems and Signal Processing*, 3:15–35, 1989.

[9] R. Kaess, A. Huber, and W. Polifke. A Time-Domain Impedance Boundary Condition for Compressible Turbulent Flow. *American Institute of Aeronautics and Astronautics*, pages 1–15, 2008.

[10] H. Levine and J. Schwinger. On the Radiation of Sound from an Unflanged Circular Pipe. *Physical Review Letters*, 73(4):24, 1948.

[11] E.C. Levy. Complex-Curve Fitting. pages 37–43, 1959.

[12] E.H. Lysen. The Trias Energica : Solar Energy Strategies for Developing Countries. *Eurosun Conference*, pages 16–19, 1996.

[13] Matlab. *Matlab Manual*.

[14] J. Oosterhuis. Performance of non-reflective boundary condition in ANSYS CFX. Technical report, 2012.

[15] T.J. Poinsot. Boundary conditions for direct simulations of compressible viscous flows. *Journal of Computational Physics*, 99(2):352, April 1992.

[16] W. Polifke, C. Wall, and P. Moin. Partially reflecting and non-reflecting boundary conditions for simulation of compressible viscous flow. *Journal of Computational Physics*, 213(1):437–449, March 2006.

[17] L. Selle. The actual impedance of non-reflecting boundary conditions : implications for the computation of resonators. pages 1–21, 2003.

[18] J. Seung-ho and I. Jeong-guon. On the multiple microphone method for measuring in-duct acoustic properties in the presence of mean flow. 103(3):1520–1526, 1998.

[19] P. Stopford. E-Mail communication with ANSYS. pages 1–3, 2013.

[20] J.W. Strutt Baron Rayleigh. *The Theory Of Sound*. MacMillan and Co, 1877.

[21] C.K.W. Tam. Computational Aeroacoustics : Issues and Methods. 33(10), 1995.

[22] W. Thompson. Time- Dependent Boundary Conditions for Hyperbolic Systems , II. *Journal of Computational Physics*, 439461:439–461, 1990.

[23] Ansys Tutorials. ANSYS CFX-Solver Modeling Guide. 15317(November):724–746, 2011.

[24] C.H. Venner. On the significance of the Laplace equation and the advection diffusion equation as model problems for Computational Fluid Dynamics. pages 1–5, 2012.

[25] J. Ward, B. Clark and G. Swift. Design Environment for Low-amplitude Thermoacoustic Energy Conversion DeltaEC Users Guide. 2012.

[26] P. Zwart. Non-reflective Boundary Conditions in CFX-5. 2005.

Nomenclature

CEL	Command Expression Language	
CFD	Computational Fluid Dynamics	
CFL number	Courant-Friedrichs-Lewy number	
FIR	Finite response filter	
IIR	Infinite response filter	
LODI	Locally One-Dimensional Inviscid	
LRM	Linear Relaxation Method	
MMM	Two-Microphone Method	
MMS	Memory Management System	
MPI-CH	Message Passing Interface CHameleon	
NRBC	Non-Reflecting Boundary Condition	
NSCBC	Navier-Stokes Characteristic Boundary Conditions	
ODE	Ordinary Differential Equation	
PDE	Partial Differential Equation	
PVM	Parallel Virtual Machine	
PWM	Plane-Wave Masking	
PWM	Plane-Wave Masking	
SF	Singularity Factor	
SPMD	Single-Program Multiple-Data	
STP	Standard Temperature and Pressure	
SWR	Standing Wave Ratio	
TD-IBC	Time-Domain Impedance Boundary Condition	
TMM	Two-Microphone Method	
E	Total specific energy	$[J/kg]$
K	Relaxation factor	$[1/s]$

L	Characteristic length of domain	$[m]$
P	Pressure	$[Pa]$
Q	Heat (energy)	$[J]$
Re	Reynolds number	$[-]$
T	Temperature	$[K]$
U	Internal energy	$[J]$
W	Work (energy)	$[J]$
Z	Impedance	$[Rayl]$
Z_0	Characteristic impedance	$[Rayl]$
Δt	Time increment	$[s]$
Ω	Domain space	
$\bar{\bar{I}}$	Identity tensor	
$\bar{\bar{\tau}}$	Stress tensor	$[N/m2]$
$\delta\hat{p}$	Acoustic pressure amplitude	$[Pa]$
$\delta\hat{p}^+$	Amplitude of forward traveling acoustic pressure wave	$[Pa]$
$\delta\hat{p}^-$	Amplitude of backward traveling acoustic pressure wave	$[Pa]$
$\delta\hat{p}^{tr}$	Amplitude of transmitted acoustic pressure wave	$[Pa]$
$\delta\hat{u}$	Acoustic velocity amplitude	$[m/s]$
δp	Acoustic pressure	$[Pa]$
δu	Acoustic velocity	$[m/s]$
$\dot{Q}$	Volumetric heat source	$[J/m^3 s]$
γ	Temperature	$[K]$
$\hat{f}$	Amplitude of characteristic wave f	$[m/s]$
$\hat{g}$	Amplitude of characteristic wave g	$[m/s]$
$\hat{p}_x$	Amplitude of pressure gradient	$[Pa/m]$
$\hat{u}_x$	Amplitude of velocity gradient	$[m/s/m]$
λ_i	Characteristic speed	$[m/s]$
$\langle ... \rangle$	Instantaneous area average of variable	
$\mathbf{A}^+$	Moor-Penrose pseudo-inverse of $\mathbf{A}$	
$\mathbf{J}$	Column vector of source terms	
$\mathbf{U}$	Column vector of conserved quantities	

$\mathbf{U}^p$	Column vector of primitive quantities	
$\mathcal{L}_i$	Amplitude variations of characteristic waves	$[varies]$
$\mathcal{M}$	Mean flow Mach number	$[-]$
ω	Angular frequency	$[^{rad}/_{s}]$
ω_n	Normalized angular frequency (w.r.t. grid)	$[^{rad}/_{rad}]$
$\partial\Omega$	Domain boundary	
ϕ	Velocity potential	$[^{m^2}/_{s}]$
ψ	Phase of reflection coefficient R	$[rad]$
ρ	Density	$[^{kg}/_{m^3}]$
ρ_0	Mean density	$[^{kg}/_{m^3}]$
σ	Coupling parameters in LRM method	$[^{m}/_{s}]$
τ	Time coordinate in convolution	$[s]$
$\vec{\mathbf{F}}^{inv}$	Inviscid flux vector	
$\vec{\mathbf{F}}^{visc}$	Viscid flux vector	
$\vec{f}$	Volumetric force field	$[^{N}/_{kg}]$
$\vec{q}$	Heat conduction	$[^{J}/_{m^2 s}]$
$\vec{u}$	Velocity vector	$[^{m}/_{s}]$
$\vec{u}_{\partial v}$	Velocity vector of boundary	$[^{m}/_{s}]$
a_i	Filter coefficients of f	$[-]$
b_i	Filter coefficients of g	$[-]$
c	Sound speed	$[^{m}/_{s}]$
c_0	Characteristic sound speed	$[^{m}/_{s}]$
d	Space coordinate used in MMM	$[m]$
e	Internal energy	$[^{J}/_{kg}]$
f	Forward traveling characteristic wave amplitude	$[^{m}/_{s}]$
f_s	Sampling frequency	$[^{1}/_{s}]$
f_{in}	Excitation frequency at inlet	$[^{1}/_{s}]$
g	Backward traveling characteristic wave amplitude	$[^{m}/_{s}]$
$g*$	External perturbation	$[^{m}/_{s}]$
k	Angular wave number	$[^{rad}/_{m}]$
p	Pressure	$[Pa]$

p'	Turbulent contribution to pressure	$[Pa]$
p_0	Mean pressure	$[Pa]$
p_∞	Pressure at infinity	$[Pa]$
t	Time coordinate	$[s]$
u	Velocity in x-direction	$[m/s]$
u'	Turbulent contribution to velocity	$[m/s]$
u_0	Mean velocity	$[m/s]$
v	Specific volume	$[m^3/kg]$
x	Space coordinate	$[m]$
z	Complex argument in z-transform	$[-]$

Appendix

Contents

Chapter 1

Acoustic theory

In this chapter an overview is given on important equations and derivations relevant to acoustics in general.

1.1 Derivation of the small signal sound speed

The sound speed is the speed at which an acoustic wave propagates through a medium. The sound speed for an isentropic process is defined by equation 1.1. To evaluate the sound speed, the pressure p needs to be expressed in terms of the density ρ via an equation of state. The isentropic equation of state used, is the adiabatic gas law (eq. 1.2) and is subsequently expanded by using a Taylor series around the static density ρ_0 (eq. 1.4). The acoustic density $\delta\rho$, pressure δp, and velocity δu are defined in equation 1.3.

$$c^2 \equiv \left.\frac{\partial p}{\partial \rho}\right|_{s=const} \tag{1.1}$$

$$(\frac{p}{p_0}) = (\frac{\rho}{\rho_0})^\gamma \tag{1.2}$$

$$\delta\rho = \rho - \rho_0 \qquad \delta p = p - p_0 \qquad \delta u = u - u_0 \tag{1.3}$$

$$p(\rho_0 + \delta\rho) = p_0 + A(\frac{\delta\rho}{\rho_0}) + \frac{B}{2!}(\frac{\delta\rho}{\rho_0})^2 + \frac{C}{3!}(\frac{\delta\rho}{\rho_0})^3 + \mathcal{O}(\delta\rho^4) \tag{1.4}$$

$$where \quad A = p_0\gamma, \quad B = p_0\gamma(\gamma-1), \quad C = p_0\gamma(\gamma-1)(\gamma-2) \tag{1.5}$$

However, the adiabatic gas law is only valid for perfect gases. In the case of liquids, the coefficients (A,B,C, etc) are determined using experiments or other analysis. By differentiating equation 1.4 an expression for the sound speed is derived (see eq. 1.6). By noting that fluctuations of the density due to acoustics are generally very small, the terms after the constant term $\frac{A}{\rho_0}$ are practically zero in these cases and can be neglected. One arrives at an expression from which the adiabatic bulk modulus can be extracted (eq. 1.7).

$$c^2 = \frac{A}{\rho_0} + \frac{B}{\rho_0}(\frac{\rho-\rho_0}{\rho_0}) + \frac{C}{2\rho_0}(\frac{\rho-\rho_0}{\rho_0})^2 + \mathcal{O}(\delta\rho^3) \tag{1.6}$$

$$c_0^2 = \frac{A}{\rho_0} \qquad \rightarrow \qquad A = \rho_0 c_0^2 \tag{1.7}$$

To construct an expression for c_0 in gases, equation 1.4 is rewritten in form 1.8 with the adiabatic bulk modulus A substituted, and in form 1.9 which is the ideal gas form. When compared, one can identify c_0 (see eq. 1.10).

$$\delta p = c_0^2\delta\rho\left[1 + \frac{B}{2!A}(\frac{\delta\rho}{\rho_0}) + \frac{C}{3!A}(\frac{\delta\rho}{\rho_0})^2 + \mathcal{O}(\delta\rho^3)\right] \tag{1.8}$$

$$\delta p = \frac{\gamma p_0}{\rho_0}\delta\rho\left[1 + \frac{(\gamma-1)}{2!}(\frac{\delta\rho}{\rho_0}) + \frac{(\gamma-1)(\gamma-2)}{3!}(\frac{\delta\rho}{\rho_0})^2 + \mathcal{O}(\delta\rho^3)\right] \tag{1.9}$$

$$c_0 = \sqrt{\frac{\gamma p_0}{\rho_0}} \tag{1.10}$$

Important to mention is the difference between isothermal and adiabatic sound speeds. For isothermal calculations the adiabatic index is $\gamma = 1$ instead of $\gamma = 1.4$, resulting in a lower sound speed as calculated by equation 1.10. In the isothermal case, heat exchange between the wave and the environment need to take place at moments of rarefaction and compression to keep the temperature constant, which is only accurate when the period of the wave is of the same order or longer than the time it takes to cool the medium. Generally this is not the case for acoustic waves on earth, and these compress and expand by approximation adiabatically.

1.2 Derivation of the wave equations

To arrive at the linearized wave equation (eq. ??) from the Euler equations (eq. 1.11), the following assumptions are made:

1. Homogeneous fluid, the fluid composition is uniform throughout the domain.
2. No mass, momentum, and energy source terms.
3. Assuming one dominant direction of propagation (i.e. one-dimensional).
4. In the absence of sound, the fluid is at static pressure, static density, and with a mean flow of zero.
5. Acoustic disturbances are small compared to static pressure, density, and sound speed. $|\delta\rho| \ll \rho_0 \qquad |\delta pitemize| \ll \rho_0 c_0^2 \qquad |\delta u| \ll c_0$

Using the first three assumptions equations 1.12 and 1.13 are constructed based on the mass and momentum Euler equations.

