ISRN LUTMDN/TMHP-14/5316-SE

ISSN 0282-1990

Modelling of Wave Propagation in Combustion Engines

Emma Gustavsson

Examensarbete på Civ. ingenjörsnivå
Avdelningen för Förbränningsmotorer
Institutionen för Energivetenskaper
Lunds Tekniska Högskola | Lunds Universitet

Modelling of Wave Propagation in Combustion Engines

Emma Gustavsson

June 2014, Lund

Föreliggande examensarbete på civilingenjörsnivå har genomförts vid Avd. för Förbränningsmotorer, Inst. för Energivetenskaper, Lunds Universitet - LTH samt vid Modelon i Lund. Handledare på Modelon: Daniel Andersson; handledare på LU-LTH: Prof. Per Tunestål; examinator på LU-LTH: Docent Martin Tunér.

Examensarbete på Civilingenjörsnivå

ISRN LUTMDN/TMHP-14/5316-SE

ISSN 0282-1990

Förbränningsmotorer
Institutionen för Energivetenskaper
Lunds Universitet - Lunds Tekniska Högskola
Box 118, 221 00 Lund

www.energy.lth.se

Abstract

In engine design and control a phenomenon having a large impact is the pressure waves in the intake and outlet manifolds. The transient behaviour of the flow will affect the performance of an engine and is therefore important to be modelled correctly. There exist a number of different methods solving transient flow situations, mainly divided into two groups: lumped- and distributed parameter models. The first are often lacking in physical accuracy, for example by using mean valued methods, but are easy to simulate. The latter are accurate but are often heavy to simulate. In this report a lumped-parameter model called the Quasi-Propagatory Method, the QPM, is presented. This method takes into account the inertia of the fluid and captures the waves in a physically adequate manner. The lumping property of the model enables the possibility of simulation in real-time, still the method is accurate and true to nature. Starting from the governing equations for unsteady flow and using theory from the Method of Characteristics, a dynamic expression for the velocity in a pipe is found. In this report the theory behind, and an implementation in Modelica of the Quasi-Propagatory Method is given. The method presented is general and suffers from the need of solving non-linear systems of equations which may in certain cases prohibit real-time simulations. A loss in generality would however solve this problem which is described in the report. The model is found to give reasonable results that captures transients and is then compared to two other models. The comparisons are not optimal, since the assumptions made in the different models are not equivalent and none is proven to be superior, therefore experimental results would be desirable to ensure the applicability of the model.

Acknowledgements

First of all I would like to thank my advisors Daniel Andersson, at Modelon, and Per Tunestål, at the Energy Science Department at Lund University, for helping and guiding me through the project. I would also like to thank Modelon for the opportunity to do my master thesis there. Special thanks to Antonio Sciarretta for taking time to answer my questions about your articles. Lastly, thanks to father, family and friends for supporting me through the project.

Contents

Nomenclature

Lower case

a	local speed of sound [m/s]
c	wave propagation speed [m/s]
f	Fanning friction factor [-]
f_D	Darcy-Weisbach friction factor [-]
h	specific enthalpy [J/kg]
k_0	model parameter (=1) [-]
p	pressure in the middle of the pipe [kg/(m·s^2)]
p_{u0}	pressure of linearised upstream boundary condition u=0 [kg/(m·s^2)]
p_{d0}	pressure of linearised downstream boundary condition u=0 [kg/(m·s^2)]
p_s	critical pressure (sonic limit) [kg/(m·s^2)]
q	heat transfer per unit mass per unit time [J/(kg · s)]
t	time [s]
u	velocity in x-direction in the middle of the pipe [m/s]
x	spatial coordinate [m]

Upper case

$\mathcal{A}$	dimensionless local speed of sound [-]
A	slope of upstream boundary condition [kg/(m^2·s)]
B	slope of downstream boundary condition [kg/(m^2·s)]
C	characteristic slope [kg/(m^2·s)]
D	diameter of pipe [m]
F	cross-section area of pipe (circular) [m^2]
G	friction force per unit mass [m/s^2]
L	length of pipe [m]
$\dot{Q}$	net heat energy transfer [J/s]
R	specific gas constant [J/(mol · K)]
U	dimensionless velocity [-]

Greek Letters

$\alpha_{1,2}$	non-homentropic term [kg/(m $\cdot$ s^2)]
$\Delta_{1,2,3}$	non-homentropic term [m/s^2]
Δp	pressure difference [kg/(m$\cdot$s^2)]
κ	specific heat ratio [-]
λ	model parameter [-]
ξ	model parameter [-]
τ	time constant [s]
ρ	density [kg/m^3]
ϕ	area ratio downstream [-]
ψ	area ratio upstream [-]
ω	pulsation [rad/s]

Subscripts

∞	steady-state
d	downstream end
int	intersection
u	upstream end
j	time succession

Abbreviations

CFD	Computational Fluid Dynamics
MOC	Method of Characteristics
MTF	Method of Transfer Functions
QPM	Quasi-Propagatory Method
SBF	Spatial Base Functions

Chapter 1

Introduction

It has long been known that the performance of an engine depends on the transient behaviour of the flow in the intake- and exhaust manifolds. In [1](p. 13) it is stated that as early as 1938, Framer refers to work from one hundred years before, where it was found that the volumetric efficiency, that is connected to the performance, of an engine can be affected by the length of the exhaust pipe. Due to the periodic process in an internal combustion engine, pressure waves will travel back and forth in the manifolds, affecting the mass flow into and out of the cylinder and thus the performance of the engine. Ergo, it is important that these transients are correctly modelled to be able to optimise the volumetric efficiency when designing an engine. Another aspect that is highly up-to-date is the importance of being able to estimate the amount of air inside the cylinder to optimise the fuel consumption and the amount of emissions in order to reduce the environmental impacts.

In this work the aim is to find a model to simulate the pressure waves in the manifolds in real-time. There are many methods existing to solve the governing equations for a flow, but to find an accurate model with the capacity to be solved in real-time is a challenge. The focus in this report will be on a method called the Quasi-Propagatory Method. First, some other methods will be mentioned and given a brief overview below, with focus on methods aiming for real-time simulations. In the second chapter a deeper description of the theory behind the problem and the Quasi-Propagatory Method is given. This is followed up by a discussion of some implementational issues when implementing the method and some preliminary tests are provided to give an insight into how the model works. Finally, comparisons with other models, from Modelon's library and provided through [2], are made and results are presented and discussed. In the Appendix, derivations of boundary conditions used, and the implementation of the model in Modelica can be found. Also, some extra derivations are given for certain equations in the Appendix for interested readers.

1.1 Methods for Solving Pipe Flow

In order to solve the pressure waves in a pipe, many models are available, with their pros and cons. By manipulating the governing equations, the equations describing the flow, and make different simplifying assumptions the models provide results with more or less accuracy. General assumptions made in e.g. system- and control design in engine applications are that the flow can be considered as one-dimensional and that the fluid can be modelled as an ideal. One of the earliest approaches to solve the unsteady flow was the Method of Characteristics, MOC. This method was dominating earlier, but is now replaced by finite-differences, FD, and finite-volume, FV, methods [1](p. 54). These methods are accurate but computationally heavy since they are solving an approximate pressure profile on a grid throughout the whole region. Models using this approach are called distributed-parameter models. The opposite, models capturing the nature of the flow in discrete points are called lumped-parameter models. One widely used lumped-parameter model is the filling-emptying method, connecting the pressure difference over a region with a mean velocity of the flow within the region. This method models the flow in a steady manner, ignoring the inertia of the fluid, thus the velocity is changing immediately with a change of pressure. The MOC, FV and FD methods are accurate and may be necessary when simulating three-dimentional flows. However, in a pipe one can, as said, in general assume one-dimensional flow as a compromise between accuracy and computational time. A number of works have been performed trying to find a more efficient method to capture the pressure wave phenomenon without loss in accuracy in one dimension, some of them will be reviewed in the following.

The first method to be mentioned, found in [3], uses analogies with mass-spring-damper systems and the transfer functions describing these. One considers the compressibility of the gas as equivalent to a spring, the air in the pipe as a mass and the viscosity as a damper. Through the Laplace transform a transfer function can be identified connecting the pressure with the mass flow in the frequency domain. Not having to solve the differential equations in the time domain makes this method efficient and possible to simulate in real-time. However, the transfer function depends on empirically determined parameters and is thereby only applicable in certain cases and configurations. It is shown that the parameter can be determined analytically, as a function of the length and diameter of the pipe, when using a simple geometry. In these simplified cases this is a cheap method that gives accurate results. A major drawback is nevertheless this dependance on experimental data, further investigations and new experiments would have to be made to include for example junctions, throttles and temperature differences.

The next method, found in [4], uses empirically determined spatial base functions, SBF, onto which the partial differential equations, the governing equations, are projected to obtain ordinary differential equations. These are then solved using a finite volume scheme on a grid, staggered in time and space. The use of SBF allows the properties of the fluid to be non-uniform over the control volume, giving the possibility to reduce the number of cells without reducing the accuracy. This approach

could give the possibility of real-time simulations, but, as in the previous case, the need of experimentally, or through CFD simulations, decided base functions, depending on both geometry and boundary conditions is an inconvenience limiting the applicability of this method.

One last method that could be interesting to investigate further in a future work is the Method of Transfer Functions, MTF. In this method the flow is modelled in analogy with electrical phenomena and the equations are again solved using the Laplace transform. In [5] this method is improved including non-homentropic conditions and more accurate modelling of the boundary conditions. It is then said to have the same accuracy as the Method of Characteristics and the Quasi-Propagatory Method, and less computational time.

Finally, the method that is focused on in this report, the Quasi-Propagatory Method, will be introduced. The Quasi-Propagatory Method is a lumped-parameter model that, in contrast with the filling-emptying method, includes the inertia of the fluid. The method is described in [6], [7] and [2]. In the Quasi-Propagatory Method, further on called the QPM, the geometry is divided into one or more branches connected to each other by finite volumes called capacities. In the capacities the pressure and the temperature of the fluid are evaluated. When knowing the pressure and temperature in the capacities, the velocity in the branches in-between the capacities can be calculated in the middle points. The velocities in the branches affect the pressures in the capacities that will change and thus change the velocities in the branches and so on. One wants to link the pressure difference over the branch with its middle-point velocity and this is done through an analysis of waves using the Method of Characteristics. Through MOC the proceeding of velocity states in the pipe towards steady state is provided and with given boundary conditions, dynamical expressions for the velocity in the middle point is found. The idea is that a system will always strive, as described by the dynamic expression, towards steady state. When the boundary conditions are changing the steady value of the velocity is changing and through the dynamic model the velocity will strive towards this new value. The lumped-parameter approach of the method decreases the computational time, and a physically authentic derivation of equations used, increases the accuracy of the method. In [6], [7] and [2] the method is shown to give accurate results. The reason this method was chosen was that it is relatively well-documented, and has a low enough complexity to be implemented and simulated in real-time.

Chapter 2

Theory

In this chapter the necessary theory will be given, starting with the governing equations describing the flow, followed by an introduction to the flow situation in engine manifolds. Finally, the derivation of the method to be used, the Quasi-Propagatory Method, is given. Some equations are derived in more detail in Appendix B for the interested reader.

2.1 Governing Equations

The governing equations describing the flow in a pipe are the continuity equation, momentum equation and energy equation for unsteady and compressible flow. In this case the study of the pressure waves in the intake and exhaust manifolds in an engine are of interest. For a pipe section in the manifolds the flow can be treated as one-dimensional if the ratio of the pipe's length and diameter is large enough for the flow to become fully developed, a valid assumption in the pipe sections of an engine manifold where the ratio is usually small and the velocity is large, [8] (p. 6-7). This assumption is not valid when considering bends or junctions of a pipe, which is often compensated using empirically determined loss coefficients [1](p. 27). The governing equations describing this case are given in [9](p. 62-68), simplified for an ideal gas in a pipe with cross-section area F, possibly varying with x. They are the continuity equation

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + \rho\frac{\partial u}{\partial x} + \frac{\rho u}{F}\frac{dF}{dx} = 0, \tag{2.1.1}$$

the momentum equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{1}{\rho}\frac{\partial p}{\partial x} + G = 0 \tag{2.1.2}$$

and the energy equation

$$\frac{\partial p}{\partial t} + u\frac{\partial p}{\partial x} - a^2\left(\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x}\right) - (\kappa - 1)\rho(q + uG) = 0, \tag{2.1.3}$$

where $G = \frac{1}{2}u|u|f\frac{4}{D}$, with u as the velocity in the x-direction, D as the diameter of the pipe and f as the Fanning's friction factor. Furthermore, the pressure of the fluid is denoted p, the local speed of sound a, the density ρ and the specific heat ratio κ. Finally, the heat transfer energy per unit mass and unit time q, found in the energy equation, is calculated through the net heat energy transfer rate $\dot{Q}$ as $q = \frac{\dot{Q}}{\rho FL}$, where L is the length of the pipe. The governing equations are partial differential equations that are not possible to solve analytically; numerical methods are needed.

2.2 Four-Stroke Engines

A four-stroke engine is characterised by, as the name suggests, four strokes in each cycle. The first is the intake stroke, the second the compression stroke, the third is called the power stroke and the last is the exhaust stroke. During the intake stroke, the intake valve is open and the piston is moving downwards, increasing the volume in the cylinder making the fluid in the intake duct flow into the cylinder. This fluid may be air or air mixed with fuel, possibly also mixed with exhaust gas when using EGR, depending on the type of engine. In the next stroke, the compression stroke, the intake valve closes and the piston is now going upwards making the pressure increase in the cylinder. In the beginning of the power stroke the combustion is started increasing the pressure even more, causing the piston to be pushed down, performing work. Finally, the exhaust valve opens and the exhaust stroke starts when the piston is at the bottom. The piston is going up causing the combustion gases to exit the cylinder before the cycle repeats [10](p. 3).

2.2.1 Pressure Waves

In the following, the focus will be laid on the first and the last stroke in the cycle, investigating the pressure waves developed when the valves are opened and closed. The reason for this interest is that the pressure at the intake valve to a considerable degree influences the performance of the engine. The performance of the engine can is dictated by the volumetric efficiency that is the ratio between the amount of air that can be ingested in the cylinder and the maximal amount of air that theoretically can be contained in the cylinder [1](p. 8). Since the mass flow is connected to the pressure difference over the valve, a high pressure at the inlet valve will give a high mass flow into the cylinder, thus increasing the volumetric efficiency. One part of engine design is to consider this and optimise the length of the intake duct, ensuring the pressure wave to arrive at the right time to maximise the amount of air in the cylinder. Another interest is to be able to predict the transient behaviour in real-time to use online in control systems. It is therefore crucial to correctly model the behaviour of these pressure waves [1](p. 14). This consideration of the transient behaviour is mainly of interest in naturally aspirated engines. In contrast, turbocharged engines, using compressors and turbines to force

air into the cylinders, will not be as affected by these phenomena [1](p. 5).

The pressure waves emerge when the valves are open and the movement of the piston is causing the pressure in the cylinder to decrease, when the intake valve is open and the piston is moving downwards, or increase, when the exhaust valve is open and the piston is going upwards. This reduction, respectively enhancement, in pressure will propagate into the intake or exhaust pipe respectively. In the intake pipe a rarefaction wave will be developed and travelling through the pipe. When reflected at the end of the pipe this rarefaction wave will become a compression wave, due to the open-end boundary, travelling back towards the cylinder. Optimally, the compression wave would reach the intake valve when it is closing, boosting the inflow. At the exhaust pipe, a compression wave will propagate from the cylinder towards the end of the pipe, where it is being reflected and becomes a rarefaction wave. Depending on the period of the valves, if the intake and exhaust valves are open partially at the same time, this wave could propagate into the intake pipe, affecting the flow situation [10](p. 459-462). This is yet another thing to consider when optimising the design of an engine. The rarefaction wave can also benefit the scavenging of the cylinder, i.e. the emptying and filling of the cylinder [1](p. 206).

2.2.2 The Manifolds

Simplified, the intake manifold of an engine consists of one or more pipes, one for each cylinder in the engine, with either one air filter for each pipe or a single filter from which the airflow can be distributed into the different pipes. Upstream of the inlet manifold there is in gasoline engines a throttle valve used to control the mass flow into the cylinder, [11] (p. 83).

The exhaust manifold is essentially the reverse of the intake manifold, the exhaust gas from the cylinders is collected into one exhaust pipe [11] (p.88-92). In the exhaust system there is also a silencer and a catalytic converter.

The flow and the propagation of pressure waves through junctions are a multi-dimensional problem that is hard to represent and solve in one- dimension. Instead one often relies on empirical studies providing loss coefficients. The same holds for the throttle, the air filter, the silencer and the catalytic converter, as well as the valves. The loss coefficients are given for most situations in tables and enables calculations of the pressure drop caused by the component.