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla}\cdot\vec{\mathbf{F}}^{inv} = \mathbf{J} \tag{1.11}$$

$$\rho_t + (\rho u)_x = 0 \tag{1.12}$$

$$(\rho u)_t + (\rho u^2)_x + P_x = 0 \tag{1.13}$$

Using equation 1.8, and neglecting higher order terms, a relation for acoustic pressure δp can be derived (see eq. 1.14). Expanding the 1D continuity (eq. 1.12) and momentum (eq. 1.13) equations using the product rule, substituting the expression for the density in terms of a static density and the acoustic density (eq. 1.3), and substituting acoustic velocity δu (note: $u_0 = 0$), one arrives at equations 1.15 and 1.16.

$$\delta p = c_0^2\delta\rho \tag{1.14}$$

$$\delta\rho_t + \delta\rho_x\delta u + \rho_0\delta u_x + \delta\rho\delta u_x = 0 \tag{1.15}$$

$$\rho_0\delta u_t + \rho_0\delta u\delta u_x + \delta\rho\delta u_t + \delta\rho\delta u\delta u_x + \delta p_x = 0 \tag{1.16}$$

For equation 1.15 the second and fourth term are products of small signal quantities, and therefor extremely small and can be neglected (i.e. linearized). The same argumentation is valid

for the second, third, and fourth terms in equation 1.16, and can be neglected. This results in a linearized set of equations (see 1.17).

$$\delta\rho_t + \rho_0 \delta u_x = 0 \tag{1.17a}$$
$$\rho_0 \delta u_t + \delta p_x = 0 \tag{1.17b}$$

The usual convention in acoustics is to use pressure and velocity as the primary field variables. Taking the time-derivative of equation 1.14 and combining it with equation 1.17a to eliminate the density, one arrives at equation 1.18.

$$\delta p_t + \rho_0 c_0^2 u_x = 0 \tag{1.18}$$

Using equation 1.17b and 1.18, taking the time- and spatial derivatives, one can choose to eliminate either pressure or velocity and arrive at form 1.19 or 1.20 of the linear wave equation respectively. The three dimensional form is derived analogously, and is often written in scalar velocity potential form (see eq. 1.21). The velocity is defined in equation 1.22.

$$c_0^2 u_{xx} - u_{tt} = 0 \tag{1.19}$$
$$c_0^2 p_{xx} - p_{tt} = 0 \tag{1.20}$$
$$\vec{\nabla}^2 \phi - \frac{1}{c_0^2}\phi_{tt} = 0 \tag{1.21}$$
$$\nabla\phi = \vec{u} \tag{1.22}$$

1.3 Fourier and Laplace Transform

Fourier and Laplace transforms are often used to represent signals and functions in other domains, especially periodic functions. In this report the Fourier transform is used to transform a signal into its Fourier componenents, i.e. frequency content, in the frequency domain. The Laplace transform is related, but somewhat different to the Fourier transform, and is often used to solve equations that are hard to solve in the time domain but relatively easy in the s-domain. The s-domain can be seen as a complex frequency domain. The Fourier and inverse Fourier transforms are given in equations 1.23 and 1.24. The Laplace and its inverse are given in equations 1.25 and 1.26.

$$\bar{u}(x,\omega) = \mathcal{F}[u(x,t)] = \int_{-\infty}^{\infty} u(x,t)\cdot e^{-i\omega t} dt \tag{1.23}$$
$$u(x,t) = \mathcal{F}^{-1}[\bar{u}(x,t)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} \bar{u}(x,\omega)\cdot e^{i\omega t} d\omega \tag{1.24}$$
$$F(s) = \mathcal{L}[f(t)] = \int_{0}^{\infty} f(t)\cdot e^{-st} dt \tag{1.25}$$
$$f(t) = \mathcal{L}^{-1}[F(s)] = \frac{1}{2\pi i}\lim_{T\to\infty}\int_{\gamma-iT}^{\gamma+iT} F(s) ds \tag{1.26}$$

Also discrete forms of the Fourier and Laplace transform exist, which are called the Discrete-Time Fourier Transform (DTFT) and the Z-transform respectively.

Convolution integral in Laplace domain An important property is that convolution in the time-domain is multiplication in the Laplace domain.

Time-domain	Laplace domain
$(f * g)(t) = \int_0^t f(\tau)g(t-\tau)d\tau$	$F(s) \cdot G(s)$

1.4 Derivation of the Helmholtz equation

Many acoustic quantities like impedance and reflection are defined as a function of frequency. However, the wave equation and other time-domain methods cannot use these quantities directly. To be able to use and characterize these quantities a frequency-domain method must be used.

The Helmholtz equation is an equation that works in the frequency domain instead of the time-domain. In steady-state analysis of acoustics, the time-series behave in a predictable periodic manner in time, and can be eliminated by assuming time-harmonic behavior. The time-harmonic behavior assumption used (eq. 1.27) is substituted in the wave equation (eq. 1.21) to obtain, after further derivation, the Helmholtz equation (eq. 1.31). This method of decoupling $\vec{x}$ and t is effectively the method of separation of variables.

For more complex signals a sum of Fourier coefficients can be used via a Fourier transform.

$$\phi(\vec{x}, t) = \phi(\vec{x}, \omega)e^{i\omega t} \tag{1.27}$$

By calculating the gradients and time-derivatives (eq. 1.28), substituting them into the velocity potential form of the wave-equation (eq. 1.21), an equation (eq. 1.29) can be derived from which the time-dependent part can be eliminated by bringing the term outside the brackets (see eq. 1.30). Finally arriving at the Helmholtz equation (eq. 1.31).

$$\vec{\nabla}\phi(\vec{x}, t) = [\vec{\nabla}\phi(\vec{x}, \omega)]e^{i\omega t} \quad \rightarrow \quad \vec{\nabla}^2\phi(\vec{x}, t) = [\vec{\nabla}^2\phi(\vec{x}, \omega)]e^{i\omega t} \tag{1.28a}$$

$$\phi_t(\vec{x}, t) = \phi(\vec{x}, \omega)ie^{i\omega t} \quad \rightarrow \quad \phi_{tt}(\vec{x}, t) = -\phi(\vec{x}, \omega)\omega^2 e^{i\omega t} \tag{1.28b}$$

$$[\vec{\nabla}^2\phi(\vec{x}, \omega)]e^{i\omega t} + [\frac{1}{c^2}\phi(\vec{x}, \omega)\omega^2]e^{i\omega t} = 0 \tag{1.29}$$

$$[\vec{\nabla}^2\phi(\vec{x}, \omega) + \frac{\omega^2}{c^2}\phi(\vec{x}, \omega)]e^{i\omega t} = 0 \qquad \forall t \tag{1.30}$$

$$\vec{\nabla}^2\phi(\vec{x}, \omega) + k^2\phi(\vec{x}, \omega) = 0 \qquad where \quad k = \frac{\omega}{c} \tag{1.31}$$

The Helmholtz equation is a function of position and frequency rather than of position and time like the wave-equation. In the equations, k is the wave number, which can be seen as a spatial frequency. The Helmholtz equation is a PDE of the elliptic type.

1.5 Reflection and impedance

The definition for reflection and impedance are given in equations 1.32 and 1.33 respectively. Where $\hat{p}$ is the acoustic pressure phasor, and $\hat{u}$ is the acoustic particle velocity phasor. These variables can become complex, signifying a phase difference. This means also the reflection and impedance can become a complex quantity. The $+$ and $-$ subscript signify the complex pressure of the forward and backward traveling waves respectively. These quantities can be derived from the solution of the Helmholtz equations, and is discussed in section 2.1.

$$R(f) = \frac{\hat{p}_-(f)}{\hat{p}_+(f)} \tag{1.32}$$

$$Z(f) = \frac{\hat{p}(f)}{\hat{u}(f)} \tag{1.33}$$

When the same assumptions as for the construction of the wave equations apply to the problem, the characteristic impedance (eq. 1.34) can be used. Which is an inherent property of the medium.

$$Z_0 = \rho_0 \cdot c_0 \tag{1.34}$$

Using the definition of the impedance (eq. 1.33) and the characteristic impedance (eq. 1.34) the acoustic pressure δp can be easily expressed as a function of the acoustic velocity δu and the characteristic impedance Z_0 (eq. 1.35).

$$\delta p = \rho_0 c_0 \delta u = Z_0 \delta u \tag{1.35}$$

The impedance is a complex quantity, and often expressed in a resistive ($\mathcal{R}$) and reactive ($\mathcal{X}$) part (see eq. 1.36).

$$Z = \mathcal{R} + i\mathcal{X} \qquad \mathcal{R} = \mathcal{R}(\vec{x}, \omega) \quad \textit{and} \quad \mathcal{X} = \mathcal{X}(\vec{x}, \omega) \tag{1.36}$$

Chapter 2

Reflection measurement theory

Different methods exist to quantify reflection or impedance of a sample or interface. Among the most used are the Two-Microphone Method (TMM), the Multi-Microphone Method (MMM), and the Standing-Wave Ratio Method (SWR Method). The MMM is the method of choice because the other two methods have certain disadvantages that make them less suitable for this application. The TMM is simple, but very sensitive to sensor positioning, especially if the microphone separation is near one half wavelength [1][2]. The SWR method is on the other hand very accurate and reliable, but very time-consuming because a probe with a pressure sensor needs to moved through the tube [1][2]. As *ANSYS CFX* does not support moving monitor points (to measure pressure), and approximating the Standing-Wave Ratio as a function of position might be a cumbersome task, the SWR-Method is at a disadvantage. Also, the MMM method is the only method that takes into account mean flow explicitly.

2.1 Two Microphone Method

The Two-Microphone Method (TMM) uses, as the name implies, two microphones positioned a certain distance from each other within the domain (see fig. 2.1).

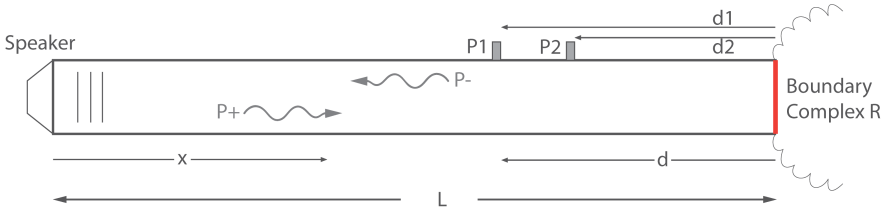


Figure 2.1: Two Microphone Method in impedance tube

The method is based on the Helmholtz equation (2.1) and its solution (2.2). The Helmholtz equation works in the spatial frequency domain.

$$\nabla^2 \hat{p} + k^2 \hat{p} = 0 \qquad where \quad k = \frac{\omega}{c_0} \tag{2.1}$$

$$\hat{p}(x, f) = A(f) \cdot e^{-ikx} + B(f) \cdot e^{ikx} \tag{2.2}$$

By evaluating the solution of the Helmholtz equation (eq. 2.2) for the position of sensor 1 and 2 (eq. 2.3 and 2.4), a fully defined system of equations (eq. 2.5) can be created with which the forward and backward travelling pressure phasors $\hat{p}_+$ and $\hat{p}_-$ at the boundary can be calculated using either matrix inversion or Gaussian elimination. With this information, the calculation of the complex reflection coefficient is straightforward by using eq. 1.32. The

introduction of a new coordinate d simplifies matters a little bit because $\hat{p}_+$ and $\hat{p}_-$ can be evaluated at d=0 by using the fact that $L - d = x$.

$$\hat{p}(x_1, f) = A(f)e^{-ikx_1} + B(f)e^{ikx_1} = A(f)e^{-ikL}e^{ikd_1} + B(f)e^{ikL}e^{-ikd_1} \tag{2.3}$$

$$\hat{p}(x_2, f) = A(f)e^{-ikx_2} + B(f)e^{ikx_2} = A(f)e^{-ikL}e^{ikd_2} + B(f)e^{ikL}e^{-ikd_2} \tag{2.4}$$

$$\begin{bmatrix} e^{ikd_1} & e^{-ikd_1} \\ e^{ikd_2} & e^{-ikd_2} \end{bmatrix} \begin{Bmatrix} \hat{p}_+ \\ \hat{p}_- \end{Bmatrix} = \begin{Bmatrix} \hat{p}_1 \\ \hat{p}_2 \end{Bmatrix} \tag{2.5}$$

Where $\hat{p}_+ = A \cdot e^{-ikL}$ and $\hat{p}_- = B(f) \cdot e^{ikL}$. By normalizing equation 2.3 and 2.4 by $\hat{p}(x_1, f)$, a version of the formula for the reflection coefficient which is not dependent on the absolute position of the pressure sensors, but just the distance s between the sensors can be derived(eq. 2.6). $H_{21} = \frac{\hat{p}_1}{\hat{p}_2}$.

$$R(f) = \frac{H_{21} - e^{-iks}}{e^{iks} - H_{21}} \tag{2.6}$$

2.2 Multi-Microphone Method

In the Multi-Microphone Method (MMM) three or more microphones are used, rendering the system of equations over-determined. The system is solved using a Least-Squares method.

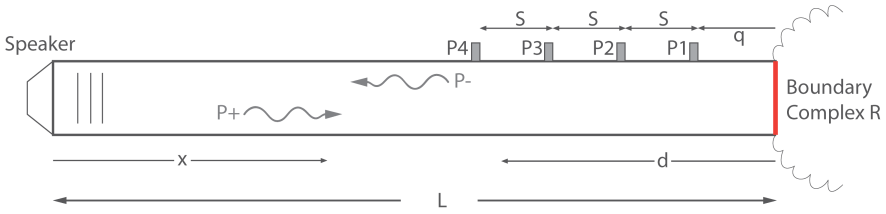


Figure 2.2: Multi Microphone Method in impedance tube

Instead of just the wave numbers, plane wave propagation constants Γ (see eq. 2.7) are used which correct for the effects of mean flow and attenuation effects due to viscosity and turbulence. When these effects are not present, Γ just evaluates to the wave number k.

$$\Gamma_\pm = \mp \frac{k - i\delta}{1 \pm \mathcal{M}} \tag{2.7}$$

Where k is the wave number, $\mathcal{M}$ is the Mach number, and δ is an attenuation constant. The new system of equations $\mathbf{A}\vec{x} = \vec{b}$ (eq. 2.8) resembles equations 2.5, but is now over determined.

$$\begin{bmatrix} e^{-ikx_1} & e^{ikx_1} \\ e^{-ikx_2} & e^{ikx_2} \\ \dots & \dots \\ e^{-ikx_n} & e^{ikx_n} \end{bmatrix} \begin{Bmatrix} \hat{p}_+ \\ \hat{p}_- \end{Bmatrix} = \begin{Bmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \dots \\ \hat{p}_n \end{Bmatrix} \tag{2.8}$$

The Least-Squares solution can be obtained using a Moor-Penrose inverse $\mathbf{A}^+$ (eq. 2.9). Where $\mathbf{A}^H$ is the Hermitian matrix of $\mathbf{A}$.

$$\vec{x} = \mathbf{A}^+\vec{b} = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\vec{b} \tag{2.9}$$

The sensitivity to measurement errors is expressed in the Singularity Factor SF. The lower the SF value, the lower the estimation error. Seung-Ho et al [1] showed that the SF is minimal at a k_{ratio} (see eq. 2.10) of 0.5. Where k_{cr} is a critical wave number corresponding to the maximum

wave number that can be uniquely determined by the set-up. Also the spacing between different sensors seems to have an influence on the SF value, where a uniform equidistant spacing between the sensors resulted in the lowest SF. There are no limitations on the spacing regime in the test cases, therefore a uniform equidistant spacing is used. Because of the use of uniform equidistant spacing, a simpler formula to determine the optimal distance between sensors can be used (eq. 2.11). The wave number k of the excited wave is known at forehand, from this k_{cr} can be calculated using eq. 2.10. Subsequently Δx_{mic} can be determined using 2.11.

$$k_{ratio} = \frac{k}{k_{cr}} \tag{2.10}$$

$$k_{cr} \cdot \Delta x_{mic} = \pi \cdot (1 - \mathcal{M}^2) \tag{2.11}$$

When no mean flow is present, four sensors per wavelength are optimal.

Attenuation effects Two effects contribute to the attenuation constant δ (eq. 2.12): the viscothermal effects (δ_ν) and the turbulent effects (δ_t). The viscothermal effects are described by equation 2.13, and the turbulent effects are described by equation 2.14.

$$\delta = \delta_\nu + \delta_t \tag{2.12}$$

$$\delta_\nu = \frac{\omega}{2\sqrt{2}a_0 c_0}\left[\sqrt{\frac{\mu}{\rho_0 \omega}} + (\gamma - 1)\sqrt{\frac{\kappa}{\rho_0 \omega c_p}}\right] \tag{2.13}$$

$$\delta_t = \frac{\Psi \mathcal{M}}{2a_0}\left(1 + \frac{\Psi' Re}{2\Psi}\right) \tag{2.14}$$

Where μ is the shear viscosity constant, κ is the heat conduction coefficient, and Re is the Reynolds number. The coefficient of friction for turbulent flow Ψ is calculated from the Prandtl universal resistance law (eq. 2.15). See literature from Sueng-Ho [1] for more detailed information on the subject.

$$\frac{1}{\sqrt{\Psi}} = 2 \cdot log_{10}(Re\sqrt{\Psi}) - 0.8 \tag{2.15}$$

2.3 Standing Wave Ratio Method

In practice the standing wave field is measured by moving a microphone along the tubes length (see fig. 2.3), which retrieves the pressure maximum at each point. From this the Standing Wave Ratio (SWR) (eq. 2.16) can be determined, which is in words the absolute value of the pressure maximum divided by the absolute value of the pressure minimum within the domain. The amplitude of the reflection coefficient can then be calculated using equation eq. 2.17.

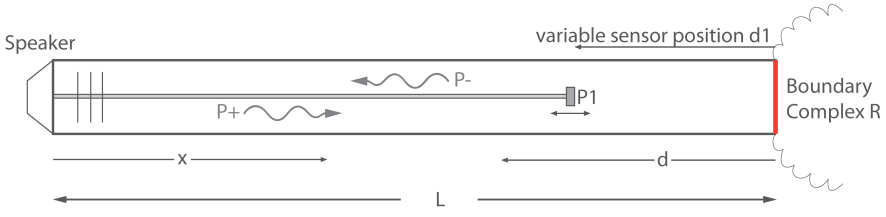


Figure 2.3: SWR Method in impedance tube

$$SWR \equiv \frac{\mid \hat{p}_{max} \mid}{\mid \hat{p}_{min} \mid} \tag{2.16}$$

$$|R| = \frac{SWR - 1}{SWR + 1} \tag{2.17}$$

However, with this method one loses phase information of the standing wave field. By measuring the distance d from the boundary to the first extremum (be it minimum or maximum), the complex reflection coefficient including phase information can be calculated (eq. 2.18)

$$R = \frac{SWR - 1}{SWR + 1} e^{i2(kd)_{max}} \qquad R = -\frac{SWR - 1}{SWR + 1} e^{i2(kd)_{min}} \tag{2.18}$$

Chapter 3

Non-reflective boundary condition theory

This chapter discusses the assumptions made, and the derivation of, the Navier-Stokes Characteristic Boundary Conditions (NSCBC). In the last section a formalism to construct boundary conditions based on the NSCBC method is discussed.

3.1 Navier-Stokes Characteristic Boundary Conditions

It is possible to decouple the quasi-linear first order Euler equations and extract expressions for characteristic waves (Riemann invariants). Using this Riemann invariant transformation to construct a set of relations called the Locally One-Dimensional Inviscid (LODI) relations, the dependent flow variables on the boundary can be determined in terms of characteristic waves.

In short, the methodology's assumptions are:

- Ideal gas assumption as equations of state
- The Viscous and heat terms are neglected (Euler equations)
- Locally one-dimensional (thus one direction of wave propagation), and the transverse terms are neglected.

The NSCBC is derived from the Euler equations, however, this does not limit the calculation of the interior domain to the Euler equations. The interior of the domain can be solved using the full Navier-Stokes equations by adding boundary conditions that account for the viscous dissipation and thermal diffusion. To obtain the missing dependent variables on the boundary one can extrapolate from the inner domain, or use a conservation equation on the boundary. The NSCBC method uses the latter [6].

3.1.1 Characteristic Analysis

The characteristic analysis used in the construction of the hyperbolic boundary condition formalism by Thompson [6], uses the following general hyperbolic equation as a basis:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^1}{\partial x_1} + \frac{\partial \mathbf{F}^2}{\partial x_2} + \frac{\partial \mathbf{F}^3}{\partial x_3} + \mathbf{D} = 0 \tag{3.1}$$

Where $\mathbf{U}$ are the conserved quantities, $\mathbf{F}$ the fluxes, and $\mathbf{D}$ the inhomogeneus source terms. From this equation it can be observed that it has the same form as the Euler equations in

conservative PDE form (1.11). The hyperbolic model equation (3.1) is in conservative form, however, boundary conditions further on are greatly simplified by using the primitive form (primitive quantity vector denoted by a p superscript: $\mathbf{U}^p$). For example: it is easier to prescribe the velocity u at the boundary than the momentum ρu. Thompson transforms the conserved form to a primitive form before characteristic analysis:

$$\frac{\partial \mathbf{U}}{\partial t} = \mathbf{P}\frac{\partial \mathbf{U}^p}{\partial t} \quad where \quad P_{ij} \equiv \frac{\partial U_i}{\partial U_j^p} \tag{3.2}$$

Which can be obtained by using the chain rule, after all the conserved quantities $\mathbf{U}$ can be written as a function of primitive quantities $\mathbf{U}^p$: $\mathbf{U} = f(\mathbf{U}^p)$. For example, momentum ρu is a function of ρ and u. For the flux terms one may do the same:

$$\frac{\partial \mathbf{F}^k}{\partial x_k} = \mathbf{Q}^k\frac{\partial \mathbf{U}^p}{\partial x_k} \quad where \quad k = 1, 2, 3 \tag{3.3}$$

The matrix $\mathbf{Q}^k$ has elements:

$$\mathbf{Q}^k_{ij} \equiv \frac{\partial F_i^k}{\partial U_j^p} \tag{3.4}$$

Substitution into eq. (3.1) and left multiply by P^{-1} results in the primitive form of the model equation:

$$\frac{\partial \mathbf{U}^p}{\partial t} + \mathbf{A}^1\frac{\partial \mathbf{U}^p}{\partial x_1} + \mathbf{A}^2\frac{\partial \mathbf{U}^p}{\partial x_2} + \mathbf{A}^3\frac{\partial \mathbf{U}^p}{\partial x_3} + \mathbf{D}^p = 0 \tag{3.5}$$

Where $\mathbf{A}^k = \mathbf{P}^{-1}\mathbf{Q}^k$ and $\mathbf{D}^p = \mathbf{P}^{-1}\mathbf{D}$. The system of equations is subsequently lumped into two parts (3.6). The $\mathbf{C}^p$ part is composed of the transverse and source terms which do not substantially add to the characteristic analysis, but are just carried along passively.

$$\frac{\partial \mathbf{U}^p}{\partial t} + \mathbf{A}^1\frac{\partial \mathbf{U}^p}{\partial x_1} + \mathbf{C} = 0 \qquad C = \mathbf{A}^2\frac{\partial \mathbf{U}^p}{\partial x_2} + \mathbf{A}^3\frac{\partial \mathbf{U}^p}{\partial x_3} + \mathbf{D} \tag{3.6}$$

The system of partial differential equations (PDE) (3.5) is quasi-linear and first order, such a system is hyperbolic iff the eigenvalues of the matrices $\mathbf{A}^k$ are real. However, these multiple A matrices in multi-dimensional problems are not simultaneously diagonalizable, and therefor cannot be decoupled and consequently do not have an unique direction of propagation. Multi-dimensional problems cannot be put into characteristic form, however, one-dimensional problems with only one $\mathbf{A}$ matrix can be diagonalized and thus put into characteristic form. For multi-dimensional problems locally one-dimensional wave propagation is assumed and the transverse terms C drops out:

$$\frac{\partial \mathbf{U}^p}{\partial t} + \mathbf{A}^1\frac{\partial \mathbf{U}^p}{\partial x_1} = 0 \tag{3.7}$$

The coefficient matrix $\mathbf{A}^1$ in the system of equations (3.7) can be used for eigenvalue extraction. The left $\mathbf{l}$ and right $\mathbf{r}$ eigenvectors have the following properties:

$$\mathbf{l}_i^T\mathbf{A}^1 = \lambda_i\mathbf{l}_i^T \qquad \mathbf{A}^1\mathbf{r}_i = \lambda_i\mathbf{r}_i \qquad where \quad i = 1, \ldots, m \tag{3.8}$$

The eigenvalues can be determined via the well-known formula $det(\mathbf{A}^1 - \lambda\mathbf{I}) = 0$. By using a similarity transformation the matrix $\mathbf{A}^1$ can be transformed to a diagonal matrix with eigenvalues:

$$\mathbf{S}^{-1}\mathbf{A}^1\mathbf{S} = \mathbf{\Lambda} \tag{3.9}$$

The matrix $\mathbf{S}$ and $\mathbf{S}^{-1}$ can be constructed by adding the right eigenvectors as columns or the left eigenvectors as rows respectively. By substituting eq. (3.9) into eq. (3.6), and subsequently left multiply with $\mathbf{S}^{-1}$ one arrives at:

$$\mathbf{S}^{-1}\frac{\partial \mathbf{U}}{\partial t} + \mathcal{L} + \mathbf{S}^{-1}\mathbf{C} = 0 \qquad where \quad \mathcal{L} = \mathbf{\Lambda}\mathbf{S}^{-1}\frac{\partial \mathbf{U}}{\partial x_1} \tag{3.10}$$

In component form:

$$\mathbf{l}_i^T \frac{\partial \mathbf{U}}{\partial t} + \mathcal{L}_i + \mathbf{l}_i^T \mathbf{C} = 0 \qquad where \quad \mathcal{L}_i \equiv \lambda_i \mathbf{l}_i^T \frac{\partial \mathbf{U}}{\partial x_1} \tag{3.11}$$

The original hyperbolic equations are now put into a characteristic form, as a function of the amplitudes of the characteristic waves $\mathcal{L}_i$.

3.1.2 LODI-relations

By using the characteristic analysis procedure in section 3.1.1 on the Euler equations (1.11) the so-called Locally One-Dimensional Inviscid (LODI) relations can be derived. First the column vectors constructed:

$$U \equiv \begin{Bmatrix} \rho \\ e \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \end{Bmatrix} \quad U_p \equiv \begin{Bmatrix} \rho \\ p \\ u_1 \\ u_2 \\ u_3 \end{Bmatrix} \quad F^k \equiv \begin{Bmatrix} \rho u_k \\ (e+p)u_k \\ \rho u_k u_1 + \delta_{k1} p \\ \rho u_k u_2 + \delta_{k2} p \\ \rho u_k u_3 + \delta_{k3} p \end{Bmatrix} \quad D \equiv \begin{Bmatrix} 0 \\ -\rho \sum_{k=1}^{3} u_k g_k \\ -\rho g_1 \\ -\rho g_2 \\ -\rho g_3 \end{Bmatrix} \tag{3.12}$$

Where δ is the Kronecker delta function. Next the Jacobian matrix $\mathbf{P}$ (3.2) and $\mathbf{Q}$ (3.4) are constructed:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2}\sum_{i=1}^{3} u_i^2 & \frac{1}{(\gamma-1)} & \rho u_1 & \rho u_2 & \rho u_3 \\ u_1 & 0 & \rho & 0 & 0 \\ u_2 & 0 & 0 & \rho & 0 \\ u_3 & 0 & 0 & 0 & \rho \end{bmatrix} \tag{3.13}$$

Using the two equations of state (eq. 3.14), the matrix $\mathbf{Q}^k$ can be determined from (3.4):

$$\begin{aligned} e &\equiv \tfrac{1}{2}\rho \textstyle\sum_{k=1}^{3} u_k^2 + \epsilon \\ p &\equiv (\gamma - 1)\epsilon \end{aligned} \tag{3.14}$$

$$\mathbf{Q}^k = \begin{bmatrix} u_k & 0 & \delta_{k1}\rho & \delta_{k2}\rho & \delta_{k3}\rho \\ \frac{1}{2}\sum_{i=1}^{3} u_i^2 u_k & (\frac{1}{\gamma-1}+1)u_k & f_{k1} & f_{k2} & f_{k3} \\ u_k u_1 & \delta_{k1} & (1+\delta_{k1})\rho u_k & \delta_{k2}\rho u_1 & \delta_{k3}\rho u_1 \\ u_k u_2 & \delta_{k2} & \delta_{k1}\rho u_2 & (1+\delta_{k2})\rho u_k & \delta_{k3}\rho u_2 \\ u_k u_3 & \delta_{k3} & \delta_{k1}\rho u_3 & \delta_{k2}\rho u_3 & (1+\delta_{k3})\rho u_k \end{bmatrix} \tag{3.15}$$

Where f_{ki} is defined as:

$$f_{ki} = \frac{1}{2}\rho((2+\delta_{ki})u_i u_k + \delta_{ki}\sum_{j=1}^{3}(1-\delta_{ij})u_j^2) \tag{3.16}$$

In which $\mathbf{A}^k = \mathbf{P}^{-1}\mathbf{Q}^k$ can be used to find $\mathbf{A}^1$. Matrix $\mathbf{Q}^1$ becomes:

$$\mathbf{Q}^1 = \begin{bmatrix} u_1 & 0 & \rho & 0 & 0 \\ \frac{1}{2}\sum_{i=1}^{3} u_i^2 u_1 & (\frac{1}{\gamma-1}+1)u_1 & \frac{1}{2}\rho(3u_1^2+u_2^2+u_3^2) & \rho u_2 u_1 & \rho u_3 u_1 \\ u_1^2 & 1 & 2\rho u_1 & 0 & 0 \\ u_1 u_2 & 0 & \rho u_2 & \rho u_1 & 0 \\ u_1 u_3 & 0 & \rho u_3 & 0 & \rho u_1 \end{bmatrix} \tag{3.17}$$

The inverse of the Jacobian matrix $\mathbf{P}$ is found to be:

$$\mathbf{P}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2}\sum_{i=1}^{3} u_i^2(\gamma-1) & (\gamma-1) & -(\gamma-1)u_1 & -(\gamma-1)u_2 & -(\gamma-1)u_3 \\ -u_1\rho^{-1} & 0 & \rho^{-1} & 0 & 0 \\ -u_2\rho^{-1} & 0 & 0 & \rho^{-1} & 0 \\ -u_3\rho^{-1} & 0 & 0 & 0 & \rho^{-1} \end{bmatrix} \tag{3.18}$$

Which results in $\mathbf{A}^1$:

$$\begin{bmatrix} u_1 & 0 & \rho & 0 & 0 \\ 0 & u_1 & \gamma p & 0 & 0 \\ 0 & \rho^{-1} & u_1 & 0 & 0 \\ 0 & 0 & 0 & u_1 & 0 \\ 0 & 0 & 0 & 0 & u_1 \end{bmatrix} \tag{3.19}$$

From which the eigenvalues can be determined using cofactor expansion (where Z is defined for convenience):

$$det(\mathbf{Z}) = det(\mathbf{A}^1 - \lambda I) = 0 = \begin{bmatrix} u_1-\lambda & 0 & \rho & 0 & 0 \\ 0 & u_1-\lambda & \gamma p & 0 & 0 \\ 0 & \rho^{-1} & u_1-\lambda & 0 & 0 \\ 0 & 0 & 0 & u_1-\lambda & 0 \\ 0 & 0 & 0 & 0 & u_1-\lambda \end{bmatrix} \tag{3.20}$$

$$\begin{aligned} det(Z) &= (u_1-\lambda)\cdot det(Z_{2:5,2:5}) = 0 \\ (u_1-\lambda)\cdot[(u_1-\lambda)\cdot det(Z_{3:5,3:5}) - \rho^{-1}\cdot det(Z_{2,3:5} : Z_{4:5,3:5})] &= 0 \\ (u_1-\lambda)\cdot[(u_1-\lambda)^4 - \rho^{-1}\gamma p(u_1-\lambda)^2] &= 0 \\ (u_1-\lambda)\cdot[(u_1-\lambda)^4 - c^2(u_1-\lambda)^2] &= 0 \\ (u_1-\lambda)^3\cdot[(u_1-\lambda)^2 - c^2] &= 0 \\ (u_1-\lambda)^3 \rightarrow \lambda_{2,3,4} &= u_1 \\ (u_1-\lambda)^2 - c^2 = [(u_1-\lambda)+c][(u_1-\lambda)-c] \rightarrow \lambda_{1,5} &= (u_1 \mp c) \end{aligned} \tag{3.21}$$

The eigenvalues λ are sorted in increasing order, and represent the characteristic velocity at which a particular wave mode propagates. The speed of sound is $c^2 = \frac{\gamma p}{\rho}$ (ideal gas, adiabatic). The (left) eigenvectors are determined using (3.8), putting them into the form $\mathbf{l}_i^T(\mathbf{A}^1 - \lambda\mathbf{I}) = \vec{0}$, and solve for $\mathbf{l}$ corresponding to each eigenvalue.