EGR, exhaust gas recirculation, meaning that some of the exhaust gas is mixed with the inflow can be handled by considering a new fluid. Based on information about the exhaust gas, new properties of the fluid can be determined.

2.3 Method of Characteristics

In this section the Method of Characteristics, MOC, will be briefly summarised. For further investigations a detailed analysis can be found in [9].

To begin with, different sources use different foci; in [9], the analysis is made with respect to u and a, the velocity and the velocity of sound, whereas [6], [7] and [2] are considering u and p, the velocity and the pressure, as independent variables. For an ideal gas with constant specific heats it holds that $a^2 = \frac{\kappa p}{\rho}$, so the analysis can easily be translated from one set of states to the other.

Now, starting by combining the governing equations to two new equations as $\frac{1}{\rho a}$(Eq. 2.1.3 + a^2 Eq. 2.1.1 $\pm$ ρa Eq. 2.1.2) = 0 (here written with notations as in [2]),

$$\frac{1}{C}\left(\frac{\partial p}{\partial t} + (u+a)\frac{\partial p}{\partial x}\right) + \left(\frac{\partial u}{\partial t} + (u+a)\frac{\partial u}{\partial x}\right) + \Delta_1 + \Delta_2 + \Delta_3 = 0 \tag{2.3.1}$$

and

$$\frac{1}{C}\left(\frac{\partial p}{\partial t} + (u-a)\frac{\partial p}{\partial x}\right) - \left(\frac{\partial u}{\partial t} + (u-a)\frac{\partial u}{\partial x}\right) + \Delta_1 + \Delta_2 - \Delta_3 = 0. \tag{2.3.2}$$

Here $C = \frac{\kappa p}{a}$, $\Delta_1 = -\frac{(\kappa-1)(q+uG)}{a}$, $\Delta_2 = \frac{au}{F}\frac{dF}{dx}$ and $\Delta_3 = G$. The energy equation can be written as

$$\frac{1}{C}\left(\frac{\partial p}{\partial t} + u\frac{\partial p}{\partial x} - a^2\left(\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x}\right)\right) + \Delta_1 = 0. \tag{2.3.3}$$

The idea is now to find lines, so called characteristics, along which these partial differential equations can be transformed to a set of ordinary differential equations. The solution of the partial differential equations is of the form $p = p(x,t)$, $u = u(x,t)$. Assuming there exists a function $c = c(p(a), u)$ uniquely connecting the pressure and the velocity, the solution is of the form $c = c(x,t)$. By the chain rule $\frac{dp}{dt} = \frac{\partial p}{\partial t} + \frac{\partial p}{\partial x}\frac{dx}{dt}$ and $\frac{du}{dt} = \frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\frac{dx}{dt}$. Now, defining the characteristic lines as $\frac{dx}{dt} = c$ and taking $c(p(a), u) = u + a$, Eq. 2.3.1 is given as an ordinary differential equation. That is, the characteristics for Eq. 2.3.1 are given by

$$\frac{dx}{dt} = u + a, , \tag{2.3.4}$$

$$\frac{dp}{dt} = \frac{\partial p}{\partial t} + (u+a)\frac{\partial p}{\partial x} \tag{2.3.5}$$

and

$$\frac{du}{dt} = \frac{\partial u}{\partial t} + (u+a)\frac{\partial u}{\partial x}. \tag{2.3.6}$$

Using these expressions Eq. 2.3.1 can be written as

$$\frac{1}{C}\frac{dp}{dt} + \frac{du}{dt} + \Delta_1 + \Delta_2 + \Delta_3 = 0. \tag{2.3.7}$$

This is called the compatibility equation and Eq. 2.3.4 is called the direction equation. The direction equation gives the slope of the characteristics in the x-t plane and the compatibility equation gives the relation between the states p and u along these lines.

In the same way one gets the other direction equations and their compatibility equations. To summarise there are three sets of characteristic equations: $\frac{dx}{dt} = u \pm a$ with

$$\frac{1}{C}\frac{dp}{dt} \pm \frac{du}{dt} + \Delta_1 + \Delta_2 \pm \Delta_3 = 0 \tag{2.3.8}$$

and $\frac{dx}{dt} = u$ with

$$\frac{1}{C}\left(\frac{dp}{dt} - a^2\frac{d\rho}{dt}\right) + \Delta_1 = 0. \tag{2.3.9}$$

The characteristic equations are ordinary differential equations that, when solved using the correct time step, as described by the direction equation, will describe the unsteady flow, thus including the pressure waves.

2.4 The Quasi-Propagatory Method

In the Quasi-Propagatory Method, the QPM, analysis of waves using MOC provides a derivation of a dynamical model that links the pressure difference over a pipe to the velocity in the middle of the pipe. As previously mentioned, this is called a lumped-parameter model, since the whole pipe is described in one velocity. In the following the derivation of the dynamical model is performed starting with a simplified QPM, assuming homentropic flow. Finally, the QPM for non-homentropic flow is introduced.

2.4.1 Homentropic Flow

In the following, homentropic flow will be considered in a pipe of constant cross-section area. For homentropic flow a graphical representation of the problem can be provided, giving a feeling for how the method works. Homentropic flow is defined as a flow where the entropy is uniform in time and space. For now it will be assumed that the flow has no heat transfer or wall friction, that is $q = 0$ and $f = 0$, as described in [1] (p.32). In this case $\Delta_1 = \Delta_2 = \Delta_3 = 0$. The energy equation is now $a^2 = \frac{dp}{d\rho}$, which is the relation between the pressure, density and speed of sound at isentropic conditions, and the characteristic equation sets are

$$\frac{dx}{dt} = u \pm a \tag{2.4.1}$$

and

$$\frac{dp}{du} = \mp C = \mp\frac{\kappa p}{a}. \tag{2.4.2}$$

These equations are often represented in the so-called position and state diagram respectively. The state of a point defined by the intersection of two characteristics in the position plane, is given by the corresponding intersection point in the state plane. From a boundary condition that models the relation between p and u, or a and u, the solution of the problem can be achieved by finding intersection points in the state diagram using the known slopes of the characteristics given by Eq. 2.4.2 and then the corresponding points in the position diagram. In MOC, the states, the velocities and pressures, of the fluid are resolved from the state diagram and the positions and time points where and when these states are achieved can be resolved from the position diagram.

One can note that $u + a$ is the speed of a pressure wave in the flow direction and $u - a$ the speed of a pressure wave in the opposite direction. The characteristics are thus corresponding to waves propagating information in the flow.

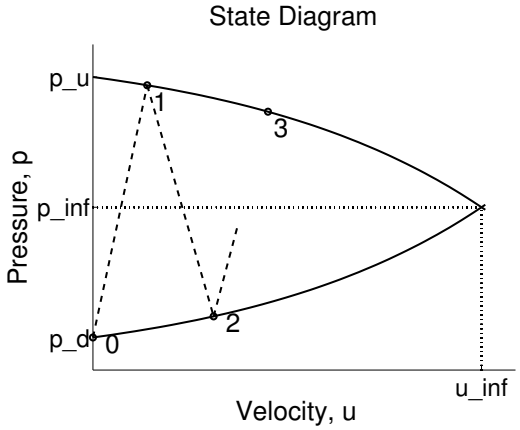

Figure 2.1: The state diagram with the boundary curves for the upstream and downstream boundary conditions.

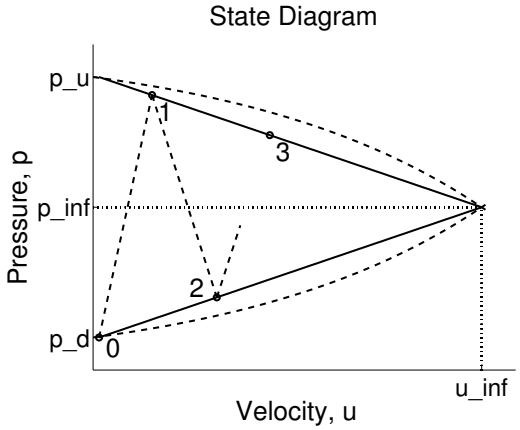

Figure 2.2: The state diagram with the boundary curves and the linearised boundary conditions for the upstream and downstream boundary conditions.

A dynamical model describing how the velocity strives towards the steady state is sought. A state diagram showing the boundary conditions is given as in [6] and [7] and is seen in Fig. 2.1. The pressure upstream of the pipe is denoted p_u and the pressure downstream is called p_d. As indicated in the figure the steady state values, p_∞ and u_∞, are given by the intersection of the boundary conditions. The boundary conditions are, as mentioned earlier, models of the relation between the velocity and pressure at the boundary. Modelled at the upstream end is the pressure drop, from p_u to the pressure inside the boundary, occuring at an accelleration to a certain velocity inside of the boundary. As seen in Fig. 2.1, the upstream boundary condition shows a pressure decreasing as the flow is accellerating. At the downstream end the model describes the pressure needed inside of the boundary to have an outflow at a certain velocity given the downstream pressure p_d, a higher velocity demands a higher pressure inside of the boundary. The steady state is reached when both these conditions are fulfilled. Note that this steady value holds within the pipe,

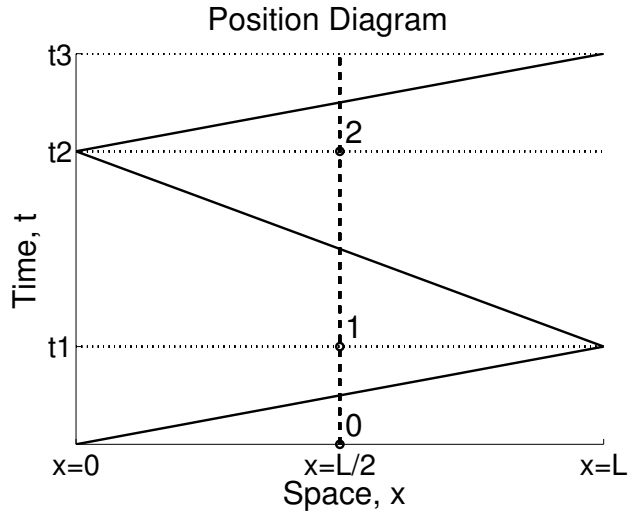

Figure 2.3: The position diagram corresponding to the characteristics in Fig.2.2. The time points for the states to arrive at the middle point are indicated.

outside each end of the pipe there is still a pressure difference driving the flow at a velocity u_∞ at steady state. Some states are indicated in the figure as points and the characteristic lines are shown as dashed lines, showing how the velocity state strives towards steady state. The slope of the characteristics are $\pm C = \pm\frac{\kappa p}{a}$, as stated in Eq. 2.4.2.

Given the steady values one can easily linearise the boundary conditions as seen in Fig. 2.2. The linearised boundary conditions are given by

$$p = p_u - \frac{p_u - p_\infty}{u_\infty} u = p_u - Au \tag{2.4.3}$$

and

$$p = p_d + \frac{p_\infty - p_d}{u_\infty} u = p_d + Bu, \tag{2.4.4}$$

where A and B are the linearised slopes of the upstream respectively downstream boundary conditions. From the different slopes in the linearised state diagram the succession of the velocity states can now be shown to be

$$u_{2j+2} = (1-\lambda)u_\infty + \lambda u_{2j} \tag{2.4.5}$$

and

$$u_{2j+1} = (1-\xi)u_\infty + \xi u_{2j}, \tag{2.4.6}$$

where $\lambda = \frac{(C-B)(C-A)}{(C+B)(C+A)}$ and $\xi = \frac{C-B}{C+A}$.

The position diagram corresponding to the state diagram Fig. 2.2 is seen in Fig. 2.3. In this case the system starts from a zero velocity state and a disturbance in pressure is occurring at $t = 0$ at the upstream end. The slopes of the characteristics in the position diagram are $u \pm a$ as stated in Eq. 2.4.1. The time points at which the middle point will achieve the states (0, 1, 2, ...) are indicated as dots, as in [6]. As seen this

is a simplification, assuming that the states are achieved when the wave has travelled from one end to the other the time points will be slightly shifted. At first the whole system has the initial states, $p = p_d$, $u = 0$. At time t_1 the information from the upstream boundary will reach the middle point, the state is now as indicated in Fig. 2.2, u_1 and p_1. Again, this is a simplification as the information actually reached the middle point earlier, at time t_1 the information has reached the downstream end at $x = L$. This value is kept until the conditions from the downstream boundary propagate to the middle point that achieves the state u_2, p_2, at time t_2. Assuming that the slopes of the characteristics, i.e. the propagation velocities from upstream and downstream is $c = a \pm u$ depending on the propagation direction of the wave, then the time points in Fig. 2.3 can be calculated as $t_j = j\frac{L}{c}$, where L is the length of the branch.

It can now be shown that for $\lambda > 0$ the even velocity states $(u_0, u_2, u_4, ...)$ are given by

$$u(t) = \left(1 - e^{-\frac{t}{\tau}}\right) u_\infty \tag{2.4.7}$$

and the odd velocity states are given by

$$u(t) = \left(1 - \frac{\xi}{\sqrt{\lambda}} e^{-\frac{t}{\tau}}\right) u_\infty, \tag{2.4.8}$$

with $\tau = -\frac{2L}{cln\lambda}$. These equations correspond to the dynamical model

$$\frac{du}{dt} = \frac{u_\infty - u}{\tau}. \tag{2.4.9}$$

If $\lambda < 0$ the even velocity states are given by

$$u(t) = \left(1 - e^{-\frac{t}{\tau}} cos(\omega t)\right) u_\infty \tag{2.4.10}$$

and the odd velocity states by

$$u(t) = \left(1 - \frac{\xi}{\sqrt{\lambda}} e^{-\frac{t}{\tau}} sin(\omega t)\right) u_\infty, \tag{2.4.11}$$

with $\tau = -\frac{2L}{cln|\lambda|}$ and $\omega = \frac{\pi c}{2L}$. The dynamical model is then

$$\frac{d^2u}{dt^2} = -\frac{2}{\tau}\frac{du}{dt} + \left(\frac{1}{\tau^2} + \omega^2\right)(u_\infty - u). \tag{2.4.12}$$

The sign of λ is shown in [7] to be dependent on the area ratios at the throats, that are the ratios between the throat areas and the pipe cross-section areas at each end. When there is a small opening the velocity will oscillate around the steady value before reaching it, with an angular frequency ω. From these dynamical models the velocity in the middle point of a branch can be calculated from the conditions in the capacities surrounding it. The solution procedure as described in [6] is to first define a time interval. Then, at each time step the velocities in each branch are calculated through either Eq. 2.4.9 or Eq. 2.4.12 depending on the value of λ, depending on

the boundary conditions at that time point. From these, the mass and energy flow rates can be calculated and used to evaluate the pressures and temperatures in the capacities through the mass and energy equations.

In Appendix A boundary condition equations are derived, as well as their linearisation, when possible, giving A, p_{u0}, B and p_{d0}.

2.4.2 Non-Homentropic Flow

In a non-homentropic case the entropy is no longer constant and both wall friction and heat transfer are present. One could probably assume homentropic flow in the inlet duct since the temperature does not affect the flow significantly, but the exhaust pipe flow is a totally different case where there are large temperature fluctuations that have a substantial impact on the flow. Therefore, it is important to derive a model considering the non-homentropic effects for usage in engine applications.

The idea of the method is still the same; for a specific pressure difference over a duct with certain geometry, the velocity will strive towards a steady value. If the boundary conditions are changing, the steady state will change and the velocity will follow.

In [2] the velocity and pressure states resulting from upstream respectively downstream conditions are denoted $u_{u,j}$ and $p_{u,j}$ for the upstream end and $u_{d,j}$ and $p_{d,j}$ for the downstream end. In [2] the boundary conditions for homentropic flow are still used, now linearised around the value of the velocity in the middle point, u^*, and relations for the variables are claimed to be

$$p_{u,j} = p_{u0} - Au_{u,j}, \tag{2.4.13}$$

$$p_{d,j} = p_{d0} + Bk_0u_{d,j}. \tag{2.4.14}$$

Here k_0 is the ratio between the steady entropy levels upstream and downstream of the pipe, that is equal to one when the flow is homentropic. However, since the motivation of these relations is not stated, boundary conditions are derived considering entropy changes and found in Appendix A, thus non-homentropic flow are used, thus the correction is not needed and one can set $k_0 = 1$.

Combining Eq. 2.4.13 and 2.4.14 with 2.3.8, integrated over the time between two wave reflections, one can eventually reach the expressions for the progression of the upstream and downstream velocities as

$$u_{u,i+1} = \frac{C-B}{C+B}\frac{C-A}{C+A}u_{u,i} + \frac{2}{C+B}\frac{C}{C+A}(p_{u0}-p_{d0}) - \frac{1}{C+A}\left(\alpha_1\frac{C-B}{C+B} - \alpha_2\right) \tag{2.4.15}$$

and

$$u_{d,i+1} = \frac{C-B}{C+B}\frac{C-A}{C+A}u_{d,i} + \frac{2}{C+B}\frac{C}{C+A}(p_{u0}-p_{d0}) - \frac{1}{C+B}\left(\alpha_1 - \alpha_2\frac{C-A}{C+A}\right), \tag{2.4.16}$$

with $\alpha_1 = \frac{\Delta_1+\Delta_2+\Delta_3}{a+u}CL$ and $\alpha_2 = -1\frac{-\Delta_1-\Delta_2+\Delta_3}{a-u}CL$.