The left eigenvectors are:

$$\begin{aligned} \mathbf{l}_1^T &= (0, 1, -\rho c, 0, 0) \\ \mathbf{l}_2^T &= (c^2, -1, 0, 0, 0) \\ \mathbf{l}_3^T &= (0, 0, 0, 1, 0) \\ \mathbf{l}_4^T &= (0, 0, 0, 0, 1) \\ \mathbf{l}_5^T &= (0, 1, \rho c, 0, 0) \end{aligned} \tag{3.22}$$

From eq. (3.11) the characteristic waves can be determined:

$$\begin{aligned} \mathcal{L}_1 = \lambda_1(0,1,-\rho c,0,0)\frac{\partial \mathbf{U}}{\partial x_1} &= \lambda_1(\frac{\partial p}{\partial x_1} - \rho c\frac{\partial u_1}{\partial x_1}) \\ \mathcal{L}_2 = \lambda_2(c^2,-1,0,0,0)\frac{\partial \mathbf{U}}{\partial x_1} &= \lambda_2(c^2\frac{\partial \rho}{\partial x_1} - \frac{\partial p}{\partial x_1}) \\ \mathcal{L}_3 = \lambda_3(0,0,0,1,0)\frac{\partial \mathbf{U}}{\partial x_1} &= \lambda_3\frac{\partial u_2}{\partial x_1} \\ \mathcal{L}_4 = \lambda_4(0,0,0,0,1)\frac{\partial \mathbf{U}}{\partial x_1} &= \lambda_4\frac{\partial u_3}{\partial x_1} \\ \mathcal{L}_5 = \lambda_5(0,1,\rho c,0,0)\frac{\partial \mathbf{U}}{\partial x_1} &= \lambda_5(\frac{\partial p}{\partial x_1} + \rho c\frac{\partial u_1}{\partial x_1}) \end{aligned} \tag{3.23}$$

Rewriting these equation in terms of the spatial derivative of the primitive flow quantities along propagation x_1:

$$\begin{array}{rcl} \frac{\partial \rho}{\partial x_1} & = & \frac{1}{c^2}[\frac{\mathcal{L}_2}{u_1} + \frac{1}{2}(\frac{\mathcal{L}_5}{u_1+c} + \frac{\mathcal{L}_1}{u_1-c})] \\ \frac{\partial p}{\partial x_1} & = & \frac{1}{2}(\frac{\mathcal{L}_5}{u_1+c} + \frac{\mathcal{L}_1}{u_1-c}) \\ \frac{\partial u_1}{\partial x_1} & = & \frac{1}{2\rho c}(\frac{\mathcal{L}_5}{u_1+c} - \frac{\mathcal{L}_1}{u_1-c}) \\ \frac{\partial u_2}{\partial x_1} & = & \frac{\mathcal{L}_3}{u_1} \\ \frac{\partial u_3}{\partial x_1} & = & \frac{\mathcal{L}_4}{u_1} \end{array} \tag{3.24}$$

These two sets of equations (3.23)(3.24) form the LODI-relations. Albeit, different forms of (3.24) may be constructed, depending on the application. Using eq. (3.7) the time derivatives of the primitive flow quantities can easily be obtained using the spatial derivatives (3.24) and the matrix $\mathbf{A}^1$ (3.19), which results in the set of LODI relations in temporal derivatives (3.26).

$$\frac{\partial \mathbf{U}^p}{\partial t} = -\mathbf{A}^1 \frac{\partial \mathbf{U}^p}{\partial x_1} \tag{3.25}$$

$$\begin{array}{rcl} \frac{\partial \rho}{\partial t} & = & -\frac{1}{c^2}[\mathcal{L}_2 + \frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1)] \\ \frac{\partial p}{\partial t} & = & -\frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1) \\ \frac{\partial u_1}{\partial t} & = & -\frac{1}{2\rho c}(\mathcal{L}_5 - \mathcal{L}_1) \\ \frac{\partial u_2}{\partial t} & = & -\mathcal{L}_3 \\ \frac{\partial u_3}{\partial t} & = & -\mathcal{L}_4 \end{array} \tag{3.26}$$

The characteristic velocities λ_i and the corresponding characteristic waves $\mathcal{L}_i$ are for numbers 1 and 5 the sound waves moving in the negative and positive directions respectively, number 2 the entropy advection, and numbers 3 and 4 are the transverse waves for u_3 and u_4 respectively.

3.1.3 Boundary conditions/treatments

In the formalism for the treatment of boundary conditions by Thompson [6], a clear distinction is made between a "boundary condition" and a "boundary treatment".

- Boundary Condition
  1. Single mathematical expression
  2. External to the calculation of $\mathbf{U}$ within interior of domain Ω.
  3. Contributes to, but does not itself define $\frac{\partial \vec{U}}{\partial t}$
- Boundary Treatment
  1. Complete algorithm for defining the values of $\frac{\partial \mathbf{U}}{\partial t}$ along the boundary.
  2. Incorporates information of $\mathbf{U}$ from both interior Ω and the boundary conditions at $\partial\Omega$.

Where $\mathbf{U}$ are the primitive flow quantities and Ω is the interior of the domain. Directly defining $\mathbf{U}$ at the boundary is prohibited in the formalism, the boundary condition only contributes to the determination of $\frac{\partial \mathbf{U}}{\partial t}$ at the boundary. To overcome this, one could set certain primitive flow quantities at the boundary as an initial condition, and force the time derivative of the primitive flow quantity to be zero at the boundary at all times.

To construct boundary conditions the characteristic velocities λ are used to determine whether a characteristic wave travels from the interior Ω outwards or into the volume. When the wave travels outwards trough the boundary, $\mathcal{L}_i$ on the boundary can just be calculated using one-sided derivative approximations and the definitions for $\mathcal{L}_i$ (3.23). When the wave travels

inwards, the corresponding $\mathcal{L}_i$ must be determined using the boundary condition specified on the boundary. From this general procedure all kind of boundary conditions can be derived. Some frequently used boundary conditions are discussed by Poinsot and Thompson [4][6], but only the subsonic nonreflecting outflow boundary condition is treated here.

A Subsonic Nonreflecting outflow A nonreflecting boundary condition is seen within this formalism as a wave traveling back into the domain Ω that does not exhibit any time-varying behavior, i.e. is constant in time. As wave motion is by definition time-varying, this is equivalent to stating that there is no reflected wave. Using equation (3.11) and neglecting transverse terms to derive an expression for the 'reflected' wave $\mathcal{L}_1$:

$$(\frac{\partial p}{\partial t} - \rho c \frac{\partial u_1}{\partial t}) + \mathcal{L}_1 + \rho c g_1 = 0 \tag{3.27}$$

The expression within the parenthesis is the time derivative of the well-known Riemann invariant in acoustics $p = \rho_0 c_0 u$. Rewriting the invariant to $p - \rho_0 c_0 u = 0$, one can easily see that the term within the parenthesis should equal zero. Eq. (3.27) reduces to $\mathcal{L}_1 = -\rho c g_1$, where g_1 are the specific volumetric forces. Often the only volumetric force terms used are those due to gravitation, which are usually more or less homogeneous over the domain (especially with a nearly incompressible fluid). Usually one does not have to account for the volumetric forces, as they are often not present in the direction of propagation x_1. The creation of a nonreflective boundary is equal to stating: $\mathcal{L}_1 = 0$.

Subsonic		
	$x_1 = a_1$	$x_1 = b_1$
With volumetric forces	$\mathcal{L}_5 = \rho c g_1$	$\mathcal{L}_1 = -\rho c g_1$
Without volumetric forces	$\mathcal{L}_5 = 0$	$\mathcal{L}_1 = 0$

Table 3.1: Wave amplitude expressions at the boundaries

3.2 Linear Relaxation Methods

The $\mathcal{L}_1$ amplitude can be set to zero to effectively eliminate reflection, but results often in an ill-posed problem. Then a drift of the mean flow field quantities occurs because far–field information about pressure cannot propagate into the domain. To overcome the problem of ill-posedness one could set the far-field pressure p_∞, but this would create an acoustically hard surface and thus reflection. One way or another, information from the far–field pressure must be able to travel into the domain, while maintaining a more or less non-reflecting boundary. The LRM model is the simplest boundary treatment for $\mathcal{L}_1$ that accommodates both:

$$\mathcal{L}_1 = K(p - p_\infty) \tag{3.28}$$

The relaxation factor K can be defined in multiple ways. An expressions for K is:

$$K = \frac{\sigma(1 - \mathcal{M}^2)c}{L} \tag{3.29}$$

For small values of K the problem might become ill-posed again, while for large values of K the boundary looses its non-reflective character. This behavior also seems frequency dependent as at lower frequencies the boundary treatment causes reflectivity, as will be looked at in section 3.2.1. Which will also deal with optimal values for coupling parameter σ, and other performance issues. Section 3.2.2 will look at the performance of the nonreflective boundary in *ANSYS CFX*. The last section **??** will look at an extension to the LRM model which in theory is fully non-reflective while maintaining a well-posed problem.

3.2.1 Performance of the LRM method

For the LRM method, Poinsot showed using numerical calculations that one-dimensional waves exit the domain (thus normal incidence) with very small reflection in the order of $R = 10^{-6}$ [4]. A vortex in a parallel flow field exits the domain without much reflection, albeit more than the one-dimensional wave, in the order of $R = 10^{-4}$.

Two aspects of the LRM model are looked at by Selle in more detail [5]:

- The effect of the LRM model on acoustic behavior.
- Optimal choices for the proportionality constant K.

The value of K determines the degree to which the far–field pressure p_∞ is enforced. A small value allows for a large pressure difference and reflects only a small fraction, but is not very able to control the mean pressure. A large value of K enforces the difference between p and p_∞ to be small, but reflects a larger fraction. Thus, the value of K has a direct relation to the reflection coefficient R.

$$R = \frac{p_{refl}}{p_{out}} \qquad R \in \mathbb{C} \tag{3.30}$$

The incoming wave $\mathcal{L}_5$ is assumed to be harmonic, and is expressed as:

$$\mathcal{L}_5 = 2\rho c\hat{u}i\omega e^{-i\omega t} \tag{3.31}$$

And the reflected wave is equal to (3.28), the LRM model. Substituting the two above expressions in the second and third LODI relations (3.26) leads to the following system of equations:

$$\frac{\partial u}{\partial t} + \frac{1}{2\rho c}(2\rho c\hat{u}i\omega e^{-i\omega t} - K(p - p_\infty)) = 0 \tag{3.32}$$

$$\frac{\partial p}{\partial t} + \frac{1}{2}(2\rho c\hat{u}i\omega e^{-i\omega t} + K(p - p_\infty)) = 0 \tag{3.33}$$

The second equation depends only on P and can be solved analytically:

$$p(t) = p_\infty + A_0 e^{-\frac{Kt}{2}} - \frac{\rho c\hat{u}i\omega}{\frac{K}{2} - i\omega} e^{-i\omega t} \tag{3.34}$$

Looking at the limit behavior when the transient term vanishes for $t \to \infty$ and $K > 0$, which results in:

$$p(t) = p_\infty - \frac{\rho c\hat{u}i\omega}{\frac{K}{2} - i\omega} e^{-i\omega t} \tag{3.35}$$

Subsituting the relations for $\mathcal{L}_1$ and $\mathcal{L}_5$ into the reflection coefficient:

$$R = \frac{p_{refl}}{p_{out}} = \frac{K(p - p_\infty)}{2\rho c\hat{u}i\omega e^{-i\omega t}} \tag{3.36}$$

The expression for p (3.35) is substituted in (3.36):

$$R = \frac{K(P_\infty - \frac{\rho c\hat{u}i\omega}{\frac{K}{2} - i\omega} e^{-i\omega t} - P_\infty)}{2\rho c\hat{u}i\omega e^{-i\omega t}} = \frac{-K}{K - 2i\omega} = \frac{-1}{1 - \frac{2i\omega}{K}} \tag{3.37}$$

Because R is a complex number it can be expressed in a magnitude $\|R\|$ and a phase angle ϕ.

$$\|R\| = \frac{1}{\sqrt{1 + (\frac{2\omega}{K})^2}} \tag{3.38}$$

$$\phi = -\pi - arctan(\frac{2\omega}{K}) \tag{3.39}$$

To illustrate how the reflection coefficient R evolves through the complex plane as a function of relaxation factor K a plot is made 3.1.