From these expressions one can calculate the steady values for the upstream and downstream velocity states by setting $u_{(\cdot,i+1)} = u_{(\cdot,i)}$. In the homentropic case, when both α_1 and α_2 are equal to zero, the upstream and downstream velocities achieve the same steady value,

$$u_{int} = \frac{p_{u0} - p_{d0}}{A + B}, \tag{2.4.17}$$

which again is the intersection between the boundary conditions.

In the non-homentropic case the expressions for the steady values are found to be

$$u_{u,\infty} = u_{int} - \frac{C + B}{A + B}\frac{1}{2C}\left(\alpha_1\frac{C - B}{C + B} - \alpha_2\right) \tag{2.4.18}$$

and

$$u_{d,\infty} = u_{int} - \frac{C + A}{A + B}\frac{1}{2C}\left(\alpha_1 - \alpha_2\frac{C - A}{C + A}\right). \tag{2.4.19}$$

It can be seen that the non-homentropic steady values are the homentropic steady value minus some non-homentropic terms including the friction and heat transfer.

With

$$\lambda = \frac{C - B}{C + B}\frac{C - A}{C + A}, \tag{2.4.20}$$

as before, the two equations Eq. 2.4.15 and Eq. 2.4.16 can be written as

$$u_{u,i+1} = \lambda u_{u,j} + (1 - \lambda)u_{u,\infty} \tag{2.4.21}$$

and

$$u_{d,i+1} = \lambda u_{d,j} + (1 - \lambda)u_{d,\infty}. \tag{2.4.22}$$

Since the same model is valid for both the upstream and downstream velocity, it is assumed to be applicable also for the middle point velocity, i.e.

$$u_{j+1} = \lambda u_j + (1 - \lambda)u_\infty, \tag{2.4.23}$$

with $u_\infty = \frac{u_{u,\infty}+u_{d,\infty}}{2}$.

The dynamical models for the different cases, $\lambda > 0$ and $\lambda < 0$, are then the same as for the homentropic case, i.e. Eq. 2.4.9 and Eq. 2.4.12.

2.4.3 Static vs Stagnation BC

As previously stated, the derivation of boundary conditions can be found in Appendix A. These are the boundary conditions for so called stagnation, or total, pressure conditions in the inflow case and static conditions in the outflow case. As described in [12] the stagnation conditions are applicable for inflow from a large reservoir when there is a pressure drop over the boundary driving the flow into the

pipe. When the boundary is not bordering a reservoir, but is located inside a pipe there is no pressure drop over the boundary. When discretising a pipe, static conditions, i.e. constant pressure over the boundary, should be used at the inlet to avoid unphysical pressure drops decreasing the velocity. There are however some difficulties associated with this condition. Since the pressure is constant over the boundary, the slope of the boundary curve, A, is zero. If the outflow side is fully opened, B will also be close to zero. This combination is of course dangerous since both the steady value of the velocity and the time variable τ are then approaching infinity. This can be handled as done in [13], finding the dynamic model for the velocity like before but with constant boundary equations as

$$\dot{u} = \frac{(p_u - p_d)c}{CL}. \tag{2.4.24}$$

To include the non-homentropic effects, the friction is included, giving the dynamic model

$$\dot{u} = \frac{(p_u - p_d)c}{CL} - G. \tag{2.4.25}$$

Chapter 3

The Model

3.1 Implementation of the Model

The model is going to be implemented in Modelica, an object-oriented, equation-based language aiming for modelling of the dynamical behaviour of multi-domain systems. The model is simulated using Dymola that is a Modelica Simulation Environment. Introductions to Modelica and Dymola can be found e.g. in [14] and [15].

All the Modelica code can be found in Appendix D. In this section the important parts in the implementation will be discussed with parts of the code present.

The model to be implemented is based on the QPM for non-homentropic flow presented in the previous section and is a component corresponding to one branch. Models for the capacities already exist in Modelon's library, calculating the pressure and temperature using the governing equations. The inflow is considered to be a non-homentropic process due to the vortices that influences the energy in the flow. The fluid is flowing out of the pipe under the assumption that the geometry is changing smoothly and that there will not be as great influence of vortices that it will affect the entropy. Therefore, the flow is being considered to be homentropic in the outflow. When linearising the boundary conditions the linearisation is made around the state u that is the current velocity in the middle of the pipe.

The goal is that the model should be able to be simulated in real-time with a given time step. It should be able to handle geometry with different area ratios at the ends and a linear change in cross-section area over the pipe. Pressure values should be given at the ends and a possible heat transfer should be given either as an ambient temperature, using some kind of heat transfer model, or as an energy flow. The model will be implemented in Modelica and simulated with Dymola, and will be compatible with Modelon's Media library for ideal gases.

To help the solver and avoid e.g. denominators equal to zero, ϕ and ψ, that are the ratios between the downstream respectively upstream throat area and the pipe section area at each end, will be bounded between 0.01 and 0.999. The velocity will

be bounded from below to 0.1 m/s in some calculations. In the model the problem with a denominator equal to zero is generally handled by limiting the denominator from below by a suitably small value.

The Dynamic Model As described above the QPM is using MOC to get a dynamical expression describing the change of the velocity in the middle of the pipe. The dynamical model for the velocity is, depending on the value of λ

$$\frac{du}{dt} = \frac{u_\infty - u}{\tau}$$

or

$$\frac{d^2u}{dt^2} = -\frac{2}{\tau}\frac{du}{dt} + \left(\frac{1}{\tau^2} + \omega^2\right)(u_\infty - u)$$

for positive or negative values of λ respectively.

Since Modelica is an equation-based language, these dynamical equations can be implemented as they are, using a help variable to be able to express the second derivative since Modelica can handle first derivatives only, as seen below.

```
if (lambda > 0) then
    der(u) = (u_inf - u) / tau;
    der(u_temp) = -u_temp / tau;
   else
    der(u_temp) = -2/tau * der(u) + (1/tau^2 + omega^2)*(u_inf - u);
    der(u) = u_temp;
  end if;
```

Here the variable u is assumed to be known in every time step, and the value of its derivative is found using a numerical algorithm to update it. There are a number of different numerical algorithms available in Dymola, like Euler and Runge-Kutta, [16]. The sign of λ will depend on the area ratios as described in [7]. For an area ratio below a critical value the velocity will oscillate around the steady value before reaching it. Calculation of the help variable $u_{temp} = \frac{du}{dt}$ has to be performed at each step, even if the value of λ is positive and the second derivative is not needed. Identifying, when $\lambda > 0$, $\frac{du_{temp}}{dt} = \frac{d}{dt}\left(\frac{u_\infty - u}{\tau}\right) = -\frac{1}{\tau}\frac{du}{dt} = -\frac{1}{\tau}u_{temp}$, this can be used as an update of the help variable when $\lambda > 0$.

To make the code faster the function *noEvent* can be used as seen below.

```
  der(u) = noEvent(
  if lambda > 0 then
    (u_inf - u) / tau
  else
    u_temp);

  der(u_temp) = noEvent(
  if lambda > 0 then
```

```
    -u_temp/tau
  else
    -2/tau*der(u) + (1/tau^2+omega^2)*(u_inf-u));
```

An event occurs when a condition is activated and will then stop the simulation to find the exact time point at which the condition was achieved and then an initialisation problem will be solved from this point. This will make the simulation slow, so if the exact time at which the conditions are achieved is considered to be of less importance *noEvent* can be used even though this will give less accuracy, [17]. In the model there is a parameter called *useEventLambda*, that is *true* if one wants to handle the events and $false$ if one thinks it is enough to use *noEvent*. For an engine application the value of λ should vary frequently and using $useEventLambda = false$ may be necessary to be able to simulate in real-time, results can then be confirmed by simulating with $useEventLambda = true$.

As been discussed, there is a special case when the upstream boundary is static and the downstream end is fully opened. When this occurs the dynamic model is as given in Eq. 2.4.25 and this is included as seen in the final code below.

```
if p_u>p_d and static_upstream and phi==1 or
  p_d>p_u and static_downstream and psi==1 then
  der(u)=(p_u-p_d)*c/(C*L)-G;
  der(u_temp)=0;
else
    if useEventLambda then
      if (lambda > 0) then
          der(u) = (u_inf - u) / tau;
          der(u_temp) = -u_temp/tau;
      else
          der(u_temp)= -2/tau * der(u) +
          (1/tau^2 + omega^2)*(u_inf - u);
          der(u)=u_temp;
      end if;
     else
      der(u) = noEvent(
      if lambda > 0 then
          (u_inf - u) / tau
      else
          u_temp);
      der(u_temp) = noEvent(
      if lambda > 0 then
          -u_temp/tau
      else
          -2/tau*der(u) + (1/tau^2+omega^2)*(u_inf-u));
    end if;
end if;
```

There is two Boolean parameters *static_upstream* and *static_downstream* present. The first condition is considering non-reversed flow and the latter is considering reversed flow. The user is defining whether or not the boundary should be given static or total conditions. The upstream end is the end corresponding to an area

ratio called ψ and the downstream end corresponds to the end where the area ratio is called ϕ. As will be discussed below this is just notations, when the flow is reversed the downstream end will physically be upstream. When the flow is non-reversed, the upstream end is static and the downstream end is fully opened the dynamic model will be given as in Eq. 2.4.25. Corresponding conditions are present for the reversed flow case.

Calculation of c The values of τ and ω are calculated as previously derived as $\tau = -\frac{2L}{cln|\lambda|}$ and $\omega = \frac{\pi c}{2L}$. The question is how to calculate the value of c that is alternating between $a + u$ and $a - u$ depending on which direction the wave is travelling. The derivation of the dynamical equations uses the analysis of one wave to link pressure differences to the velocity in the middle of the pipe. Now, if one chooses to consider a wave starting at the upstream end and tracking the wave using the knowledge of the speed and the length of the pipe, it is possible to change the value of c when it reaches the end of the pipe.

```
der(x) = c;
xMod = noEvent(mod(x,2*L));
c = noEvent(
  if xMod > 0 and xMod < L then
      a_u+u
  else
      a_u-u);
```

Here a modulo operation is performed to find out where the wave is and where it is heading. Normally, the modulo operation triggers an event and this will, as been mentioned, slow down the simulation considerably. To have an event triggered in each time step is unthinkable if one aims for real-time simulations. For this reason *noEvent* is used when calculating the modulo of x. The dynamics of this kind of system is fast, the wave could travel back and forth in milliseconds. To be able to simulate in real-time one cannot afford the vast amount of events triggered for each time c is changed either. Therefore, *noEvent* is used also for this operation.

Calculation of λ The value of λ, also used in the expressions for τ and ω, is calculated as

```
tau = -2*L/(c * log(min(0.999,abs(lambda))));
omega = Modelica.Constants.pi * c /(2*L);

lambda = (C-abs(B))/(C+abs(B))*(C-abs(A))/(C+abs(A));
C = k_u*p_u/a_u;
```

where the properties of the media at the upstream end will be used. This is a simplification since the state of the fluid outside the pipe is considered, not taking into account the throat.

When the velocity is small A and B approaches zero and λ will be close to one. This will make the dynamics very slow since τ becomes large and thus $\frac{du}{dt}$ will become small. This is a correct behaviour as described by MOC, but when λ becomes equal to one τ will get a denominator value equal to zero, something that can not be handled in the simulation. Therefore, the value of λ is limited by 0.999.

The use of absolute values of A and B is due to the reversed flow case and will be discussed later.

Calculation of A, B, p_{u0} and p_{d0} using the Boundary Equations The values of A and B, as well as p_{u0} and p_{d0} used for example in the calculation of u_∞ are calculated, as described in Appendix A, through linearisation of the boundary equations around the velocity $u^* = u$, see Eq. A.0.1, Eq. A.0.2, Eq. A.0.3 and Eq. A.0.4. In the boundary equations the state of the fluid at the respective end is used, e.g. k_u and a_u are used in the boundary equations for the upstream end. As has been said before, the outflow is considered to be homentropic and the inflow non-homentropic. Both these boundary conditions are derived in Appendix A for subsonic and sonic flow.

In the case of subsonic outflow the boundary equation does not allow linearisation, since it is not possible to find an analytic expression $p = f(u)$, see Eq. A.2.4. There are two alternative approaches to this problem. The first is to, instead of linearising around u^* as in [2], instead use the linearisation proposed in [6] and [7], Eq. 2.4.3 and Eq. 2.4.4. To do this one must find the intersection of the boundary curve describing subsonic outflow and the boundary curve describing the inflow, depending on whether the inflow is subsonic or sonic. This will lead to non-linear systems of equations. Then $p_{d0} = p_d$ and $B = \frac{p_\infty - p_d}{u_\infty}$. In [2], this approach is used under the assumption that one of the ends is always fully opened, ending up in simple analytic expressions.

The second approach is to use the definition of the derivative and linearise around u^* using $B = \frac{p_1 - p_2}{2\Delta u}$. Here p_1 is the pressure solving the boundary equation for subsonic outflow with a velocity $u^* + \Delta u$ and p_2 the pressure solving the equation for $u^* - \Delta u$. If one lets Δu approach zero this would give the derivative of the function describing the relation between p and u at subsonic outflow around u^*. In this case it is probably enough to take some percentages of the value of u^* as the value of Δu. The value of p_{d0} can in this case be set according to Eq. A.0.3, where $f(u^*)$ is the pressure value corresponding to u^*, given by the boundary equation. The advantage of this approach is a more accurate slope of the boundary condition, since the linearisation is made around u^*. The drawback is that three non-linear systems will occur, one for p_1, one for p_2 and one for solving the pressure corresponding to u^*, called p_{subOut}. Nevertheless, this is the approach chosen in the model.

To investigate the behaviour of the boundary condition for subsonic outflow, the velocity is calculated and plotted for both positive and negative pressures to see all the possible solutions the solver could find for a specific u. As seen in Fig. 3.1 there is several solutions for many values of u. There is a great risk that the wrong solution

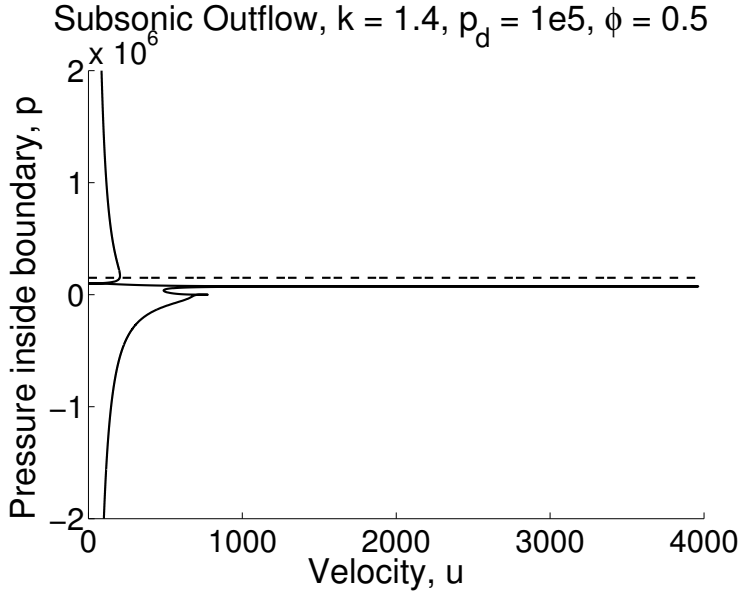

Figure 3.1: The relation between pressure and velocity at the outflow end for subsonic outflow. The downstream pressure is 1 bar. The dashed line shows the critical pressure.

will be found if using the expression for subsonic outflow as derived in Eq. A.2.4, especially for low velocities. One solution to this is to rewrite the expression not allowing the negative solution that is unphysical. Also the solution should not be found below p_d or above the critical pressure, p_s, presented below, at which the flow becomes sonic.

To do this a help variable p_{temp} is introduced, where $p = |p_{temp}| + p_d$. A boundary equation that secures the goals described above can now be found as

$$u = -sign(p_{temp})\frac{min(|p_{temp}|, p_s - p_d) - max(|p_{temp}|, p_s - p_d) + |p_{temp}| - (p_s - p_d)}{2max(10^{-8}, (max(|p_{temp}|, p_s - p_d) - min(|p_{temp}|, p_s - p_d)))} \cdot \sqrt{\left| \frac{2a_d^2}{k-1} \frac{max\left(1, \frac{|p_{temp}|+p_d}{p_d}\right)^{\frac{k-1}{k}-1}}{max\left(10^{-8}, \phi^2 max\left(1, \frac{|p_{temp}|+p_d}{p_d}\right)^{\frac{2}{k}-1}\right)} \right|} \quad (3.1.1a)$$

$$-sign(p_{temp})\frac{min(|p_{temp}|, p_s - p_d) - max(|p_{temp}|, p_s - p_d) - |p_{temp}| + (p_s - p_d)}{2max(10^{-4}, (max(|p_{temp}|, p_s - p_d) - min(|p_{temp}|, p_s - p_d)))} \cdot a_d \left(\frac{|p_{temp}| + p_d}{p_d}\right)^{\frac{k-1}{2k}} \psi p_{cr}^{-\frac{k+1}{k-1}}. \quad (3.1.1b)$$

Here the expression 3.1.1a is a slightly modified version of the subsonic outflow boundary condition and 3.1.1b is the expression for sonic outflow. The rest of the expression ensures that the subsonic expressions are used for pressure values below the critical pressure and the sonic expressions are used for pressure values above the critical pressure. Even if the solution to this expression is not going to be used

when the pressure is exceeding the critical pressure, one wants to have a smooth curve and reasonable values to start from in the next step. The reason for having a $max(10^{-8}, ...)$ in one of the denominators and $max(10^{-4}, ...)$ in the other is that Dymola will create a residual function which it will try to minimise. If the two denominators have the exact same expression Dymola will manipulate the residual function that will be "simplified" by being multiplied by the denominator, that is, $u \cdot denominator - numerator = 0$. In this case there is a solution that holds for all values of u; $|p_{temp}| = p_s - p_d$, when both the denominator and the numerator are zero.

Calculating and plotting the velocity against p_{temp} and against p, Fig. 3.2 shows that the value of p will follow the subsonic boundary condition while below p_s and then follow the sonic condition.

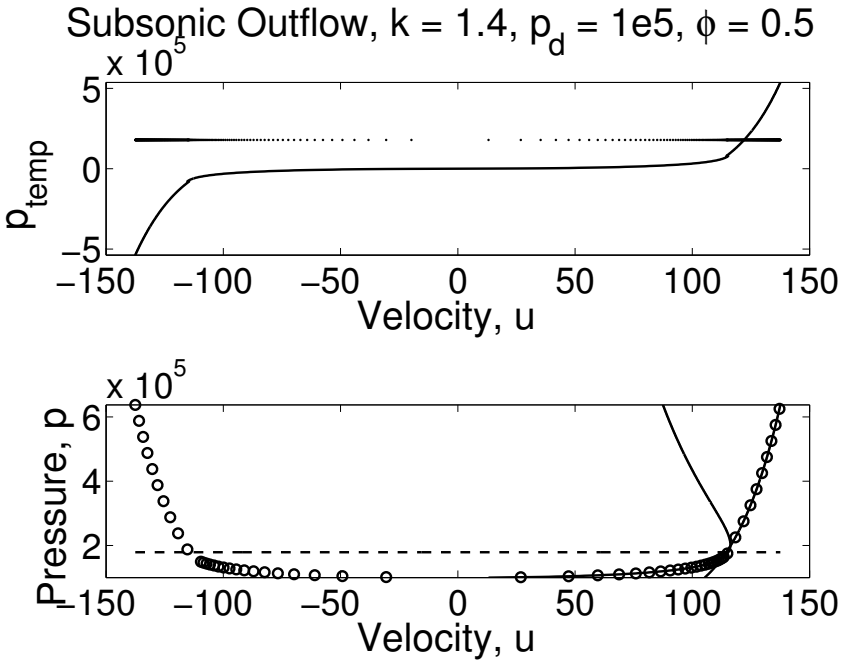

Figure 3.2: The upper figure shows the value of u for different values of p_{temp}. The real pressure, p, is shown in the lower figure as circles. The solid lines show the boundary condition for subsonic and sonic outflow. The horizontal dashed line shows the value of p_s, the critical pressure.

Critical Pressure In Appendix A derivations of the critical pressures are made for inflow and outflow through a partially open boundary. If the pressure difference over a throat is large enough the flow will become sonic in the throat and this will define how the velocity and pressure is related to each other through different boundary equations. Whether or not the flow becomes sonic over the boundary is determined by the critical pressure. At inflow the critical pressure is calculated using the pressure inside of the boundary and it is then compared with the upstream pressure. If the upstream pressure is larger than the critical pressure then the pressure difference over

the boundary is large enough to make the flow sonic. For outflow the pressure inside of the boundary is compared to a critical pressure calculated using the downstream pressure. At both boundaries, what equation to be used is decided depending on whether the flow is subsonic or not. Thus depending on the pressure inside the boundaries the boundary equations will differ. At the same time, the pressure inside the boundary also depends on whether or not the flow is sonic. That is, knowing the velocity and what boundary equation to be used, the pressure inside a boundary can be calculated, but to know what equation to be used the value of the pressure inside of the boundary is needed. If one assumes that it is enough to use the value of the pressure inside of the boundary from the previous time step one breaks up the system of equations for calculating the pressure inside the boundary at the current time. Here the function *delay* is used to get the value from the previous time step. When translating the model one must use the Dymola flag $Advanced.BreakDelayLoops = true$ to ensure that the previous value are assumed to be known in the current step. If one does not use this option the model will have six non-linear systems of equations to solve: of size two - one system of equation for the temperatures upstream and downstream of the pipe; and of size one - one each for p_1, p_2, p_{subOut} and the pressures inside the boundaries. The two last non-linear systems of equations, for the pressures inside the boundaries, are avoided if one uses $Advanced.BreakDelayLoops = true$, which could be favourable.

Like the boundary equation for subsonic outflow, the computation of the critical pressure for subsonic outflow has to be handled with care. In this case the critical pressure is $p_s = p_d \left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr}^{\frac{2\kappa}{\kappa-1}}$, where $\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr}$ fulfills Eq. A.2.7. In [2] an alternative expression is stated and this will be used since Eq. A.2.7 does not give a unique solution. The approximate expression stated in [2],

$$\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr} = 1 + \frac{k_d - 1}{4}\sqrt{1-\phi^2}, \tag{3.1.2}$$

and the exact expression, Eq. A.2.7, are shown Fig. 3.3. This shows that Eq. 3.1.2 gives a good approximation for most area ratios, ϕ (in reality $0 \leq \phi \leq 1$). The critical pressure has to be larger than the downstream pressure, but there is a chance that the solver, if one uses the exact expression, will give a critical pressure below the downstream pressure when searching a solution corresponding to a certain ϕ. Using Eq. 3.1.2 this cannot happen and one avoids the non-linear system of equations otherwise needed to be solved, that is, Eq. A.2.7.

Reversed Flow The model should be able to handle reversed flow occurring when $p_d > p_u$. The variables A and p_{u0} will still correspond to the same end, the end where the area ratio is denoted ψ, called the "upstream" end corresponding to e.g. p_u, even though it is not actually the upstream end anymore. That is however simply a matter of notations. The same holds for B and p_{d0}, these corresponds to the end where the area ratio is denoted ϕ.

To handle the reversed flow a new coordinate system is considered, namely the mirrored coordinate system. The velocity in the old coordinate system is negative

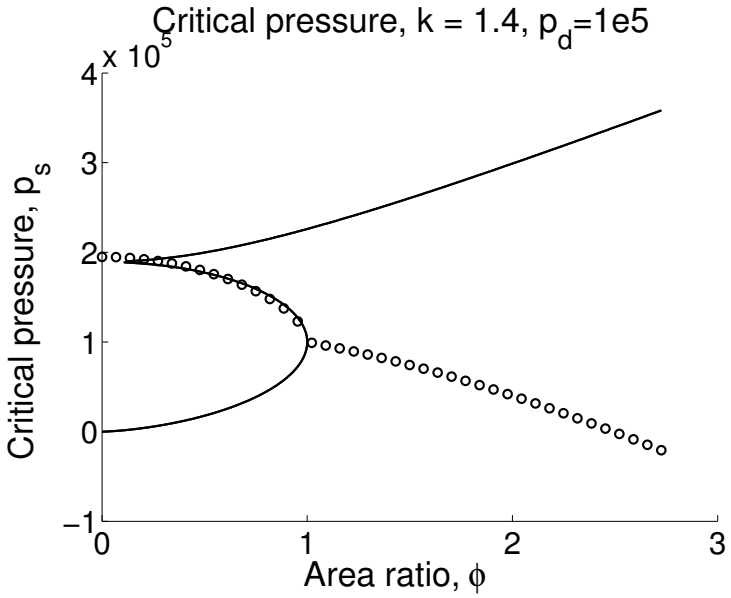

Figure 3.3: The critical pressure for different area ratios. The circles shows the approximated expression, Eq. 3.1.2 and the line the exact expression.

since the flow is opposing the positive direction of the system. In the new, mirrored system the velocity is positive with the same magnitude as in the old and here the same boundary equations can be used for outflow and inflow as before. Now, the steady value, u_∞, is searched for in this new coordinate system. It is clear that the steady value in the old system is then $-1 \cdot u_\infty$. The values of A and p_{u0} are calculated using the boundary equations for outflow in the new coordinate system and the values of B and p_{d0} using the boundary equations for inflow. The equations is thus the same, it is enough to simply switch p_u to p_d and so on. All variables used to calculate the steady value will use the absolute value of u, as well as the absolute values of A and B since they are assumed to be positive in the derivation. The value of u_{int} also needs to be corrected with a minus sign for reversed flow. Then $-1 \cdot u_\infty$ will be used in the dynamical model to calculate u, now in the old coordinate system.

Mass Flow Rate Since the model is lumping the whole pipe into one middle point velocity, the mass flow is also lumped and calculated as seen below.

```
m_flow_middle= u *(F+0.5*L*dFdx)*(rho_d+rho_u)/2;
port_d.m_flow = - m_flow_middle;
port_u.m_flow = m_flow_middle;
```

The mean of the densities are used as a middle point density. The mass flows in and out of a pipe can momentarily differ in a dynamic flow case. Here, the mass conservation is however assumed to be fulfilled over the pipe.

Numerical Issues When calculating the values of A, p_{u0}, B and p_{d0} the function *spliceFunction* from Modelon's library is used. This function enables a smooth conversion between the expressions for subsonic and sonic flow to be used. This will give better numerical properties of the model through avoiding large gradients.

There are a number of modifications used in the model, and it is important to be aware of where they are. There is for example the parameter *dp* used in *spliceFunction* defining the interval over which a smoothing function will be used. The value of *dp* has to be large enough to ensure a smooth continuous transition between the two expressions, but small enough not to affect the result. Another example is the variable Δu that has to be limited not to be too small, i.e. there is a limiting value that has to be of appropriate size. A change of the limiting value should not have a great impact on the solution. The same holds for the limiting values used to avoid zero-valued denominators. All these modifications are used to help the solver through good numerical properties, but it is important that they do not affect the solution. If the results of the model prove to diverge a lot from the real solution these values could be altered to ensure that the problem does not originate from them.

When calculating the variables connected to the boundary conditions there is an if-statement for $p_u > p_d$. When the flow is reversed the expressions are changed and events are created. This should not be a problem since there should not be a lot of reversed flow cases in the combustion engine application. In a case where reversed flow is common, maybe a trade-off between accuracy and real-time is needed, using *noEvent*.

3.2 Preliminary Tests

Some tests have been simulated to ensure that the model can handle different configurations of geometry and boundary conditions, etc. This is a first check to see that it looks reasonable, but comparison with other models and experiments is important to establish the reliability of the model. The configurations do not necessarily reflect an engine application, but will hopefully give the reader an insight into how different factors affect the flow and the model.

The tests are simulated using the explicit Euler method that uses a fixed time step, something that is necessary for real-time simulations. The CPU-time for integration given by Dymola will be noted to investigate the possibility to use the model in real-time simulations. A CPU-time lower than the simulated time will not guarantee that the model can be simulated in real-time but can give an idea of the possibility. The CPU-time should not be larger than the real time simulated.

The tests use, unless otherwise announced, a pipe without area changes inside the pipe. The length of the pipe is set to 1 m, the diameter to 5 cm and the friction factor to 0.05. The media used from Modelon's library is the *Fast Air, fixed composition,*

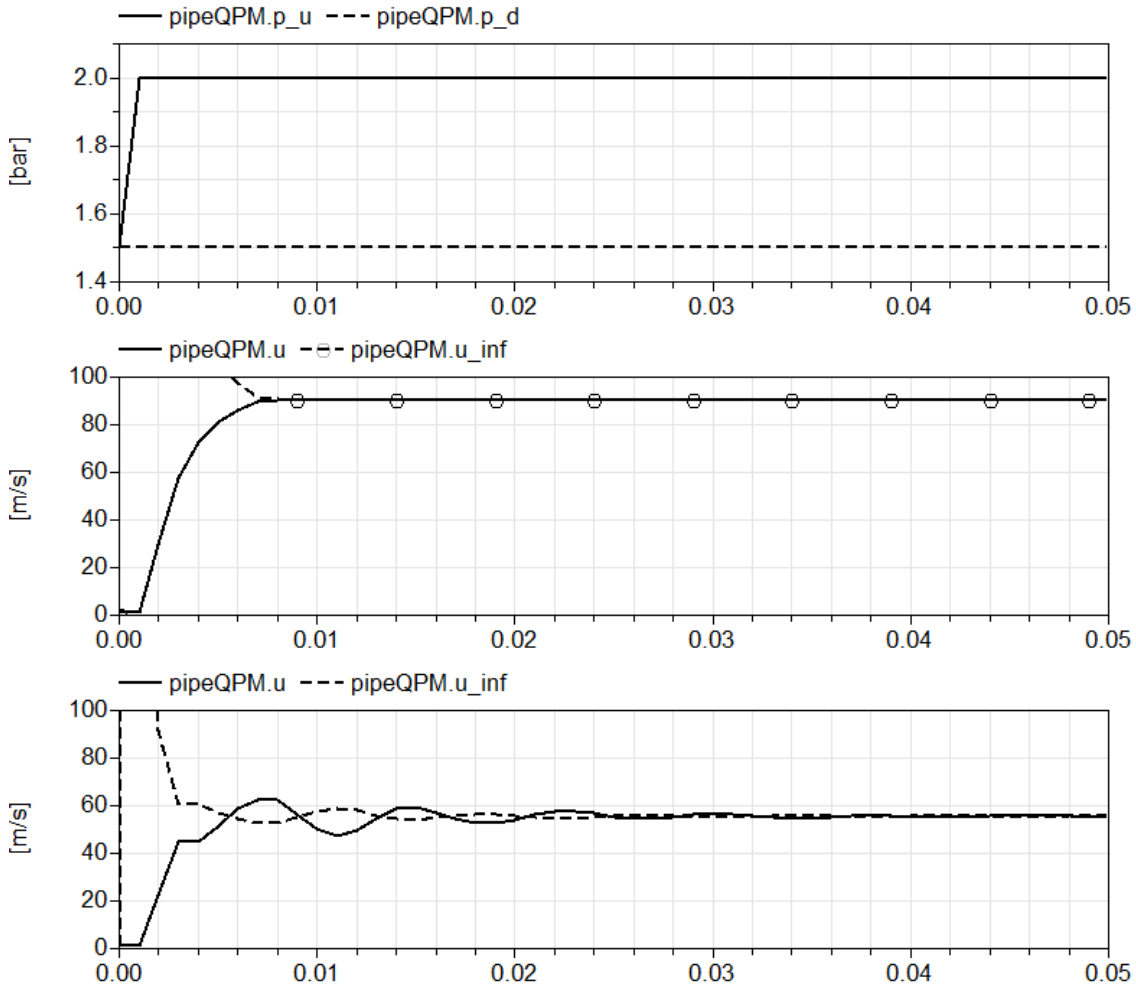

Figure 3.4: The upper figure shows the pressure upstream and downstream of the pipe. The middle figure shows the velocity together with its steady value for $\psi = 0.7$ and the lower for $\psi = 0.3$.

linear cp, 20-600 °C, an ideal gas. The temperature is 25 °C in both ends and there is no heat-transfer in general. The time step is in most cases 1 ms. A full presentation of configurations in the non-discretised cases are given in Table C.1 in the Appendix C together with the CPU-time for simulating each case.

Positive and Negative λ The first test uses a fixed geometry and a constant downstream pressure. The upstream pressure is changing as a ramp. Two different configurations of the geometry are simulated to confirm the different dynamical models for different signs of λ. In [7] a critical area ratio is calculated, below which the area ratio will give rise to oscillations as the value of λ becomes less than zero. The first configuration uses $\phi = \psi = 0.7$ and the second $\phi = 0.7$ and $\psi = 0.3$. The results are shown in Fig. 3.4.

For the case where $\psi = 0.3$ the value of λ will be negative and the oscillating nature is clearly shown in Fig. 3.4.