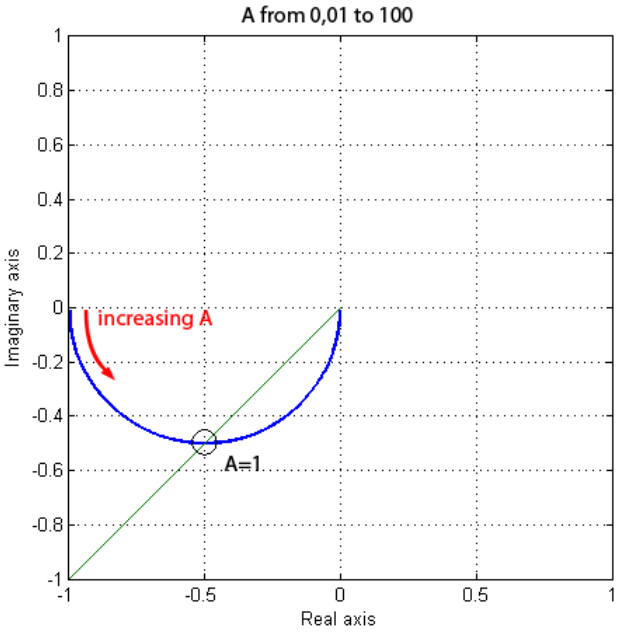


Figure 3.1: Reflection coefficient R in the complex plane for varying K

Where A is defined as:

$$A = \frac{2\omega}{K} \tag{3.40}$$

A increases proportional with ω, and inversely proportional with K.

Limit behavior of the reflection for different values of relaxation factor K are shown in table 3.2.1.

Table 3.2: Limit behavior of reflection coefficient

A	$\|R\|$	ϕ
$A = 0$	1	$-\pi$
$A = 1$	$\frac{1}{\sqrt{2}}$	$-\pi - \frac{\pi}{4}$
$A \to \infty$	0	$-\frac{3\pi}{2}$

For high frequencies (and low K) the reflection factor R tends to zero, and the boundary condition can be called non-reflecting (see fig. 3.2). For low frequencies (and high K), however, the reflection coefficient R tends to 1, and the phase to $-\pi$, basically acting as a pressure release surface. Selle defines a cut-off frequency at $A = 1$, below which most of the acoustic energy is reflected back into the domain [5].

$$f_c = \frac{\omega_c}{2\pi} = \frac{K}{4\pi} \tag{3.41}$$

In practice different cut-off frequencies may be defined depending on problem characteristics.

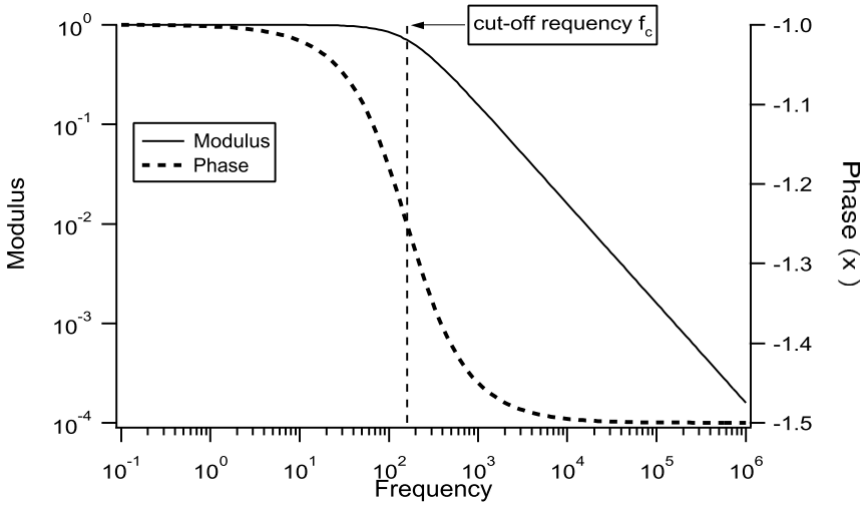


Figure 3.2: Modulus and phase of reflection coefficient as a function of frequency

Scaling strategy for K A scaling strategy for the relaxation factor K is proposed [5]. Good values of K should allow all duct modes to leave the domain. Using the cut-off frequency, which defines the reflected from the non-reflected part, lower than the lowest eigenvalue. The lowest eigen-mode is the quarter-wave mode, which translates to the frequency domain via:

$$f_0 = (1 - \mathcal{M}^2)\frac{c}{4L} \tag{3.42}$$

Using eq. (3.41) and eq. (3.42), the highest value of K which still significantly damps the $\frac{1}{4}$ wave mode can be determined:

$$K^{max} = \pi(1 - \mathcal{M}^2)\frac{c}{L} \tag{3.43}$$

When comparing eq. 3.43 to eq. 3.29 it can be seen that σ is bounded above by π. The lower bound is not fixed by acoustics, but by other parameters such as the Reynolds number and geometry. A value for σ lower than 0.1 often results in increased convergence times or no convergence at all due to too much pressure drift. This results in an adequate range of values for σ as shown in figure 3.3.

Optimal values for σ from literature range from 0.27 based on theory to 0.58 based on numerical simulations [5]. One only needs to verify if the frequency of interest (e.g. excitation frequency) is higher than the quarter-wave length frequency (see eq. 3.44). This is because eq. (3.29) is based on this frequency and the cut-off frequency. Important values for σ are summed up in table 3.2.1.

$$f_{quarter} = \frac{c_0}{4 \cdot L} \tag{3.44}$$

Using numerical analysis two values for sigma were tested. One value is way to high, with $\sigma = 10\pi$. At this value the first eigenmodes are unable to leave the domain, and remain asymptotically as a standing wave. The second value is equal to the maximum value $\sigma = \pi$. The

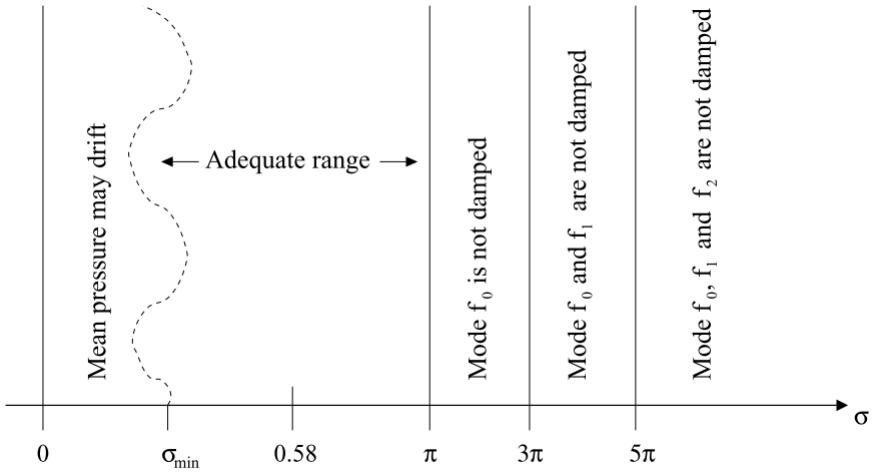


Figure 3.3: Optimal values for sigma

$\sigma^{min} \approx 0.1$	$\sigma^{max} = \pi$	$\sigma^{opt1} = 0.27$	$\sigma^{opt2} = 0.58$

Table 3.3: Important sigma values

standing waves within the tubes die out rapidly, although the eigenfrequencies are significantly shifted towards a lower frequency.

3.2.2 Performance of the non-reflective BC in Ansys CFX

A study to compare the perfomance predicted by theory with the (beta) implementation by *ANSYS CFX* is investigated in [3]. Deviations are stumbled upon, and possible explanations are sought. The non-reflecting boundary condition in *ANSYS CFX* is also based on the LODI-relations and the LRM model for the reflected wave. Using a long tube with acoustically hard cylinder walls, a prescribed harmonically varying velocity inlet, and a nonreflecting outlet, the performance of the non-reflecting boundary condition is evaluated.

The numerically determined values for the reflection coefficient agrees with theory for the higher K values. For the lower K values, however, the values do not match. Two possible reasons are proposed. The first is loss in accuracy due to the numerical scheme and discretization. The second is that a build-in mechanism in Ansys is preventing drift of the mean pressure which results with low values of K, and thus preventing convergence problems.

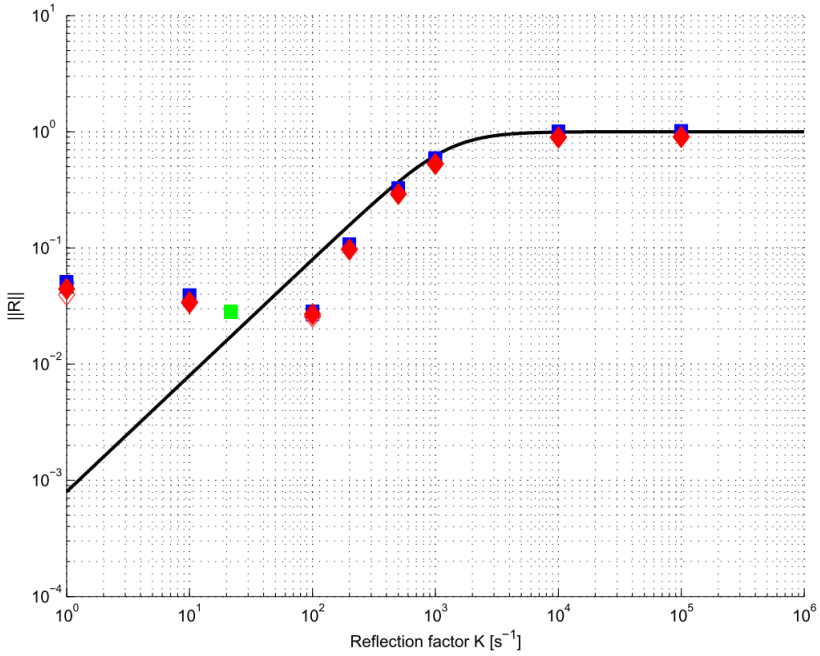


Figure 3.4: Numerically determined R (dots) vs. analytical model (line)

Chapter 4

Numerical modeling

In this chapter workflow in using the developed boundary conditions and problems encountered during programming in *FORTRAN* for *ANSYS CFX* are discussed.

4.1 General workflow

General workflow for setting up a simulation to be run with the developed boundary conditions:

1. Create a mesh with a flat boundary perpendicular to the dominant direction of propagation.
2. Optional (for PWM and TD-IBC): Create a sample plane within the mesh in a mesh creation package (see next section).
3. Call the boundary to be used as a NRBC or TD-IBC with the pre-fix *LRM**.
4. Create a CCL file with the necessary options (see example CCL files in appendix), and add to *ANSYS CFX PRE* file.
5. Compile the **.f* files using the built-in *Command Editor*, and write: *!system("cfx5mkext double NAMA.f") or die;*. Then process (assuming files are in running directory).
6. Add the junction box and user cell routines in *ANSYS CFX PRE*.
7. Call the pset routine from user cell in the pressure specification field of the boundary with the correct arguments and pressure as return dimension.
8. Specify the junction box routines in the *Solver Control* dialogue box.
9. Run the simulation.

4.2 Sample plane creation and use

The creation of a sample plane from which the area averaged pressure and velocity are obtained by the solver is not as straightforward as one might think. Monitor points cannot be used to retrieve the aforementioned quantities as these are apparently retrieved directly from the solution that is written to the disk, which can be seen as a sort-of real-time post-processing. The utility routine *user_ get_gvar* can retrieve and calculate area averaged quantities like pressure and velocity, but need a user named location within the mesh. These locations include boundary surfaces. But it is not possible to define a surface within the domain using *ANSYS CFX PRE* only, as these are only recognized as unnamed internal element surfaces. To create a named surface at the location where the sample plane needs to be, mesh modeling software needs to be used. Follow the step-by-step process listed below to implement the sample plane in question. The process is written for mesh software *ICEM*, and assumes the geometry is already fully designed and opened. Also, the naming is important and should be copied exactly as they are hard-coded in the Fortran subroutines.

1. Create a surface within the geometry, a certain distance from and parallel to the boundary in question.
2. Create a part from the surface and call it *SAMPLEP*.
3. Create two blocks which together enclose the geometry, and of which the separating face is aligned and associated to the *SAMPLEP* surface.
4. Create the mesh and save.
5. Import the mesh in *ANSYS CFX PRE*, and check if the *SAMPLEP_ 1* and *SAMPLEP_ 2* principle 2D regions are shown under the mesh item.
6. Create a 1:1 conservative general connection interface between *SAMPLEP_ 1* and *SAMPLEP_ 2*, and call it sampleplane.
7. Safe, close, and re-open the *ANSYS CFX-PRE* file (or else the automatically created boundaries do not show up).
8. Now two boundaries should appear, call them *sampleplane Side 1* and *sampleplane Side 2*.

The *sampleplane Side 1* boundary is used by the Fortran subroutines to retrieve the area averaged pressure and velocity, and is hard-coded.

4.3 Problems encountered in programming for ANSYS CFX

Several problems and bugs were stumbled upon during modeling and programming for ANSYS CFX. An overview of them is given in table 4.3, and is discussed in detail in the subsections.

Table 4.1: Problems encountered in programming for ANSYS CFX

No.	Type	Applies to	Problem	Status
1	CFX	*ucf_ template.F* template	Gives errors	S
2	CFX	Message to master process	Stops solver inside slave partition	N
3	CFX	Single precision numbers	Accumulation of error	W
4	CFX	Pressure gradient	Bug in *user_ getvar* routine	W
5	CFX	Velocity gradient	Bug in *user_ getvar* routine	W
6	CFX	Utility routines	*CACTION* argument can cause crash	S
7	CFX	User parameters	*user_ peek** fails for big lists	W
8	CFX	Sound speed	Wrong sound speed in isothermal simulations	W
9	Doc	Array shape	*user_ getvar* returns wrong array shape	S
10	Fort	iFort compiler	iFort compiler for Windows cause problems	W
11	Fort	Other compilers	Give problems, flags are difficult	W
12	Linx	X-Connection	X-Connection in Windows is slow/buggy	W
13	Linx	Closing SSH connection	Closing connection causes solver termination	S

In the *Type* column CFX means ANSYS CFX specific, Doc means ANSYS documentation, Fort means Fortran specific, and Linx means Linux specific. In the *Status* column S means Solved, W means no solution is found but a workaround method can be used to overcome the problem, and N means No solution.

ucf_ template.F gives errors

The template file *ucf_ template.F* for user defined CEL functions causes a crash when no utility function is called within the subroutine that uses the argument *CRESLT*, and the *CRESLT* argument is set to *GOOD* at the end of the subroutine. The reason is of this appearant bug is not known, but is easily solved by erasing (or make a comment of) the line of code in question. However, remember to un-comment the line when utility routines are used in the code, so the solver knows the subroutine did not encounter any errors. This also applies to a subroutine called *TStat_ Control.F* supplied in an ANSYS CFX tutorial. The tutorial can be found under help in *ANSYS CFX-PRE* by the name *Chapter 19: Air Conditioning Simulation.*

Message to master process

This bug only applies to partitioned solver runs. Printing a message to the solver output window with the command *MESAGE('WRITE', <text >)* in a DO-LOOP when the subroutine is called by a slave process (often the case in user defined CEL functions) will cause the solver to stop outputting all messages to the solver output window. The source of this bug is unknown.