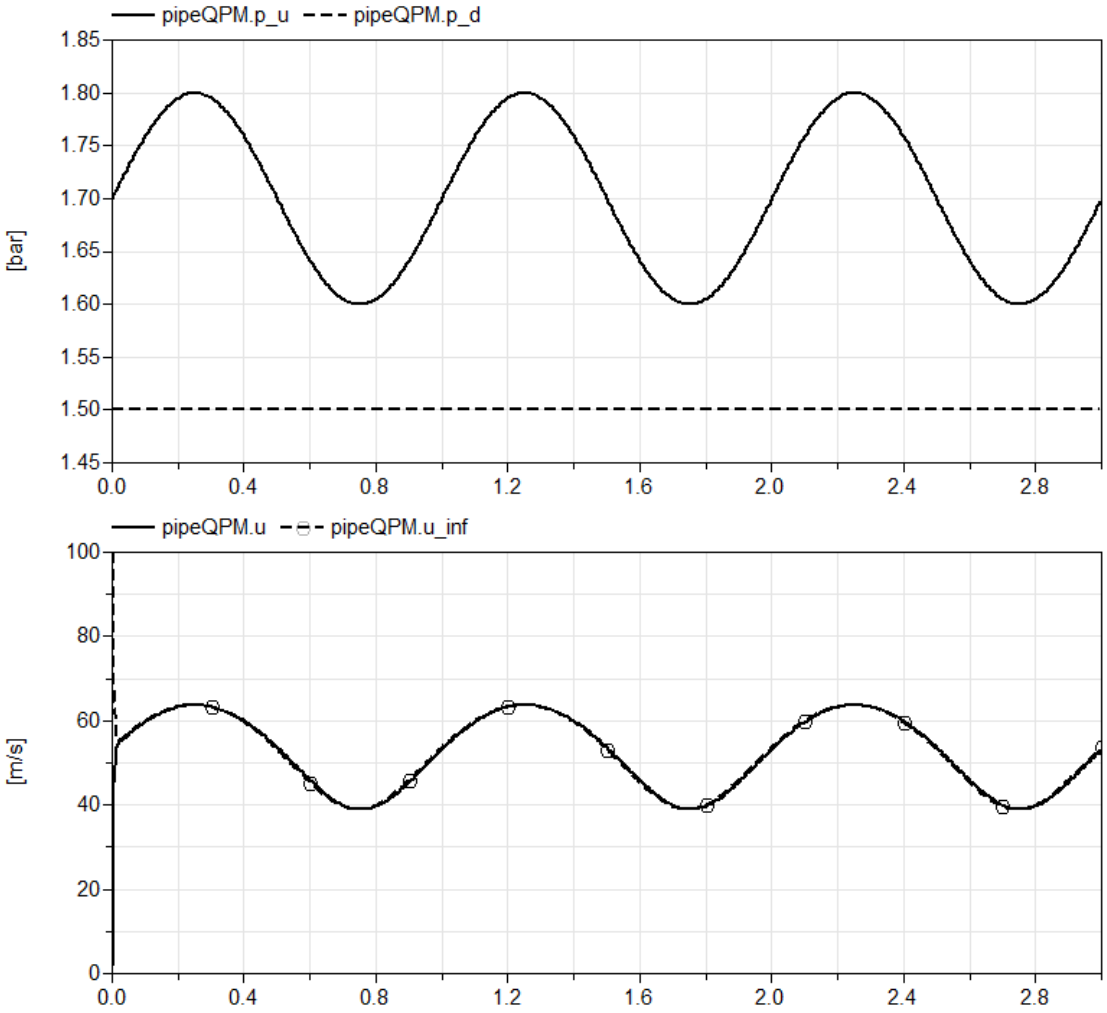

Figure 3.5: The upper figure shows the pressure upstream and downstream of the pipe. The lower figure shows the velocity in the middle of the pipe and the steady value of the velocity. The frequency of the upstream pressure is 1 Hz.

Changing Upstream Pressure Now both ϕ and ψ are set to 0.7. The downstream pressure is constant, and the higher upstream pressure is changing as a sinus function. Two configurations are simulated with frequency of the sinus function set to 1 Hz respectively 10 Hz. The first results are shown in Fig. 3.5 and the latter in Fig. 3.6.

A first notation is that the velocity is changing in accordance to the changes in the upstream pressure. When the pressure changes, the boundary condition will change and this will make the steady value, towards which the velocity strives, change. One can note that in the first case with lower frequency the velocity is almost the same as the steady value. In the case of higher frequency the inertia will prevent the velocity to reach the steady value before this is changing. This can be seen in Fig. 3.6, in the lower picture one can clearly see how the velocity strives towards the steady value but the changes are too fast for it to actually reach it.

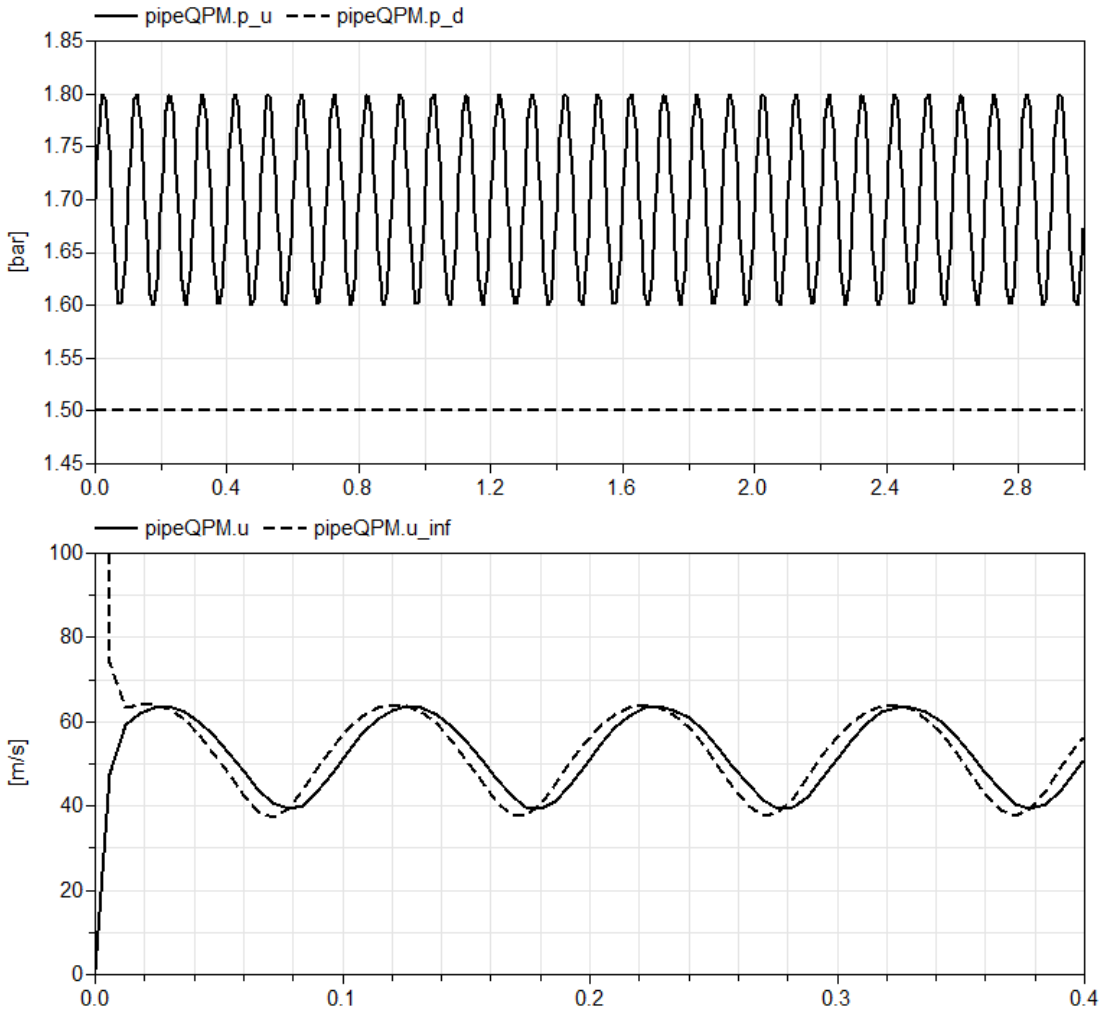

Figure 3.6: The upper figure shows the pressure upstream and downstream of the pipe. The lower figure shows the velocity in the middle of the pipe and the steady value of the velocity. The frequency of the upstream pressure is 10Hz. Note the different time scales.

Changing Area Ratio It has already been shown that the values of the area ratios have an impact on the dynamical behaviour of the velocity. Now, the pressure upstream and downstream, as well as the upstream area ratio ψ, are kept constant and the downstream area ratio, ϕ is changing between 0 and 1. As said before, the values of the area ratios actually used are limited between 0.01 and 0.99. The simulation proves to be unstable for small values of ϕ. As the area ratio approaches zero, that is, a fully closed end, the solution is oscillating and in some cases the simulation breaks down. The solution is to decrease the time step or change to a more stable method e.g. Runge-Kutta. The results can be seen in Fig. 3.7.

The velocity is oscillating when ϕ is close to zero due to natural oscillations since the area ratio is below a critical area ratio, making the value of λ negative. It is well known that the sampling frequency has to be at least twice the highest system frequency for a stable simulation. The small area ratio may give rise to dynamics of higher frequencies, thus demanding the smaller step size. The simulation time is increased with the decrease of step size, but there is still indications that the models could be simulated in real-time.

The same behaviour holds when changing ψ instead of ϕ in the same manner.

Reversed Flow Starting with an easy case where the upstream pressure is below the downstream pressure at all points in time, the resulting velocity is shown in Fig. 3.8. Here ϕ, ψ and p_d are held constant and the upstream pressure (which is only a notation, since the "upstream end" is now physically downstream), p_u is varying as a sine with a frequency of 1 Hz. For comparison the opposite case is simulated with the same pressure difference but a higher upstream than downstream pressure.

The result seems reasonable, the velocity becomes negative since the flow is reversed due to the higher pressure at the downstream end. When the pressure change decreases, so does the velocity. Once again the effects of the inertia is visible, the steady value is zero when the upstream pressure is equal to the downstream pressure, but the mid-point velocity does not have time to actually reach that value. One can see that the velocities in the two cases are essentially just mirrored around zero. There is a slight change in magnitude though, that originates from the magnitude of the pressure. Even if the pressure differences are the same, the magnitude of the pressure is lower in the reversed flow case, affecting the state of the fluid. The result is physically plausible since the geometry is symmetric.

Next step is to simulate a case where the upstream pressure is varying between larger and smaller values than the downstream pressure. This case is shown in Fig. 3.9. The upstream pressure is oscillating around the downstream pressure with a frequency of 1 Hz.

One can see that the manner of the velocity changes is plausible. The velocity is varying between a positive and a negative value of the same magnitude, which is reasonable since the pressure difference is the same only directed differently. The

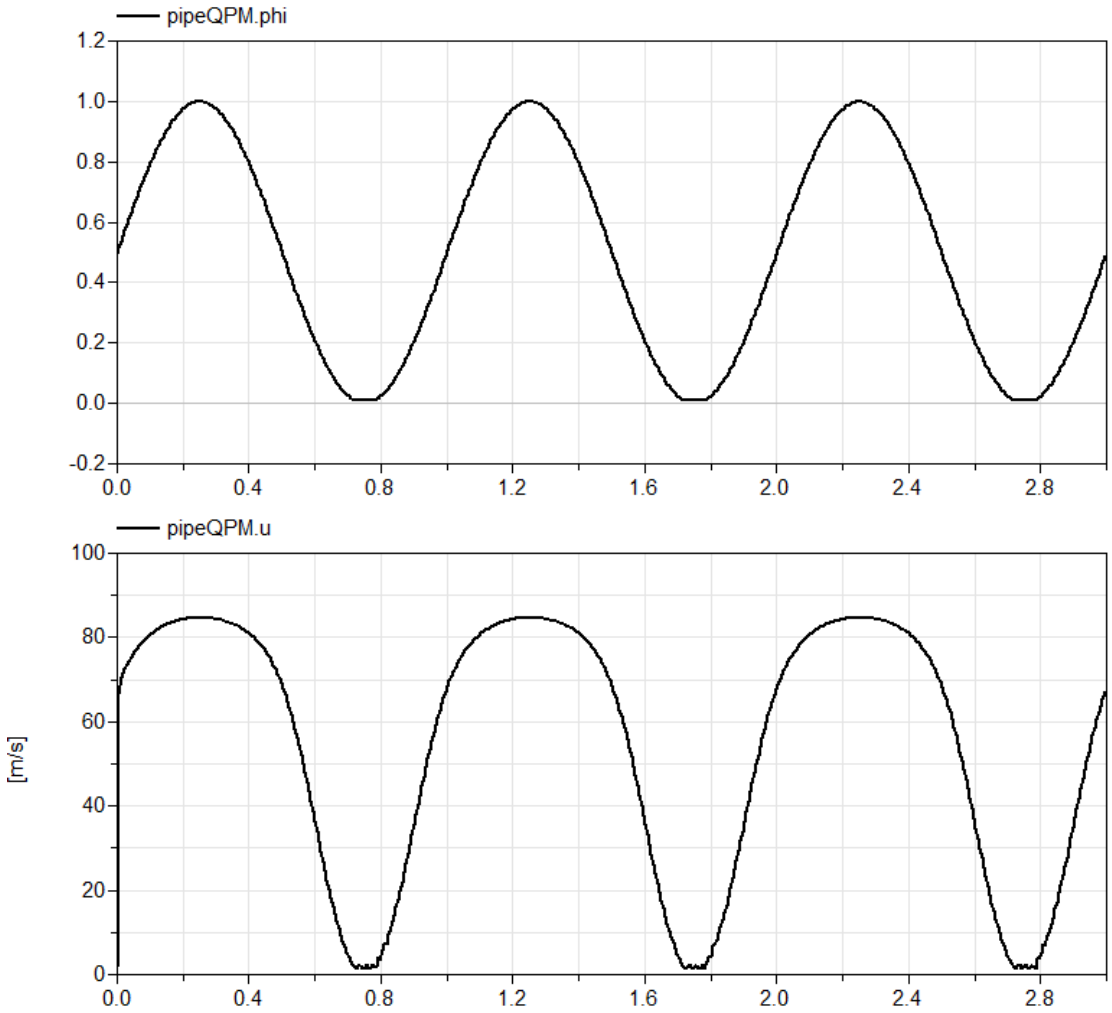

Figure 3.7: The upper figure shows the downstream area ratio. The lower figure shows the midpoint velocity.

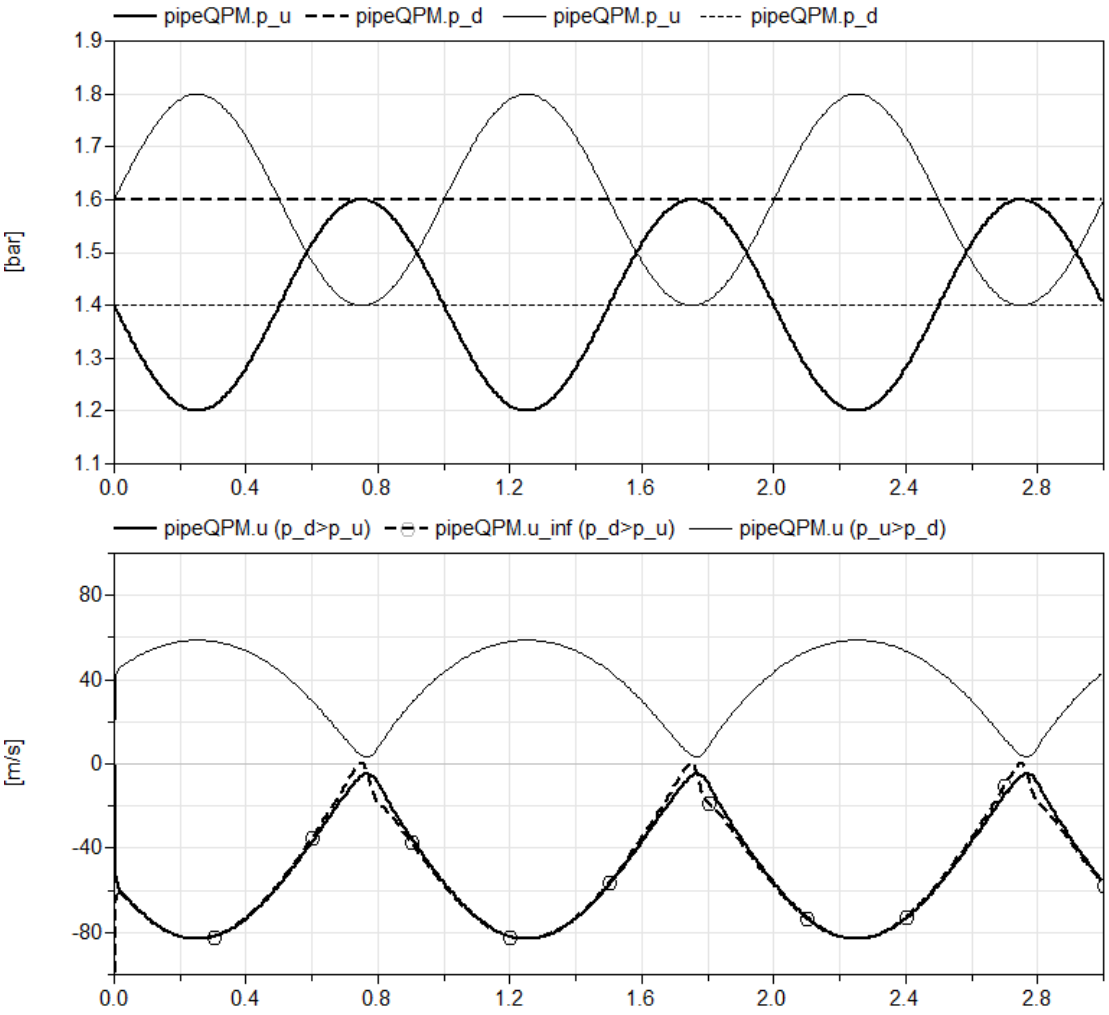

Figure 3.8: The upper figure shows the upstream and downstream pressure for the two cases. The case with higher upstream than downstream pressure is shown as thinner lines. The lower figure shows the midpoint velocity and its steady value for higher downstream pressure as thick solid and dashed lines.

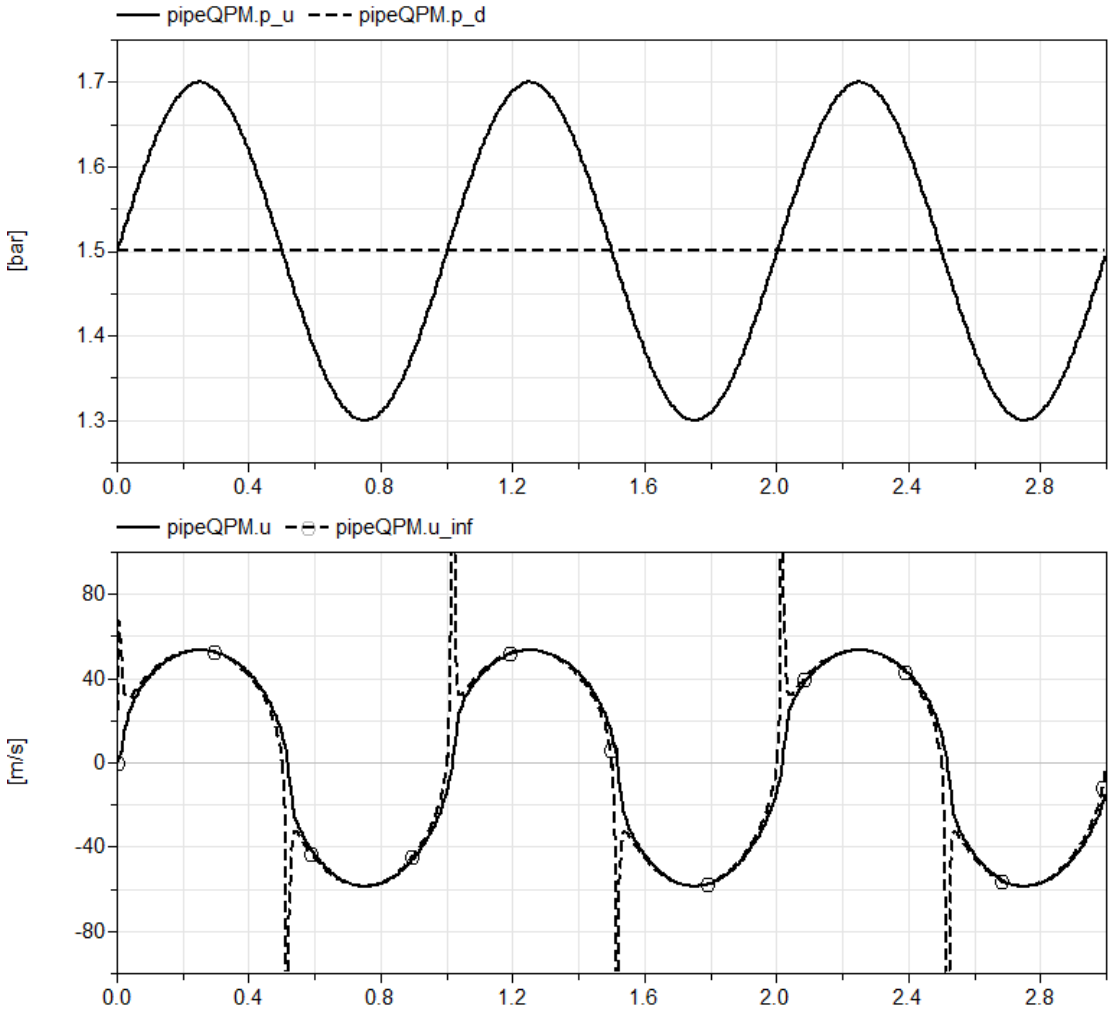

Figure 3.9: The upper figure shows the upstream and downstream pressure. The lower picture shows the velocity and its steady value.

steady value of u is having peaks when u becomes small. This is due to the behaviour of the boundary equations for small velocities, where they are close to constant, A and B that are the slopes of the boundary conditions approaches zero as u becomes small. Since the steady value can somewhat be seen as the intersection between the two boundary conditions, it will obviously become large as the boundary conditions approaches a constant behaviour. The peaks will drive the velocity away from the zero value.

The same test is made using a higher frequency and the result can be seen in Fig. 3.10. As before one can see the inertia effects more clearly for the faster dynamics, u is a bit shifted from the steady value.

Larger Friction Factor In the cases above the friction factor, f, has been given the value 0.05. Fig. 3.11 shows the results for the configuration used in Fig. 3.5, simulated with $f = 0.05$ and $f = 0.5$. One can see that the shape of the velocity

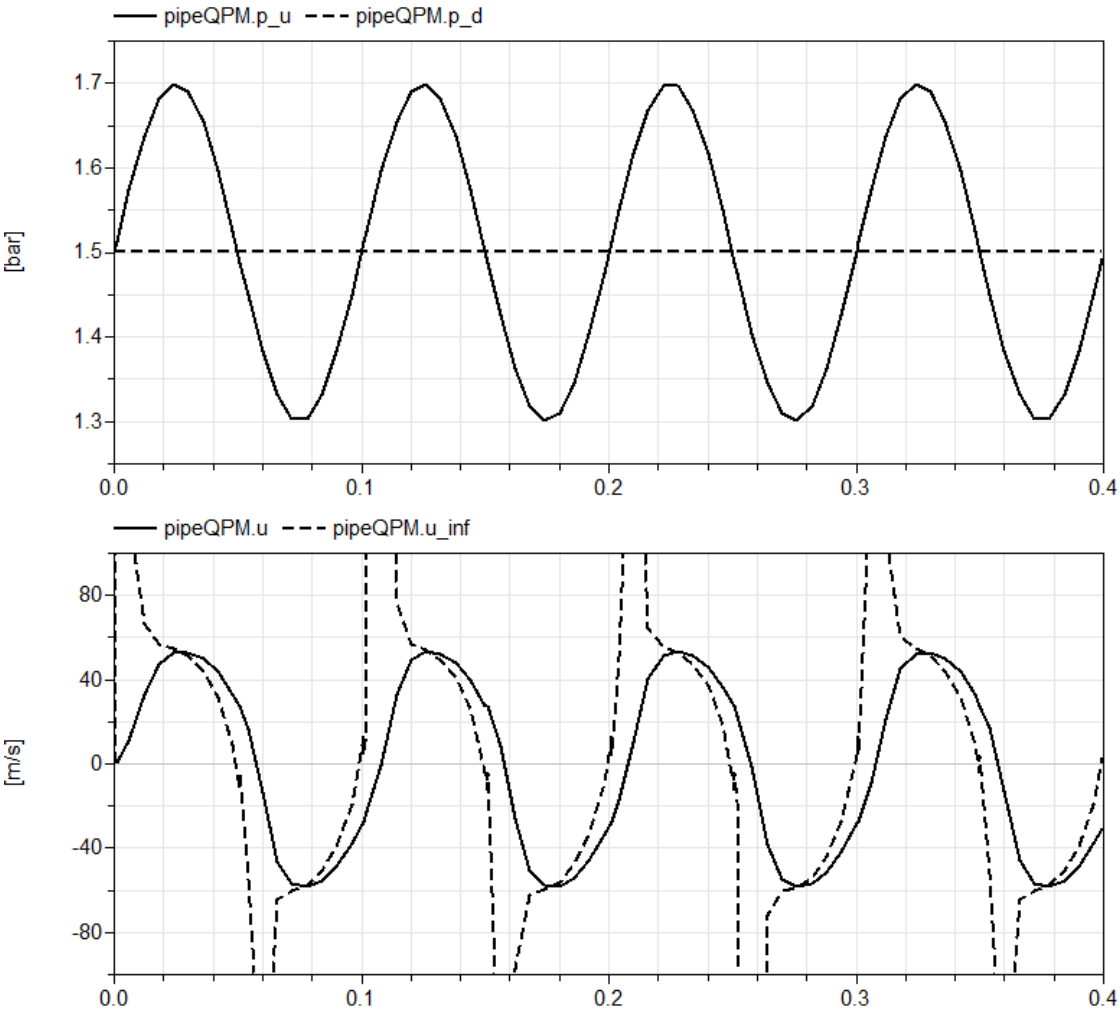

Figure 3.10: The upper figure shows the upstream and downstream pressure. The lower figure shows the velocity and its steady value.

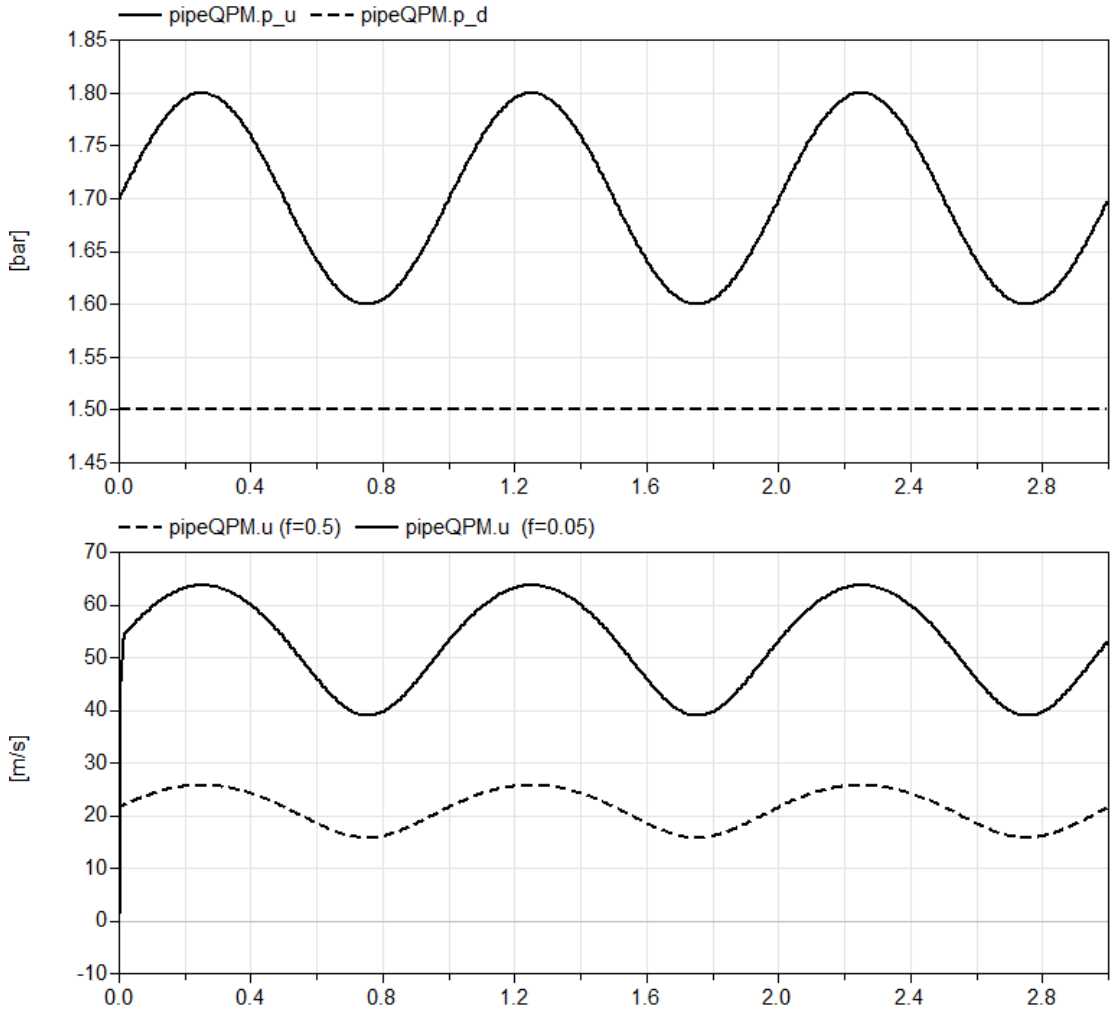

Figure 3.11: The upper figure shows the upstream and downstream pressure. The lower picture shows the velocity for $f = 0.05$ as the dashed line and $f = 0.5$ as the solid line.

is the same but the magnitude and amplitude is lower, which seems correct since a higher friction factor implies higher friction, thus more energy is lost.

Heat Transfer Using a heat transfer coefficient equal to 500, with a temperature of 25 °C at both ends of the pipe and an ambient temperature at 100 °C the velocity is barely changing as can be seen in Fig. 3.12. The mass flow is changing a bit more due to a change of the state of the fluid, the density downstream will change. The setup is again the same as in Fig. 3.5.

A Complex Configuration As a last test, using only one pipe segment, both the upstream and downstream pressures and the area ratios are changed with time. One can see in Fig 3.13 how the velocity gets a more complex behaviour. However, it is impossible to know if this is a correct behaviour. To be able to establish if this

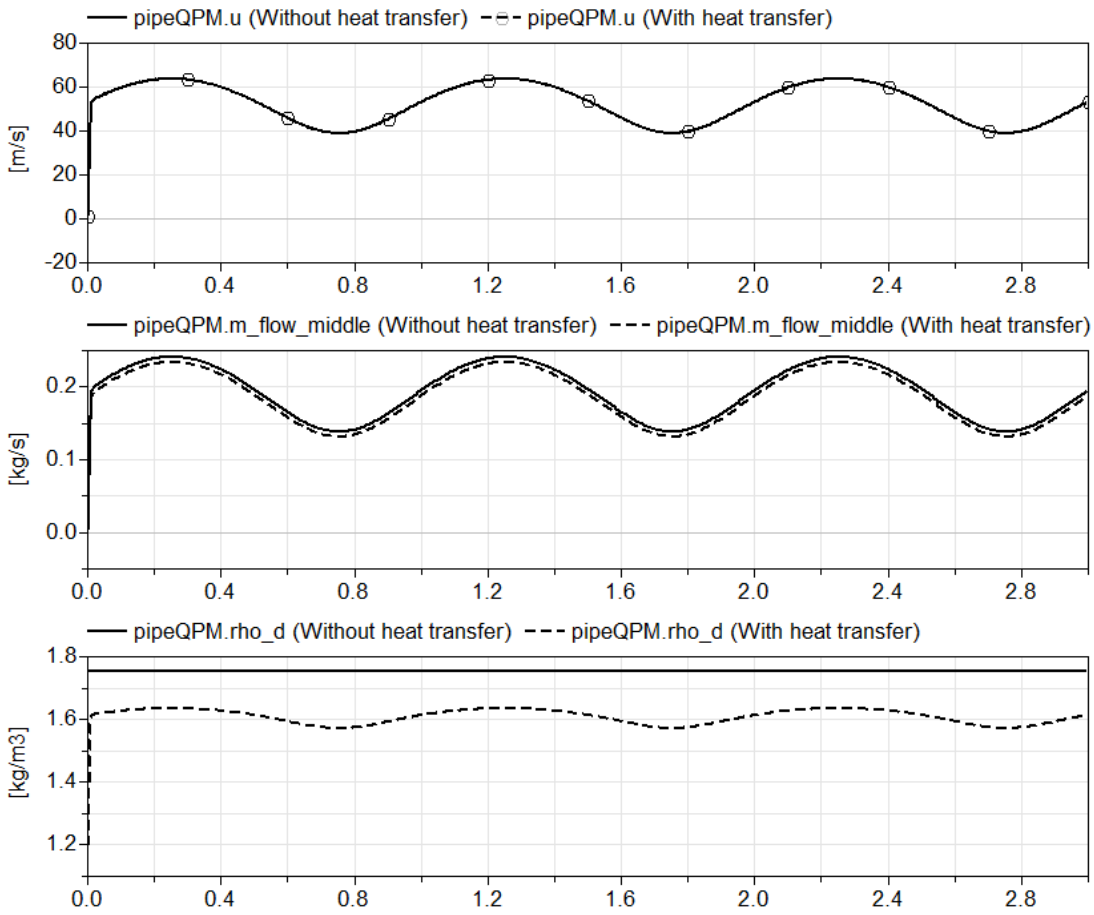

Figure 3.12: The upper figure shows the velocity with and without heat transfer. The middle figure shows the outflow in the two cases. The lower figure shows how the density is changing downstream when including heat transfer.