Single precision numbers accumulate error

If the pressure or velocity at a boundary is calculated via a user defined CEL function, and the mathematical expressions used to calculate these quantities include numbers that vary significantly in order of magnitude, the output may contain an error of such magnitude that the solution becomes inaccurate and eventually unstable (error adds up). This is caused by the

fact that single precision numbers loose significant digits in mathematical operations involving numbers which vary greatly in order of magnitude.

The solution (at least for the code used in this master thesis) is to use double precision (8 byte) numbers instead of single precision (4 byte) numbers. Double precision numbers in *Fortran* have a significand of 52 bit while a single precision number has a significand of 23 bit.

Error in retrieving pressure gradient

A utility routine called *user_ getvar* can retrieve the gradient of different quantities at a certain location. According to the documentation the *variable description* argument *<locale >.Pressure.Gradient* can be used to retrieve the pressure gradient at the given locale, but results in a crash of the solver when used. According to the error description the function *MAKDIR* called within the *user_ getvar* utility function cannot create a directory because it has an illegal character in it. The illegal character is a space. No space is present in the name of the *variable description* argument. This cannot be solved as it is a bug in the *ANSYS CFX* software.

However, when one uses the *variable description* argument *Pressure.Gradient* instead of the aforementioned, the correct gradients are retrieved without problems. Another method of retrieving gradients is by using *additional variables*, these can be created in *ANSYS CFX PRE*. First create an *additional variable*, call it *Px* for example. Go to Body domain, and then to tab *Fluid Models*. Go to additional variable and set option to *Algebraic Equation*, then fill in the *Add. Var. Value* box: *p.Gradient_ x*. Now the pressure gradient variable *Px* can be accessed via the *user_ getvar* routine via the *variable description* argument *Px*. The pressure gradient value is calculated using the latest solution field.

Error in retrieving velocity gradient

Basically the same problem as for retrieving the pressure gradient using the *user_ getvar* utility routine. But now the *variable description* argument *Velocity.Gradient* causes the same type of crash, while the *variable description* argument *<locale >.Velocity.Gradient* functions correctly.

Also the velocity gradient, like the pressure gradient, can be retrieved using the additional variable method.

Utility routines crash

Certain utlitity routines take a *CACTION* argument. One retrieves data by setting this argument to *RETURN*. However, if the memory area occupied by the data is not released by the end of the subroutine the solver crashes because the memory area cannot be accessed. To overcome this issue the utility routine call for retrieval should always be paired with a *RELEASE* value as argument afterwards. This is not a bug, but important to mention because this is easily overseen.

User parameter retrieval fails for big lists

Large lists of values (comma separated) as user parameters in the loaded CCL files can cause errors when they are retrieved using the *user_ peek** function when the list is too long. This is probably because the solver loads the whole list as a character string of finite length before parsing the data to an array. This bug can be overcome by using separate variables for each item, and call them 'A1', 'A2', ... , 'AN', and load them into an array (via a DO loop) at initialization.

Wrong sound speed in isothermal simulations

When using isothermal simulations the sound speed returned by both *monitor points* as well as returned by the utility function *user_ getvar* is wrong. Surprisingly this is exactly $\sqrt{1.4}$ too large. The hypothesis is that the sound speed does not follow directly from the solution field, but is calculated afterwards via equations for the sound speed such as $\sqrt{\gamma RT}$. The adiabatic index γ used is then still 1.4, while it should be 1.0 for isothermal calculations. It can be worked around by calculating the sound speed in the subroutines or CEL via the other field quantities (pressure, temperature, etc).

Wrong array shape for *user_ getvar*

According to the documentation the shape of the returned array of scalar gradient should be $[NCOMPT, NLOC]$ ($[row, column]$), where $NCOMPT$ represents the number of components (cartesian coordinates in 3D, so 3 cells), and $NLOC$ is the number of locations for which the gradient is calculated. However, the format is actually $[NLOC, NCOMPT]$. The documentation is wrong [7].

iFort compiler for Windows causes problems

The Intel *iFort* Fortran compiler would not work in Windows. The compiler also costs $ 1900 for Windows, while it is free for Linux. Therefore it is recommended to use Linux as the platform of choice. The drawback is that the Fortran code is compiled platform specific, and thus the compiled code for Linux will not work on Windows PC's.

Other compilers cause problems

Other compilers such as *f77* and *g77* would not work for standard settings in the *cfx5mkext.ccl* file. The number of flags that can be set runs in the hundreds, and comes with little documentation and description. The standard flags are written for the *iFort* compiler. It might be possible to use the other compilers with the correct flags, but these could not be found and could be a cumbersome task to set by oneself.

X-Connection emulation in Windows is slow and buggy

To create a connection with the X Window system on a Linux pc in Windows, special software on the Windows PC is needed. Two applications, *Cygwin* and *Exceed*, are tested. Both show problems with graphical OpenGL viewports (which are used in *ANSYS CFX*). This is not solved. A workaround is to do the graphical part of pre-processing and post-processing on a Windows PC, and transfer files between computers.

Closing SSH connection causes solver termination

When an *ANSYS CFX* solver process runs which outputs data to the terminal, and the terminal window is closed, the connection is also closed and sends a *SIGHUP* signal to the process automatically. The process can then decide to kill itself or continue. In an apparently random way the process sometimes decides to kill itself, but usually continue. The way to work around this problem is to start *ANSYS CFX* with *nohup* as a prefix from the terminal window (e.g. *nohup cfx5*). This prevents *HUP* (hangup) signal to arrive at the process.

4.4 Single and double precision numbers

Single precision numbers use 32 bit of data, where 23 bits are used for the fraction part, 8 bits for the exponent, and 1 bit for the sign (plus or minus). 23 bits for the fraction results in 24 bits of total precision due to a implicit leading bit with a value of 1 (which is assumed zero when the exponent part is zero) (see fig. 4.1). Depending on the value of the decimal number to be stored, single precision numbers give 6 to 9 significant decimal digits precision. The decimal value can be calculated from the stored bits using eq. 4.1. Where e is the decimal value of the exponent. Not all decimal fractions can be exactly represented in binary, for example 0.1, but 0.125 can (2^{-3}).

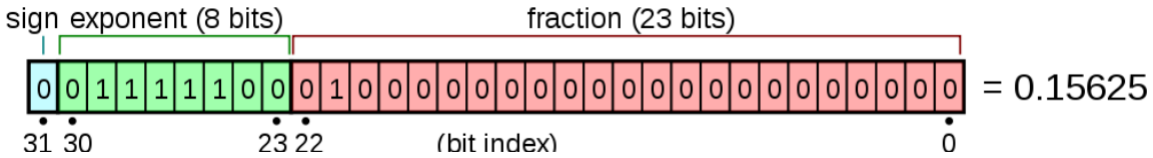


Figure 4.1: Single precision in memory [9]

$$value = (-1)^{sign}(1 + \Sigma_{i=1}^{23} b_{23-i} 2^{-i}) \cdot 2^{e-127} \tag{4.1}$$

Double precision numbers use 11 bits instead of 8 for the exponent, and 52 bits instead of 23 bits for the significand. Off course also here is an implicit bit used, so 24 bits effectively. Double precision gives 15 to 17 significant decimal digits precision, which is a lot more precise than 6 to 9 significant digits with single precision.

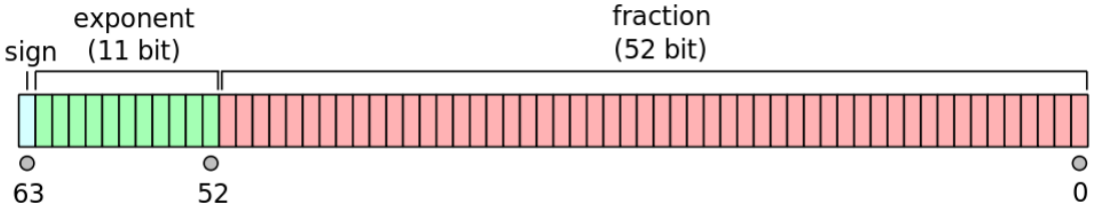


Figure 4.2: Double precision in memory [8]

Chapter 5

Simulation data

In this section all results from simulations (test cases) are shown. Table 5 shows all simulation parameters for both isothermal and total energy simulations.

Table 5.1: Parameters used in test cases

Parameter	Isothermal	Total energy
Simulation time	$0.2s$	
Characteristic length	$4m$	
Temperature	$25°C$ (fixed)	$25°C$ (initial)
Frequency	$100Hz$	
Characteristic sound speed	$346.13^m/_s$	$292.53^m/_s$
Sampled frequency range	$0.015 - 0.2s$	

The sampled frequency range parameter signifies the part of the time-series pressure data that is used in the multi-microphone method to determine the reflection. The first part is discarded because the wave has not arrived at the boundary yet, and mainly contributes to the first coefficient of the Fourier transform associated with a frequency of zero, which is constant (mean pressure over range).

Table 5.2 and 5.3 show how the different simulations are grouped and in which table they can be found for the nonreflective and TD-IBC simulations respectively.

Table 5.2: Non-reflective test cases

#	Group	No. Cases	Table
1	*ANSYS CFX* NRBC (Total energy)	16	5.4
2	*ANSYS CFX* NRBC (Isothermal)	16	5.5
3	Custom NRBC (Total energy)	16	5.6
4	Custom NRBC (Isothermal)	16	5.7
5	Custom NRBC+PWM (Total energy)	16	5.8
6	Custom NRBC+PWM (Isothermal)	16	5.9
7	Time integration schemes, custom NRBC (Total energy)	5	5.10
8	Heat transfer equations, *ANSYS CFX* NRBC	6	5.11
9	Heat transfer equations, custom NRBC	6	5.12
10	Influence of geometry	9	5.13
11	Influence of partitioning	9	5.14

Table 5.3: TD-IBC test cases

#	Group	No. Cases	Table
1	TD-IBC (Total energy)	4	5.22
2	TD-IBC (Isothermal)	4	**??**
3	*ANSYS CFX* NRBC, hard wall/pressure release (Total energy)	4	5.24

Table 5.4: (1) *ANSYS CFX* NRBC (Total energy) test cases

	Parameters			Reflection				
#	K	σ	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Cpu
1.1	1	0.0115	0	-0,0011	-0,0005	0,0012	-2,7486	3:13
1.2	10	0.1152	0	-0,0014	-0,0075	0,0077	-1,7536	3:16
1.3	23	0.265	0	-0,0020	-0,0177	0,0178	-1,6829	3:15
1.4	50	0.576	0	-0,0039	-0,0388	0,0390	-1,6713	3:16
1.5	100	1.152	0	-0,0098	-0,0772	0,0778	-1,6964	3:13
1.6	200	2.304	0	-0,0299	-0,1508	0,1537	-1,7668	3:12
1.7	500	5.76	0	-0,1412	-0,3278	0,3569	-1,9775	3:12
1.8	1000	11.52	0	-0,3748	-0,4600	0,5933	-2,2545	3:14
1.9	10000	115.2	0	-0,9493	-0,1228	0,9572	-3,0130	3:14
1.10	100000	1152	0	-0,9719	-0,0480	0,9731	-3,0923	3:15
1.11	23	0.265	0.5	-0,0028	-0,0185	0,0187	-1,7205	3:09
1.12	23	0.265	2.5	-0,0062	-0,0214	0,0223	-1,8535	3:36
1.13	23	0.265	5	-0,0102	-0,0252	0,0272	-1,9540	3:48
1.14	50	0.576	0.5	-0,0047	-0,0396	0,0399	-1,6890	3:12
1.15	50	0.576	2.5	-0,0081	-0,0424	0,0432	-1,7590	3:34
1.16	50	0.576	5	-0,0120	-0,0463	0,0478	-1,8245	3:45

Table 5.5: (2) *ANSYS CFX* NRBC (Isothermal) test cases

	Parameters			Reflection				
#	K	σ	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Cpu
2.1	1	0.0115	0	0,0813	-0,0068	0,0816	-0,0833	11:34
2.2	10	0.1152	0	0,0811	-0,0137	0,0822	-0,1679	11:35
2.3	23	0.265	0	0,0806	-0,0238	0,0840	-0,2869	11:36
2.4	50	0.576	0	0,0790	-0,0446	0,0907	-0,5142	11:32
2.5	100	1.152	0	0,0737	-0,0828	0,1109	-0,8434	11:29
2.6	200	2.304	0	0,0549	-0,1571	0,1665	-1,2347	8:24
2.7	500	5.76	0	-0,0556	-0,3436	0,3481	-1,7313	8:18
2.8	1000	11.52	0	-0,3093	-0,4971	0,5854	-2,1274	8:23
2.9	10000	115.2	0	-0,9898	-0,0762	0,9927	-3,0648	8:30
2.10	100000	1152	0	-0,9656	0,0658	0,9678	3,0735	8:27
2.11	23	0.265	0.5	0,0798	-0,0236	0,0832	-0,2875	9:50
2.12	23	0.265	2.5	0,0766	-0,2240	0,0798	-0,2839	9:58
2.13	23	0.265	5	0,0731	-0,0211	0,0761	-0,2812	9:54
2.14	50	0.576	0.5	0,0781	-0,0444	0,0899	-0,5168	9:55
2.15	50	0.576	2.5	0,0750	-0,0431	0,0865	-0,5215	9:47
2.16	50	0.576	5	0,0715	-0,0418	0,0828	-0,5291	9:53

Table 5.6: (3) Custom NRBC (Total energy) test cases

	Parameters		Reflection					P. drift	
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Cpu	Max.	Sta.
3.1	1	0	-0,01150	0,00640	0,01310	2,63420	3:37	CBD	CBD
3.2	10	0	-0,01140	-0,00160	0,01156	-3,00068	3:36	193.5	193.5
3.3	23	0	-0,01180	-0,01130	0,01630	-2,37627	3:36	120	CBD
3.4	50	0	-0,01270	-0,02910	0,03177	-1,98289	3:35	85	20
3.5	100	0	-0,01590	-0,06120	0,06320	-1,82530	3:11	70	32.5
3.6	200	0	-0,02850	-0,12360	0,12680	-1,79740	3:08	58.5	35
3.7	500	0	-0,10530	-0,28350	0,30240	-1,92630	3:12	36	36
3.8	1000	0	-0,29400	-0,43580	0,52570	-2,16430	3:12		
3.9	10000	0	-0,94820	-0,13870	0,95830	-2,99640	3:15		
3.10	100000	0	-0,97140	-0,04980	0,97270	-3,09040	3:15		
3.11	23	0.5	-0,00990	0,01190	0,01550	2,26500	3:06	151	151
3.12	23	2.5	0,04920	0,07420	0,08900	0,98540	3:35	81	81
3.13	23	5	0,11420	0,06390	0,13090	0,51030	3:41	37	37
3.14	50	0.5	-0,01060	-0,00610	0,01220	-2,61420	3:07	96	85
3.15	50	2.5	0,05120	0,05410	0,07450	0,81230	3:35	56	40
3.16	50	5	0,11610	0,04110	0,12310	0,34000	3:39	45	18