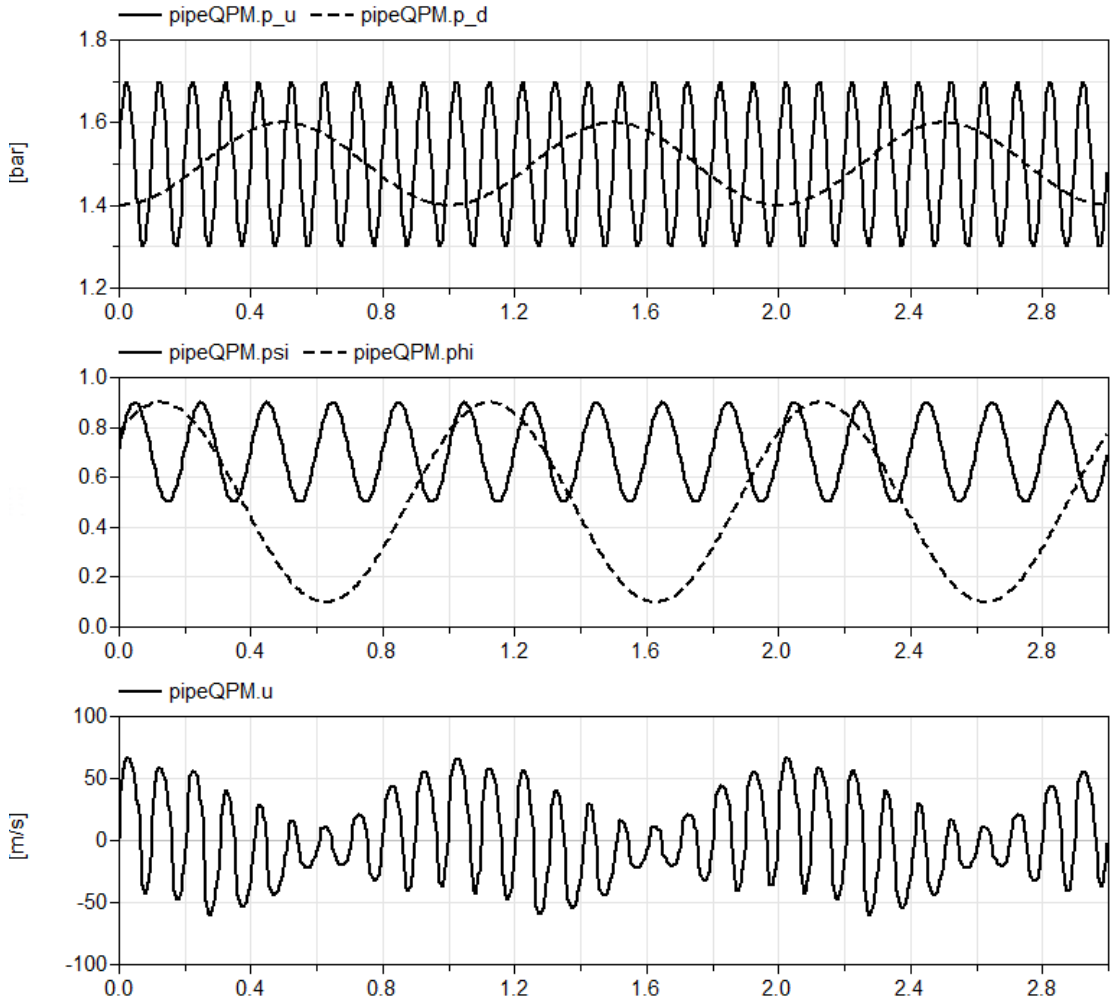

Figure 3.13: The upper figure shows the upstream and downstream pressure. The middle figure shows the area ratios. The lower picture shows the middle point velocity.

is the correct velocity profile one should compare with other models and especially with experiments.

Multiple Branches To increase the resolution of the pressure waves in the flow, one can use several branches with capacities in-between. In Fig 3.14 the mass flow out of the pipe is calculated using only one segment and then using two, three and five segments. The result is shown when using only total boundary conditions and when using static. Using several segments demands a shorter time-step due to shorter lengths, increasing the simulation time. Additional non-linear systems of equations will also increase the simulation time and it is doubtful that this could be simulated in real-time, even when using a more stable solver like Runge-Kutta.

One can see that the oscillating nature is resolved when using several pipe segments. When only using one pipe section these oscillations are smoothed out since the

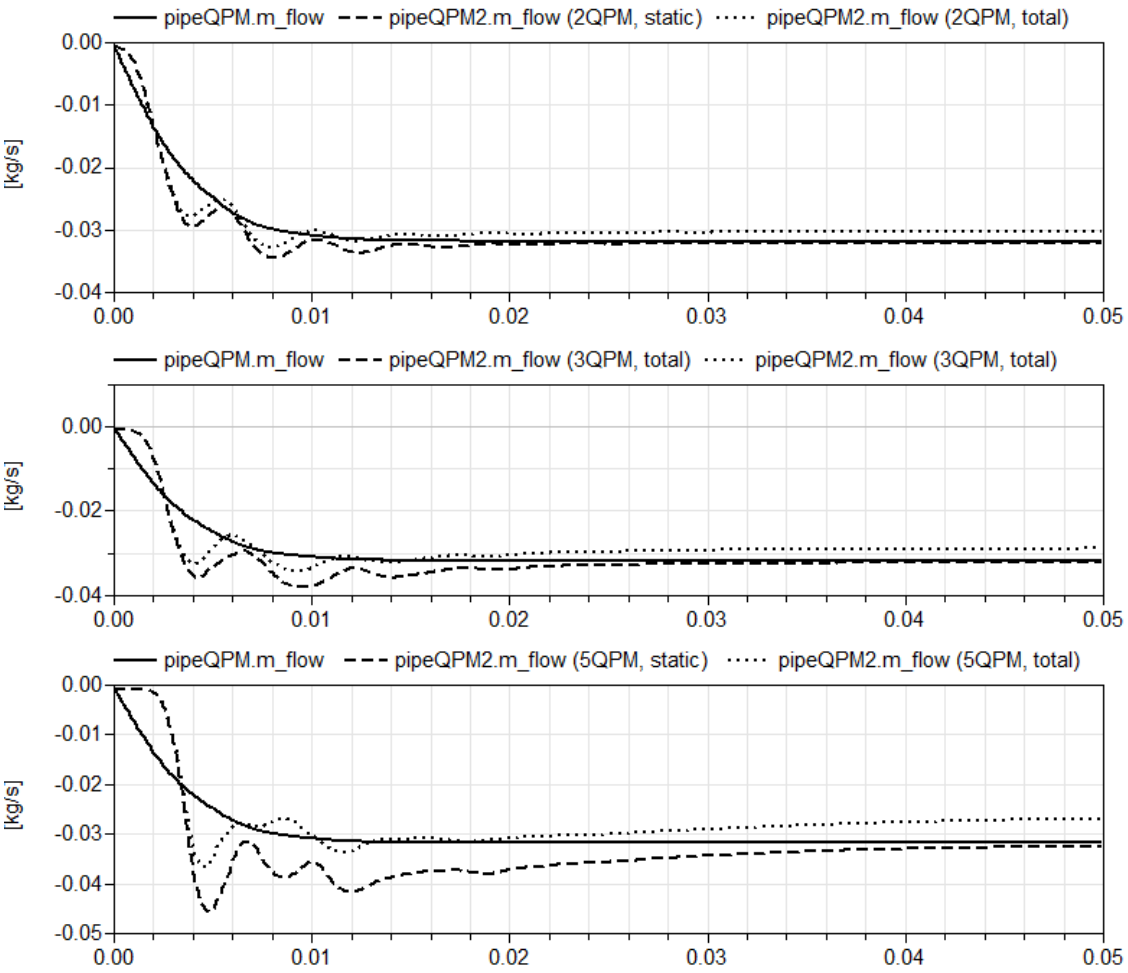

Figure 3.14: In all figures the mass outflow is shown using one QPM as a solid line. In the upper figure the result using two QPM pipes are shown using static boundary conditions, as a dashed line, and using total boundary conditions, as a dotted line.

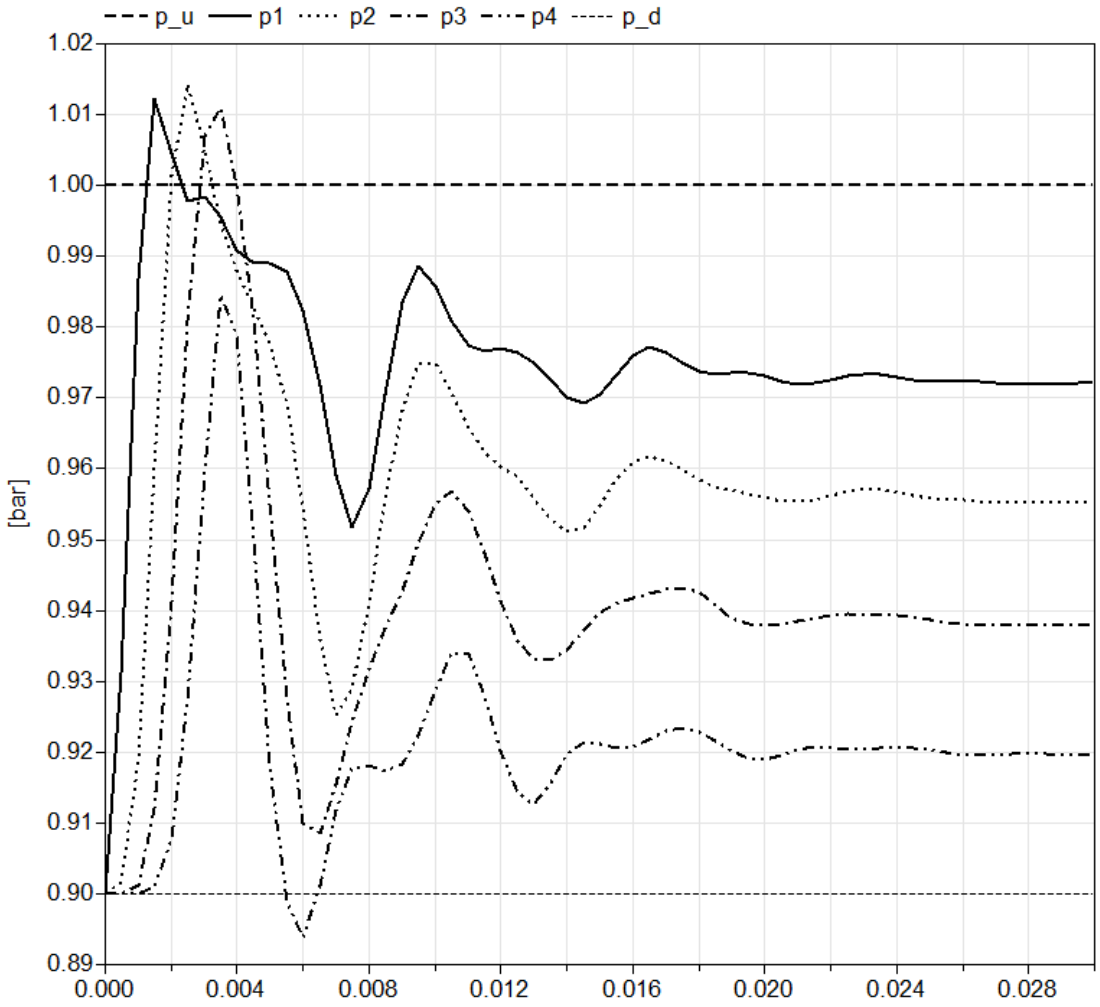

Figure 3.15: The upstream and downstream pressure of the whole pipe is shown as thick and thin dashed lines. The pressure in the volume inbetween the pipe segments are shown and denoted $p1...p4$.

pressure wave dependence is included only through the parameter τ. When using more pipe sections the wave itself is used to calculate the boundary conditions and thus the velocity. When using the static boundary condition the steady value is close to the steady value given using only one pipe segment. When only using total boundary conditions there will be pressure drops over each boundary and the mass flow out of the pipe will decrease as the velocity is decreased. One can note that when using more segments the difference in steady values when using static or not will increase. This is natural since there are more boundaries at which the pressure will drop in an unphysical manner.

The pressure in the four volumes in-between the five QPM segments are shown in Fig. 3.15. Here the progression of the wave is shown. An initial disturbance is travelling and will successively reach the different capacities. After a while the system finds a steady state.

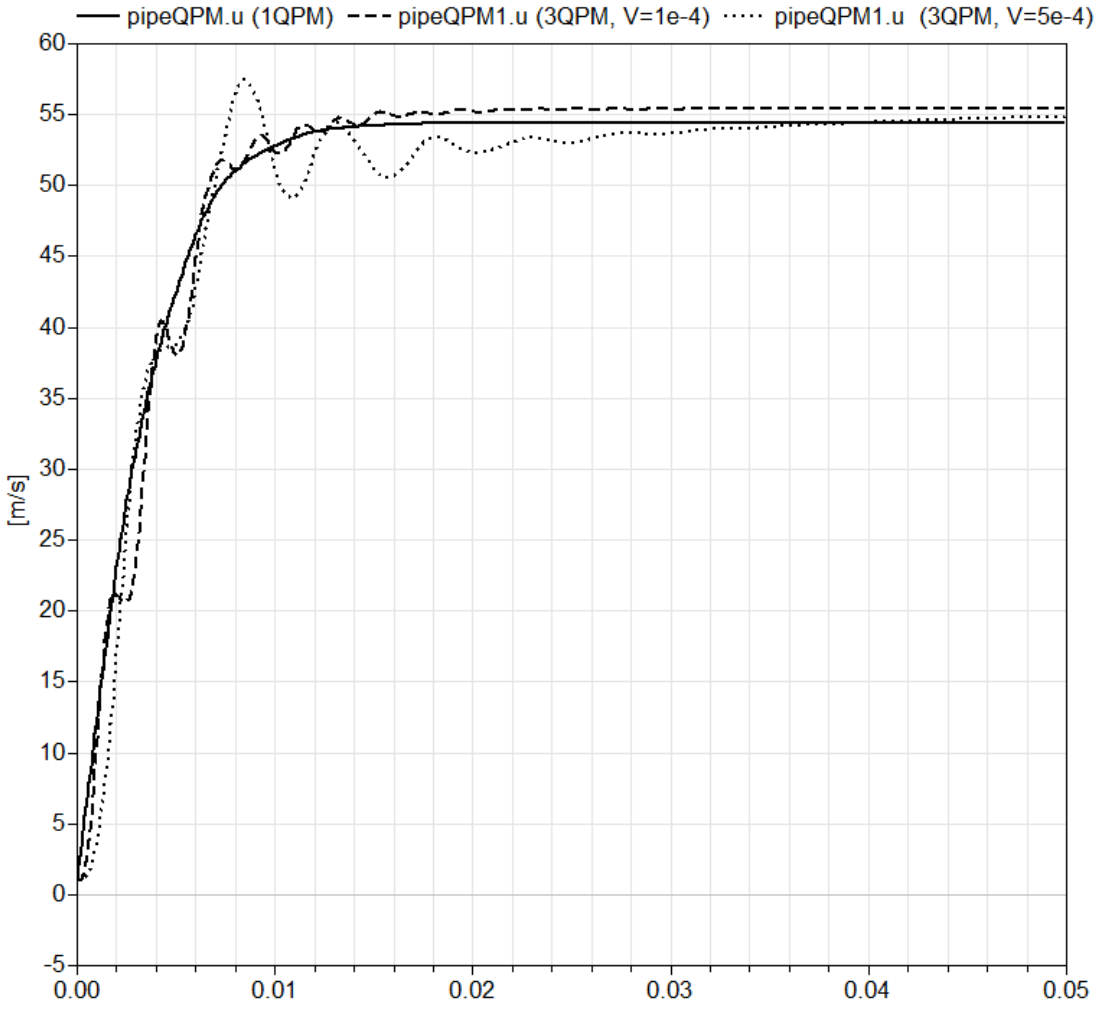

Figure 3.16: The solid line shows the middle point velocity when using only one segment. The dashed line is the result when using three segments with volumes of size 1e-4 m^3. The resulting velocity when using volumes of size 5e-4 m^3 is shown as a dotted line.