Table 5.7: (4) Custom NRBC (Isothermal) test cases

	Parameters		Reflection					P. drift	
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Cpu	Max.	Sta.
4.1	1	0	-0,0126	-0,0033	0,0131	-2,8859	7:18	6	6
4.2	10	0	-0,0127	-0,0092	0,0157	-2,5163	11:44	29.5	29.5
4.3	23	0	-0,0131	-0,0176	0,0219	-2,2124	11:49	33.5	33.5
4.4	50	0	-0,0140	-0,0349	0,0376	-1,9520	11:49	26.5	26.5
4.5	100	0	-0,0178	-0,0669	0,0692	-1,8307	11:42	29	29
4.6	200	0	-0,0315	-0,1291	0,1329	-1,8099	11:33	24	20
4.7	500	0	-0,1162	-0,2909	0,3133	-1,9507	11:36	12.5	12.5
4.8	1000	0	-0,3257	-0,4407	0,5480	-2,2072	11:38		
4.9	10000	0	-0,9895	-0,0767	0,9925	-3,0642	11:29		
4.10	100000	0	-0,9657	0,0658	0,9680	3,0735	11:34		
4.11	23	0.5	-0,0111	0,0062	0,0127	2,6294	10:06	167	167
4.12	23	2.5	0,0473	0,0719	0,0861	0,9888	9:52	100	44
4.13	23	5	0,1126	0,0686	0,1318	0,5472	9:56	48	48
4.14	50	0.5	-0,0111	-0,0110	0,0156	-2,3584	10:07	88	66
4.15	50	2.5	0,0499	0,0529	0,0728	0,8145	9:59	53	41
4.16	50	5	0,1151	0,0465	0,1242	0,3836	10:00	26	18

Table 5.8: (5) Custom NRBC+PWM (Total energy) test cases

	Parameters		Reflection					P. drift	
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Cpu	Max.	Sta.
5.1	1	0	-0,0115000	0,0070000	0,0135000	2,5928000	3:31	CBD	CBD
5.2	10	0	-0,0114000	0,0051000	0,0125000	2,7221000	3:35	206	206
5.3	23	0	-0,0115000	0,0040000	0,0122000	2,8051000	3:37	100	CBD
5.4	50	0	-0,0115000	0,0036000	0,0121000	2,8353000	3:38	53	CBD
5.5	100	0	-0,0114000	0,0039000	0,0121000	2,8110000	3:38	29	0
5.6	200	0	-0,0109000	0,0046000	0,0119000	2,7448000	3:37	14.5	0
5.7	500	0	-0,0091000	0,0060000	0,0109000	2,5591000	3:37	6.5	0
5.8	1000	0	-0,0059000	0,0066000	0,0089000	2,3003000	3:32		
5.9	10000	0	-0,0002675	-0,0007301	0,0007759	-1,9220000	3:32		
5.10	100000	0	-0,0008000	-0,0023000	0,0024111	-1,8982062	3:36		
5.11	23	0.5	-0,0095000	0,0275000	0,0291000	1,9024000	3:11	203	CBD
5.12	23	2.5	0,0486000	0,0905000	0,1027000	1,0783000	3:36	135	CBD
5.13	23	5	0,1135000	0,0811000	0,1395000	0,6205000	3:45	64	CBD
5.14	50	0.5	-0,0089000	0,0269000	0,0284000	1,8902000	3:13	103	CBD
5.15	50	2.5	0,0502000	0,0895000	0,1026000	1,0593000	3:38	68	68
5.16	50	5	0,1153000	0,0788000	0,1396000	0,5992000	3:43	32	32

Table 5.9: (6) Custom NRBC+PWM (Isothermal) test cases

	Parameters		Reflection					P. drift	
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Cpu	Max.	Sta.
6.1	1	0	-0,0126	-0,0026	0,0129	-2,9362	11:38	4	0
6.2	10	0	-0,0126	-0,0026	0,0128	-2,9411	11:47	4	0
6.3	23	0	-0,0125	-0,0025	0,0128	-2,9482	11:53	3	0
6.4	50	0	-0,0124	-0,0023	0,0126	-2,9627	11:54	3	0
6.5	100	0	-0,0123	-0,0019	0,0124	-2,9888	11:50	3	0
6.6	200	0	-0,0117	-0,0012	0,0118	-3,0370	11:49	3	0
6.7	500	0	-0,0099	0,0003	0,0099	3,1161	11:47	3	0
6.8	1000	0	-0,0066	0,0009	0,0067	3,0009	7:18		
6.9	10000	0	-0,0006	-0,0061	0,0061	-1,6744	7:20		
6.10	100000	0	-0,0011	-0,0077	0,0078	-1,7091	7:21		
6.11	23	0.5	-0,0112	0,0210	0,0238	2,0583	10:14	177	CBD
6.12	23	2.5	0,0462	0,0870	0,0985	1,0822	10:07	165	CBD
6.13	23	5	0,1113	0,0848	0,1399	0,6515	9:57	80	80
6.14	50	0.5	-0,0098	0,0215	0,0236	2,0007	10:09	95	95
6.15	50	2.5	0,0481	0,0869	0,0993	1,0650	10:04	85	85
6.16	50	5	0,1132	0,0831	0,1404	0,6333	9:56	40	40

Table 5.10: (7) Time integration schemes (Total energy)

	Params.		Reflection				
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Description
7.1	50	0	-0,0128	-0,0316	0,0341	-1,9559	First order, explicit L
7.2	50	0	-0,0127	-0,0291	0,0318	-1,9829	Second order, explicit L
7.3	50	0	-0,0127	-0,0291	0,0318	-1,9827	Third order, explicit L
7.4	50	0	-0,0131	-0,0367	0,0390	-1,9132	First order, implicit L
7.5	50	0	-0,0130	-0,0342	0,0366	-1,9333	First order, implicit predictor L

Table 5.11: (8) Influence of heat transfer options, *ANSYS CFX* NRBC

	Params.			Reflection				Heat eqs. options			
#	K	σ	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Heat equation	PT	VT	Cpu
8.1	50	0.576	0	0,1765	0,1443	0,2280	0,6854	Thermal	Off	Off	9:27
8.2	50	0.576	0	0,1765	0,1443	0,2280	0,6854	Thermal	Off	On	9:29
8.3	50	0.576	0	-0,0038	-0,0376	0,0378	-1,6725	Thermal	On	Off	4:27
8.4	50	0.576	0	-0,0038	-0,0376	0,0378	-1,6725	Thermal	On	On	4:25
8.5	50	0.576	0	-0,0039	-0,0388	0,0390	-1,6713	Total	N/A	Off	3:26
8.6	50	0.576	0	-0,0039	-0,0388	0,0390	-1,6713	Total	N/A	On	3:25

Table 5.12: (9) Influence of heat transfer options, Custom NRBC

	Params.			Reflection				Heat eqs. options			
#	K	σ	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Heat equation	PT	VT	Cpu
9.1	50	0.576	0	0,2491	0,2498	0,3528	0,7868	Thermal	Off	Off	8:06
9.2	50	0.576	0	0,2491	0,2498	0,3528	0,7868	Thermal	Off	On	8:07
9.3	50	0.576	0	-0,0122	-0,0294	0,0319	-1,9646	Thermal	On	Off	4:30
9.4	50	0.576	0	-0,0122	-0,0294	0,0319	-1,9646	Thermal	On	On	4:33
9.5	50	0.576	0	-0,0127	-0,0291	0,0318	-1,9829	Total	N/A	Off	3:21
9.6	50	0.576	0	-0,0127	-0,0291	0,0318	-1,9829	Total	N/A	On	3:25

Table 5.13: (10) Influence of geometry (Total energy)

	Params.		Reflection					
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Boundary	Description
10.1	50	0	-0,0037	-0,0382	0,0383	-1,6671	*ANSYS* NRBC	Mesh refined near BC
10.2	50	0	?	?	?	?	*ANSYS* NRBC	2D Mesh
10.3	50	0	?	?	?	?	*ANSYS* NRBC	3D Mesh
10.4	50	0	-0,0123	-0,0166	0,0207	-2,2064	Custom NRBC	Mesh refined near BC
10.5	50	0	?	?	?	?	Custom NRBC	2D Mesh
10.6	50	0	?	?	?	?	Custom NRBC	3D Mesh
10.7	50	0	-0,0143	0,0166	0,0219	2,2815	Custom NRBC+PWM	Mesh refined near BC
10.8	50	0	?	?	?	?	Custom NRBC+PWM	2D Mesh
10.9	50	0	?	?	?	?	Custom NRBC+PWM	3D Mesh

Table 5.14: (11) Influence of partitioning (Total energy)

	Params.		Reflection						
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Cpu	Boundary	Description
11.1	50	0	-0,0039	-0,0388	0,0390	-1,6713	3:16	*ANSYS* NRBC	Serial
11.2	50	0	-0,0039	-0,0388	0,0390	-1,6713	2:02	*ANSYS* NRBC	2 Partitions
11.3	50	0	-0,0039	-0,0388	0,0390	-1,6713	2:23	*ANSYS* NRBC	4 Partitions
11.4	50	0	-0,0127	-0,0291	0,0318	-1,9829	3:12	Custom NRBC	Serial
11.5	50	0	-0,0127	-0,0291	0,0318	-1,9829	2:00	Custom NRBC	2 Partitions
11.6	50	0	-0,0127	-0,0291	0,0318	-1,9829	2:19	Custom NRBC	4 Partitions
11.7	50	0	-0,0115	0,0036	0,0121	2,8353	3:16	Custom NRBC+PWM	Serial
11.8	50	0	-0,0115	0,0036	0,0121	2,8353	2:03	Custom NRBC+PWM	2 Partitions
11.9	50	0	-0,0115	0,0036	0,0121	2,8353	2:21	Custom NRBC+PWM	4 Partitions

Table 5.15: (12) Linear refinement for *ANSYS CFX* NRBC

	Parameters			Reflection			
#	K	σ	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ
12.1	1	0.0115	0	-0,00090273	0,00017813	0,00092014	2,9468
12.1	10	0.1152	0	-0,00120000	-0,00690000	0,00700000	-1,7420
12.1	23	0.265	0	-0,00180000	-0,01710000	0,01720000	-1,6750
12.1	50	0.576	0	-0,00370000	-0,03820000	0,03830000	-1,6671
12.1	100	1.152	0	-0,00950000	-0,07660000	0,07720000	-1,6943
12.1	200	2.304	0	-0,02960000	-0,15020000	0,15310000	-1,7656
12.1	500	5.76	0	-0,14080000	-0,32740000	0,35640000	-1,9770
12.1	1000	11.52	0	-0,37450000	-0,45980000	0,59310000	-2,2543
12.1	10000	115.2	0	-0,94940000	-0,12220000	0,95720000	-3,0136
12.1	100000	1152	0	-0,97190000	-0,04740000	0,97310000	-3,0929

Table 5.16: (13) Linear refinement for custom NRBC

	Parameters		Reflection				P. drift	
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Max.	Sta.
13.1	1	0	-0,0180	0,0133	0,0224	2,5033	220	CBD
13.2	10	0	-0,0152	0,0091	0,0177	2,6041	98	98
13.3	23	0	-0,0136	0,0010	0,0136	3,0655	51	39
13.4	50	0	-0,0123	-0,0166	0,0207	-2,2064	29	12
13.5	100	0	-0,0119	-0,0491	0,0505	-1,8085	34	res
13.6	200	0	-0,0175	-0,1122	0,1135	-1,7258	23	res
13.7	500	0	-0,0768	-0,2800	0,2903	-1,8387	37	res
13.8	1000	0	-0,2673	-0,4585	0,5307	-2,0986	51	res
13.9	10000	0	-0,9498	-0,1390	0,9599	-2,9963		
13.10	100000	0	-0,9714	-0,0492	0,9727	-3,0910		

Table 5.17: (14) Linear refinement for custom NRBC+PWM

	Parameters		Reflection				P. drift	
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Max.	Sta.
14.1	1	0	-0,0182	0,1390	0,0229	2,4902	230	CBD
14.2	10	0	-0,0167	0,0150	0,0228	2,4100	155	155
14.3	23	0	-0,0154	0,0159	0,0221	2,3380	99	99
14.4	50	0	-0,0143	0,0166	0,0219	2,2815	59	53
14.5	100	0	-0,0132	0,0171	0,0216	2,2284	29	29
14.6	200	0	-0,0115	0,0177	0,0212	2,1475	19	16
14.7	500	0	-0,0070	0,0183	0,0196	1,9367	5	5
14.8	1000	0	-0,0009	0,0161	0,0161	1,6253	1	1
14.9	10000	0	0,0016	-0,0020	0,0026	-0,9086		
14.10	100000	0	-0,0007	-0,0039	0,0039	-1,7500		

Table 5.18: (15) Smooth sine for *ANSYS CFX* NRBC

	Parameters			Reflection			
#	K	σ	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ
15.1	1	0.0115	0	-0,00110000	-0,00050000	0,00120000	-2,7479
15.1	10	0.1152	0	-0,00140000	-0,00750000	0,00760000	-1,7493
15.1	23	0.265	0	-0,00190000	-0,01770000	0,01780000	-1,6786
15.1	50	0.576	0	-0,00370000	-0,03870000	0,03890000	-1,6670
15.1	100	1.152	0	-0,00940000	-0,07710000	0,07770000	-1,6923
15.1	200	2.304	0	-0,02930000	-0,15060000	0,15340000	-1,7630
15.1	500	5.76	0	-0,14010000	-0,32780000	0,35650000	-1,9746
15.1	1000	11.52	0	-0,37340000	-0,46010000	0,59260000	-2,2525
15.1	10000	115.2	0	-0,94590000	-0,12130000	0,95360000	-3,0140
15.1	100000	1152	0	-0,96650000	-0,04970000	0,96780000	-3,0902

Table 5.19: (16) Smooth sine for custom NRBC

	Parameters		Reflection				P. drift	
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Max.	Sta.
16.1	1	0	-0,0115	0,0063	0,013112	2,6437	414	CBD
16.2	10	0	-0,0114	-0,0017	0,011566	-2,9953	208	208
16.3	23	0	-0,0117	-0,0113	0,016288	-2,3742	112	59
16.4	50	0	-0,0126	-0,0291	0,031701	-1,9793	75	22
16.5	100	0	-0,0157	-0,0611	0,063090	-1,8217	70	34
16.6	200	0	-0,0280	-0,1235	0,126582	-1,7936	60	38
16.7	500	0	-0,1042	-0,2835	0,302091	-1,9231	51	38
16.8	1000	0	-0,2927	-0,4359	0,525088	-2,1622	70	43
16.9	10000	0	-0,9456	-0,1369	0,955453	-2,9979		
16.10	100000	0	-0,9660	-0,0514	0,967383	-3,0884		

Table 5.20: (17) Smooth sine for custom NRBC+PWM

	Parameters		Reflection				P. drift	
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Max.	Sta.
17.1	1	0	-0,0115	0,0069	0,013429	2,6031	403	CBD
17.2	10	0	-0,0114	0,0050	0,012460	2,7299	204	CBD
17.3	23	0	-0,0115	0,0040	0,012192	2,8108	103	80
17.4	50	0	-0,0116	0,0036	0,012115	2,8401	55	13
17.5	100	0	-0,0114	0,0039	0,012076	2,8157	35	0
17.6	200	0	-0,0110	0,0045	0,011873	2,7496	19	0
17.7	500	0	-0,0092	0,0060	0,010921	2,5641	5	0
17.8	1000	0	-0,0060	0,0066	0,008886	2,3055	2	0
17.9	10000	0	-0,0003	-0,0007	0,000769	-1,9256		
17.10	100000	0	-0,0008	-0,0023	0,002405	-1,8957		

Table 5.21: (18) Special simulations

	Params.		Reflection				P. drift		
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Max.	Sta.	Boundary
Mean flow convergence of isothermal simulations									
18.1	50	25	0,04080000	-0,02900000	0,05000000	-0,6189			*ANSYS* NRBC
18.2	50	50	-0,00150000	-0,00470000	0,00490000	-1,8782			*ANSYS* NRBC
Very long simulation time to check convergence of drift. 1 second sim. time. Total E.									
18.3	50	0	-0,00980000	-0,02930000	0,03090000	-1,8946	85	20	custom NRBC
Linearly refined mesh, but with $\Delta t = 2 * 10^{-6}$, Total E.									
18.1	50	0	-0,00830	-0,01600	0,01800	-2,05010	27	15	custom NRBC

Table 5.22: (1) TD-IBC test cases (Total energy)

	Params.		Reflection				
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Description
1.1	50	0	1,0035	-0,0823	1,0069	-0,0818	Acoustically hard wall
1.2	50	0	-0,9669	0,0347	0,9675	3,1057	Pressure release surface
1.3	50	0	-0,0001	0,0100	0,0100	1,5841	Non-reflecting outlet
1.4	50	0	0,5219	0,4882	0,7146	0,7520	Complex $R = 0.5 + 0.5i$

Table 5.23: (2) TD-IBC test cases (Isothermal)

	Params.		Reflection				
#	K	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Description
2.1	50	0	1,0095	0,0623	1,0114	0,0617	Acoustically hard wall
2.2	50	0	-0,8767	0,0459	0,8779	3,0893	Pressure release surface
2.3	50	0	-0,0005	0,0045	0,0045	1,6906	Non-reflecting outlet
2.4	50	0	0,4868	0,5022	0,6994	0,8010	Complex $R = 0.5 + 0.5i$

Table 5.24: (3) Ansys hard wall and constant pressure

		Reflection				
#	u_0	$\mathcal{R}$	$\mathcal{I}$	$\|R\|$	ϕ	Description
3.1	0	-0,9767	-0,0387	0,9775	-3,1020	Constant pressure (Total energy)
3.2	0	-0,9628	0,0802	0,9661	3,0585	Constant pressure (Isothermal)
3.3	0	0,9944	-0,1034	0,9998	-0,1036	Hard wall (Total energy)
3.4	0	1,0142	0,0439	1,0151	0,0433	Hard wall (Isothermal)

Chapter 6

Source code

The source code of the *FORTRAN* subroutines, the *MATLAB* scripts and classes, the *CCL* files, and the used filter coefficient arrays for the construction of the TD-IBCs, are added to the supplied CD-ROM. The UML diagram and two example CCL files are added to this chapter.

Supplied on CD-ROM:

- *PLinit FORTRAN* subroutine source
- *PTloop FORTRAN* subroutine source, NRBC version
- *PTloop FORTRAN* subroutine source, TD-IBC version
- *Pset FORTRAN* subroutine source, NRBC version
- *Pset FORTRAN* subroutine source, TD-IBC version
- *MultiMicMethod MATLAB* class, including example run script.
- CCL files, including filter coefficient sets for TD-IBC.
- *ANSYS CFX PRE* files
- All P_1 ... P_4 pressure time series in *.csv file.

6.1 *MultiMicMethod MATLAB* class

The complete UML diagram. The source code is added to the supplied CD-ROM.

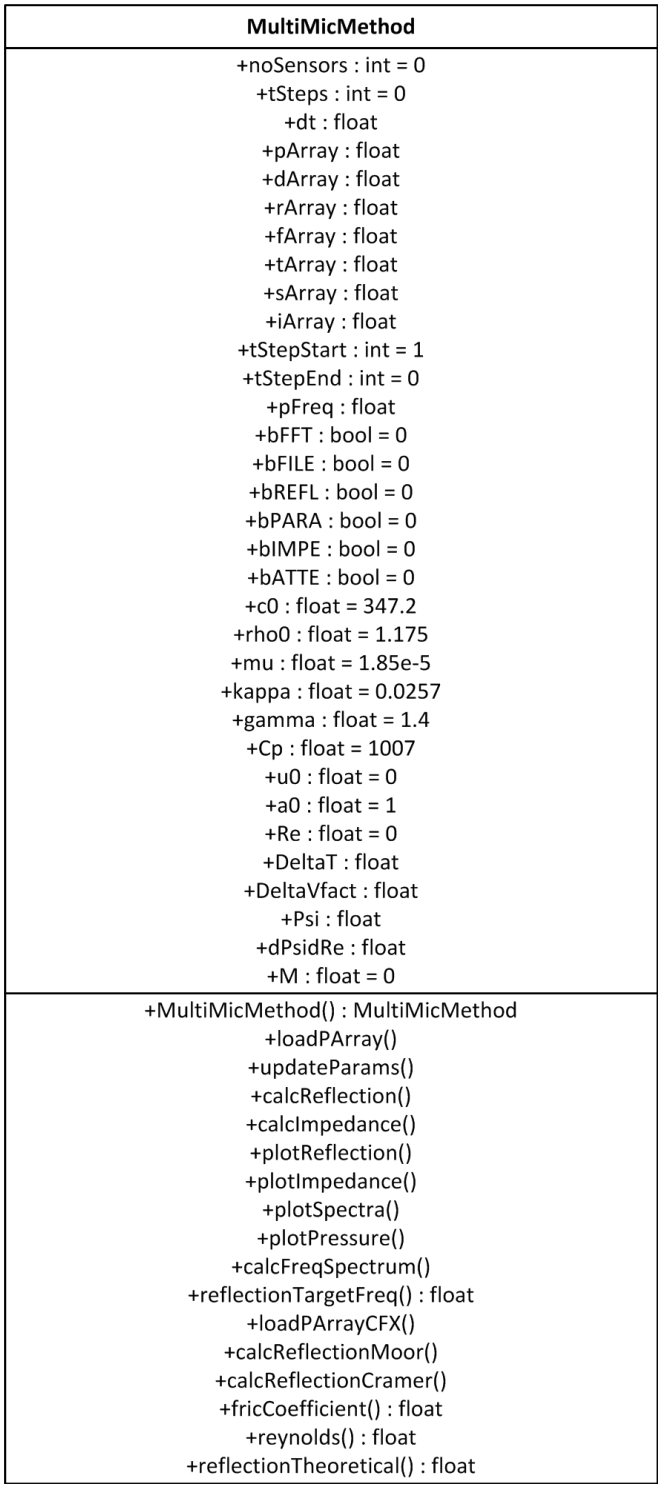


Figure 6.1: Complete UML diagram for MultiMicMethod class

6.2 CCL file with *user parameters*

The CCL file (with example settings) for both the NRBC as well as the TD-IBC.

CCL file example for NRBC

```
USER:
        # ====== Model Properties ======
        # Relative far-field pressure [Pa]
        pinf  = 0.0
        # Mean flow speed [m/s]
        u = 0.5
        # Relaxation factor K [1/s]
        K = 50.0
        # Distance of sample plane from Non-reflective BC [m]
        DXSAMPLE = 0.4
        # Use Plane-Wave Masking? (0:No, 1:Yes)
        PWM = 1
        # ====== Fluid Properties ======
        # Adiabatic index gamma
        gamma = 1.4
        # Specific gas constant
        RSPEC = 287.058
END
```

CCL file example for TD-IBC

```
USER:
        # ====== Model Properties ======
        # Relative far-field pressure [Pa]
        pinf  = 0.0
        # Mean flow speed [m/s]
        u = 0.0
        # Relaxation factor K [1/s]
        K = 50.0
        # Distance of sample plane from Non-reflective BC [m]
        DXSAMPLE = 0.4
        # Use Plane-Wave Masking? (0:No, 1:Yes)
        PWM = 1
        # ====== Fluid Properties ======
        # Adiabatic index gamma
        gamma = 1.4
        # Specific gas constant
        RSPEC = 287.058
        # ====== TD-IBC filter constants ======
        # Number of filter constants
        NFC = 10
        # A filter constants
        AFC = 0, 0, 0, 0, 0, 1, 0, 0, 0, 0
        # B filter constants
        BFC = 1, 0, 0, 0, 0, 0, 0, 0, 0, 0
END
```

6.3 Filter coefficient sets

If not mentioned, the coefficient is equal to zero. A minimum of ten coefficients need to be specified, and all coefficients need to be specified until the last non-zero coefficient (equal for both A and B). If only coefficient # 1 is non-zero, the others need to be specified with zero. If coefficient # 350 is non-zero, every coefficient to # 350 need to be specified, even if they are zero. Coefficient $B1$ always needs to be equal to 1.

Acoustically hard wall:

A0=1.0000
B0=1.0000

Pressure release surface:

A499=1.0000
B0=1.0000

Non-reflecting outlet:

B0=1.0000

Complex $R = 0.5 + 0.5i$ at fixed frequency of $f = 100Hz$.

A124=0.7071
B0=1.0000

Bibliography

[1] Seung-ho Jang and Jeong-guon Ih. On the multiple microphone method for measuring in-duct acoustic properties in the presence of mean flow. 103(3):1520–1526, 1998.

[2] Michael G Jones and Tony L Parrot. Evaluation of a Multi-Point Method for Determining Acoustic Impedance. *Mechanical Systems and Signal Processing*, 3:15–35, 1989.

[3] Joris Oosterhuis. Performance of non-reflective boundary condition in ANSYS CFX. Technical report, 2012.

[4] T.J. Poinsot. Boundary conditions for direct simulations of compressible viscous flows. *Journal of Computational Physics*, 99(2):352, April 1992.

[5] Laurent Selle. The actual impedance of non-reflecting boundary conditions : implications for the computation of resonators. pages 1–21, 2003.

[6] W Thompson. Time- Dependent Boundary Conditions for Hyperbolic Systems , II. *Journal of Computational Physics*, 439461:439–461, 1990.

[7] Ansys Tutorials. ANSYS CFX-Solver Modeling Guide. 15317(November):724–746, 2011.

[8] Wikipedia. Double-precision floating-point format. 10(2):6–9, 2013.

[9] Wikipedia. Single-precision floating-point format. 24:4–9, 2013.

Nomenclature

BC	Boundary Condition	
DTFT	Discrete-Time Fourier Transform	
EOS	Equation(s) Of State	
LODI	Locally One-Dimensional Inviscid	
LRM	Linear Relaxation Method	
MMM	Two-Microphone Method	
NSCBC	Navier-Stokes Characteristic Boundary Conditions	
PDE	Partial Differential Equation	
SWR	Standing Wave Ratio	
SWR	Standing Wave Ratio	
TMM	Two-Microphone Method	
K	Relaxation factor	$[1/s]$
L	Characteristic length (e.g. domain length)	$[m]$
R	Reflection coefficient	$[-]$
Re	Reynolds number	
Γ	Plane-wave propagation constants	
Ψ	Coefficient of friction for turbulent flow	
$\delta\rho$	Acoustic density	$[kg/m^3]$
δp	Acoustic pressure	$[Pa]$
δu	Acoustic velocity	$[m/s]$
δ	Attenuation constant in Multi-Microphone Method	
δ_t	Attenuation due to turbulent effects	
δ_ν	Attenuation due to viscothermal effects	
γ	Adiabatic index	
$\hat{p}$	Acoustic pressure phasor	$[Pa]$

$\hat{p}_+$	Forward traveling pressure phasor	$[Pa]$
$\hat{p}_-$	Backward traveling pressure phasor	$[Pa]$
$\hat{u}$	Acoustic particle velocity phasor	$[m/s]$
κ	Heat conduction coefficient	
λ	Eigenvalues of $\mathbf{A}$ / Characteristic speed LODI-relations	$[m/s]$
$\mathbb{C}$	Set of complex numbers	
$\mathbf{A}^+$	Moor-Penrose psuedo-inverse of $\mathbf{A}$	
$\mathbf{A}^H$	Hermitian matrix of $\mathbf{A}$	
$\mathcal{F}[]$	Fourier transform	
$\mathcal{L}_\rangle$	Amplitude variation of characteristic waves	$[varies]$
$\mathcal{L}[]$	Laplace transform	
$\mathcal{M}$	Mean flow Mach number	
$\mathcal{R}$	Resistive part of impedance	$[Rayl]$
$\mathcal{X}$	Reactive part of impedance	$[Rayl]$
$\mathcal{Z}$	Impedance	$[Rayl]$
$\mathcal{Z}_0$	Characteristic impedance	$[Rayl]$
μ	Shear viscosity constant	
ω	Angular frequency	
ϕ	Velocity potential	$[m^2/s]$
ρ_0	Mean density	$[kg/m^3]$
σ	Coupling parameter	$[1/s]$
τ	Time coordinate	$[s]$
a_0	Ratio of duct area to perimeter	
c_0	Small signal sound speed	$[m/s]$
d	Position coordinate (like x)	$[m]$
k	Wave number	$[rad/m]$
p_0	Mean pressure	$[Pa]$
u_0	Mean velocity	$[m/s]$