When using several segments, capacities are used in-between the branches. A test using three segments, i.e. two capacities, is simulated using different volume sizes. The result is again compared with the result using only one segment and shown in Fig. 3.16. Here, it is shown that there will be some disparities when changing the volume of the capacities.

Chapter 4

Results

In the previous chapter some preliminary tests were simulated to give the reader an insight into different features of the model. Now, this is followed by a comparison of QPM with an available model from Modelon's library, the *simplePipe*. Finally, some tests found in [2] are recomposed. Thereby the model is compared to the Method of Characteristics.

4.1 Comparing with *simplePipe*

To get an idea of the behaviour of the QPM, comparisons with another model are made. The model compared with is the *simplePipe* from Modelon's library. This model can optionally simulate dynamical behaviours of a flow by including the mass flow as a state. The mass flow is then calculated from a friction model. The pressure waves can be captured by connecting several *simplePipe*s with volumes, capacities, in-between them.

To be able to compare the models an appropriate test case has to be found. Starting with the geometry, *simplePipe* is less general than QPM and can solve flow in a pipe with constant cross-section area and with fully open ends. There are a number of different friction models that can be used. The QPM uses the Fanning friction factor, f, to model the friction. Therefore, the friction model Darcy-Weisbach formulation is used in *simplePipe* since the Darcy-Weisbach friction factor $f_D = 4f$. In the Darcy-Weisbach model the mass flow is calculated using the pressure drop

$$\Delta p = f_D \frac{L}{D} \frac{\rho u^2}{2}. \tag{4.1.1}$$

It is important when discretising the pipe, that one uses the Darcy-Weisbach friction factor corresponding to the length of the section, not the length of the whole pipe.

The simulation performed and presented here uses the configuration seen in Fig. 4.1 with three QPM-pipes connected corresponding to a pipe branching off to two cylinders. The friction factor $f = 0.01$, the length of each pipe is 0.5 m and the diameter

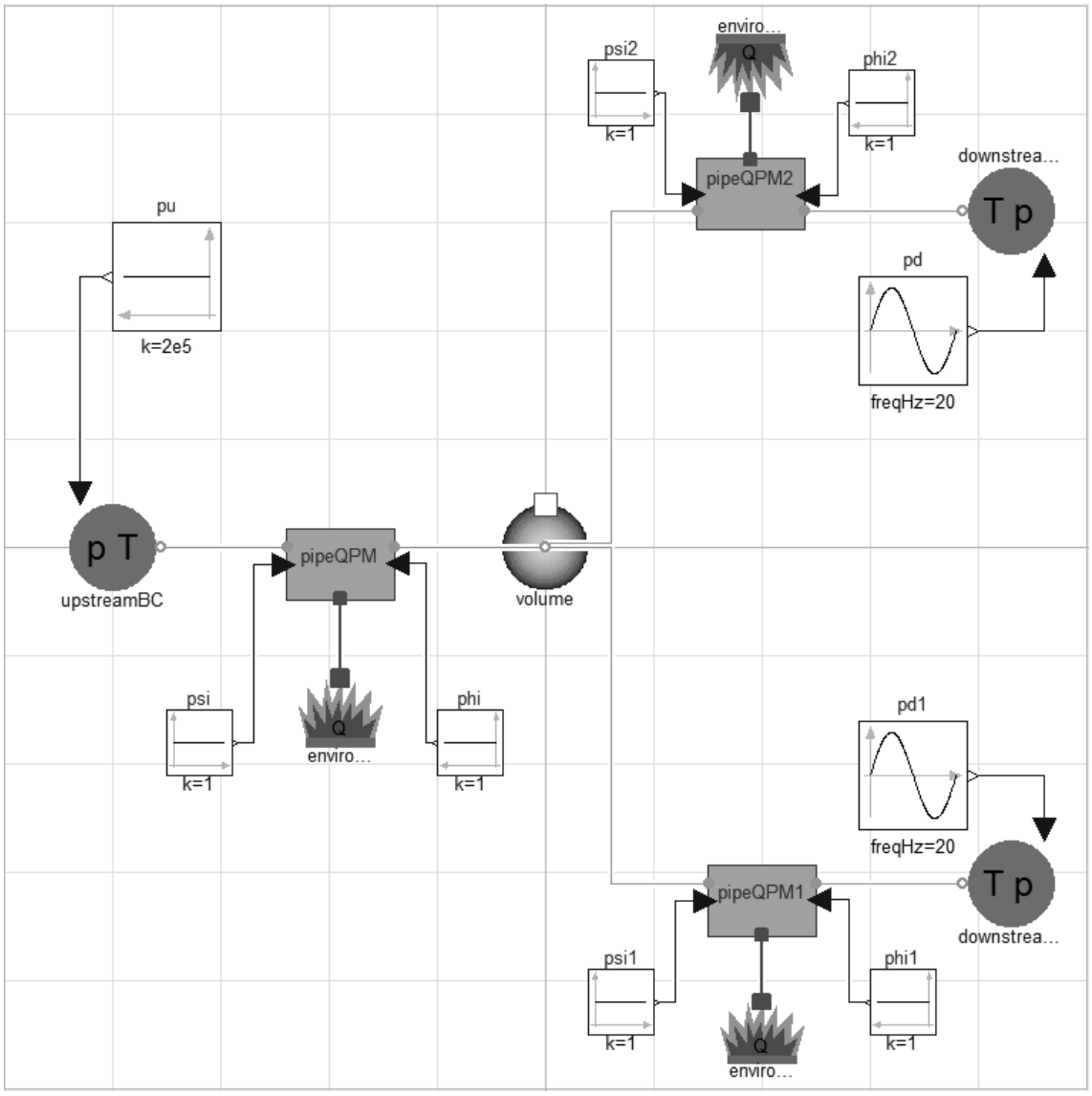

Figure 4.1: Configuration corresponding to a pipe that branches off into two pipes.

is 3 cm. The upstream pressure is constant and the downstream pressures change as sinuses with different phases. The same case is simulated using instead three *simplePipe*s. In Fig. 4.2 the mass outflow from the two branches are shown and in Fig 4.3 the mass outflow the first 0.05 seconds are shown.

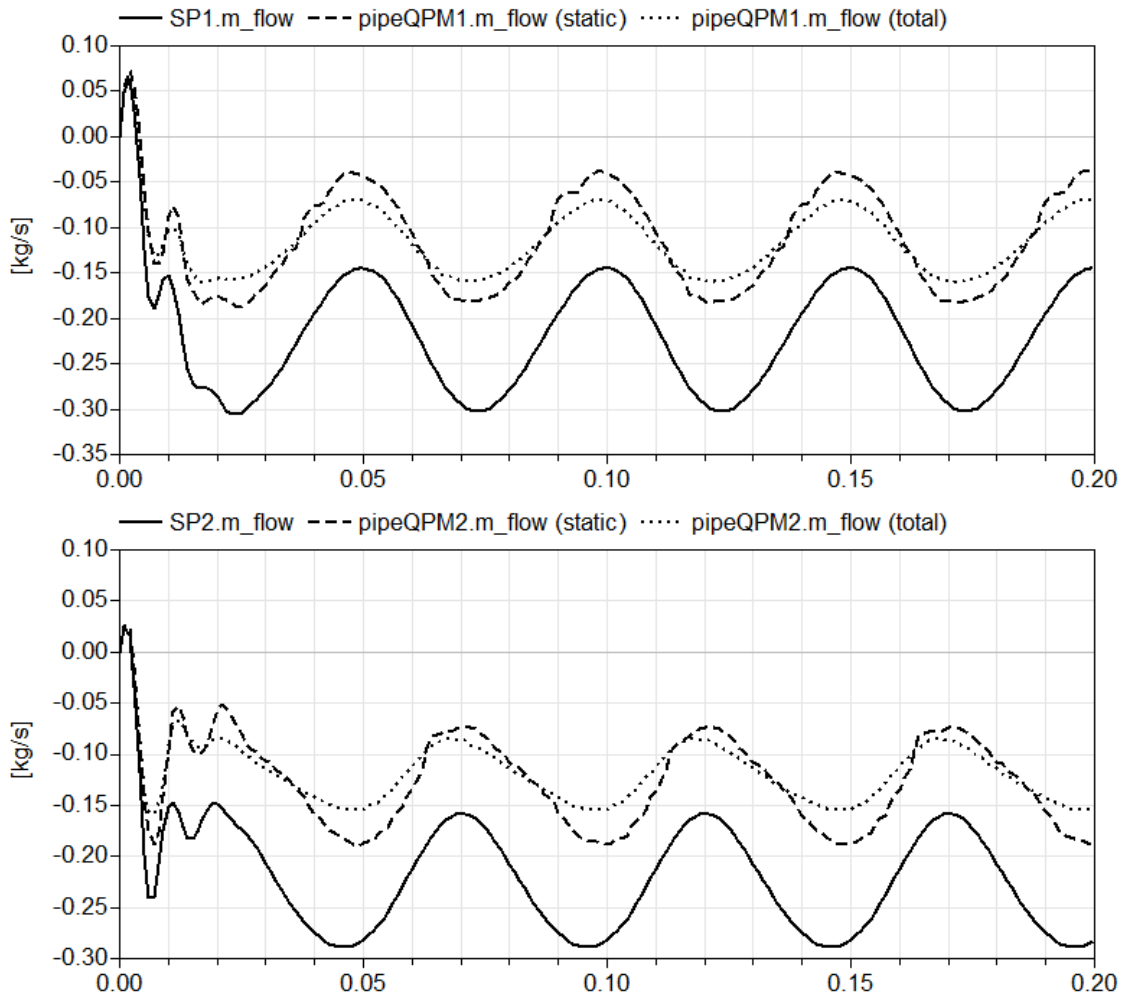

Figure 4.2: The upper figure shows the outflow from the upper branch, and the lower the outflow from the lower branch. The solid lines show the results obtained when simulating using *simplePipe*s, the dashed lines are the results given when using QPM and static boundary conditions. The case is also simulated using QPM and only total boundary conditions and the results are shown as dotted lines.

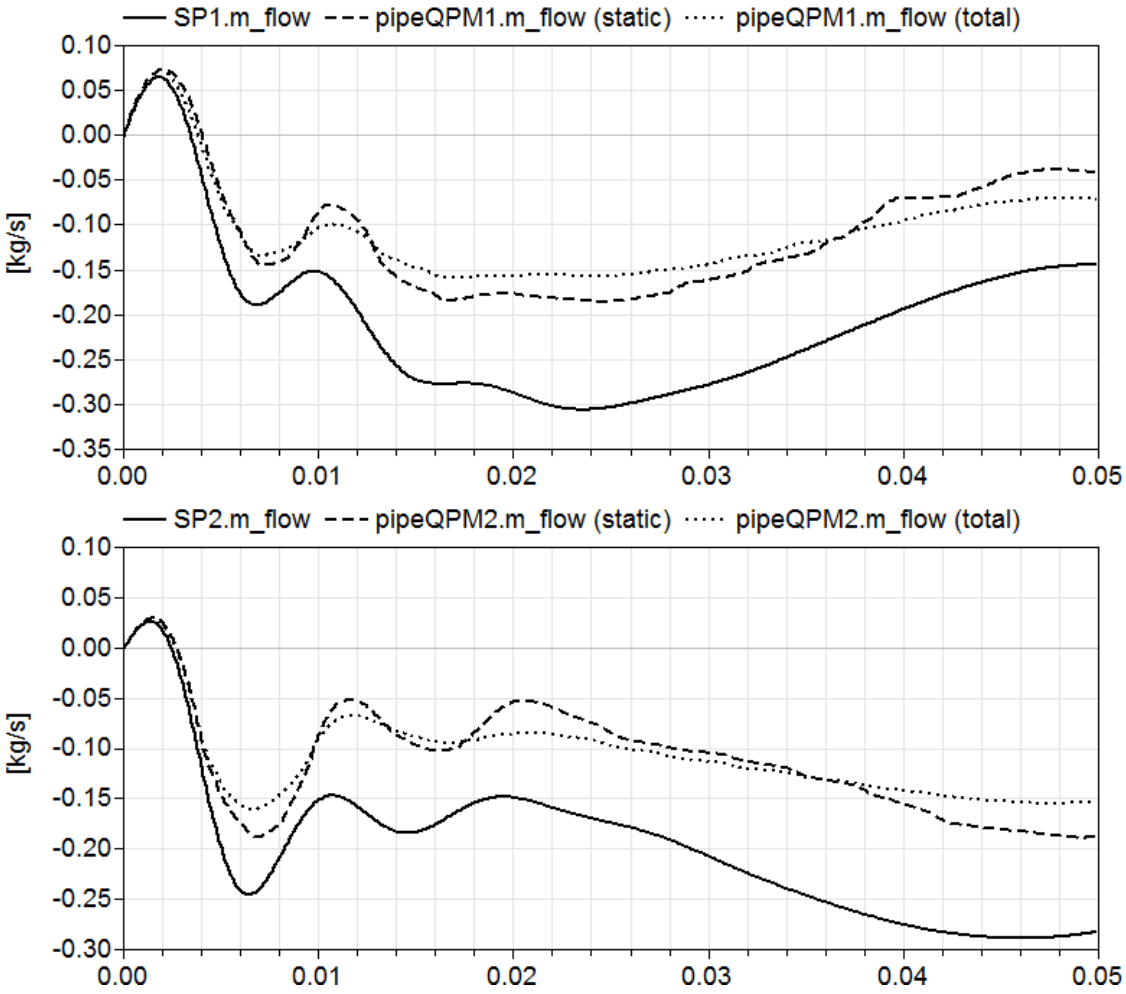

Figure 4.3: The upper figure shows the outflow from the upper branch, and the lower the outflow from the lower branch. The solid lines show the results given when simulating using *simplePipe*s, the dashed lines are the results obtained when using QPM and static boundary conditions. The case is also simulated using QPM and only total boundary conditions and the results are shown as dotted lines.

4.2 Compare with MOC

In [2] some well documented results can be found where QPM is compared to MOC. In the figures from [2] included below the MOC is plotted as a dashed line and their QPM, a bit different than the model presented here, is plotted as a solid line. The first nine cases in [2] will all be simulated using the method presented in this work. These tests uses different friction factors and different area ratios at the ends, for a pipe of length 1 m and a diameter of 0.03 m. At the downstream end the temperature is 400 K and the pressure is 0.9 bar. The upstream pressure is 1 bar or 3 bar and the temperature is 300 K. The upstream area ratio $\psi = 1$ in all cases. The number of segments used and the size of the capacities are not given in [2]. In the simulations performed three QPM-pipes are used with volumes of size $2 \cdot 10^{-4}$ m^3 in-between. The velocity in the middle pipes are compared to the results shown in [2]. Most cases do not differ substantially and hence only some will be displayed graphically. In Table 4.1 all configurations are given with respectively steady state provided through the graphical results in [2] and through the simulations using the QPM with static boundary conditions at the inner boundaries, i.e. the boundaries located inside of the pipe, not bordering a reservoir. In the table the steady value using only one segment is given in parentheses.

Some cases will be analysed in more detail, starting with test case number 1. In this case there is no friction, and the static conditions at the inner boundaries together with the fully opened ends will give problems when discretising the pipe. The dynamic model is as given by Eq. 2.4.25 with $G = 0$, that is the velocity is growing infinitely. The result is shown in Fig. 4.4. Tests 2 and 3 are similar and only test 2 is shown graphically, see Fig. 4.5. In test case number 4 the friction is again zero and only one configuration is simulated; the set up with only one segment, with the result shown in Fig. 4.6. Tests 5 and 6 are similar in shape and only test 5 is shown graphically and is seen in Fig. 4.7. In the last three test cases the downstream area ratio is changed. Here test 7 is shown in Fig. 4.8 and test 8 is shown in Fig. 4.9. The result of test case 9 gives no additional information and will not be shown graphically.

Test case number 9 has however been used to investigate the simulation time. One QPM segment has been simulated with the Runge-Kutta method with time step 0.1 ms for 0.1 s. The case has been simulated with and without the previously mentioned *Advanced.BreakDelayLoops* flag. When simulating without this flag six non-linear systems of equations are present and the simulation took 0.039 s. To investigate the amount of computing time used for each equation in the model the flag $Advanced.GenerateBlockTimers = true$ is used and one can then see that out of a total CPU time of 0.039, the *DynamicsSection* block is essentially responsible for the CPU-time. In the *DynamicsSection* two non-linear system blocks are included. The first is the non-linear system of equation of size two evaluating the temperatures, T_u and T_d. This block takes 0.015 s. The second block includes the non-linear systems of equations evaluating $p_{1,temp}$, $p_{2,temp}$ and $p_{subOut,temp}$ (the help variables for calculation of ,p_1, p_2 and p_{subOut}), and p_{BCd} and p_{BCu}, that are the pressures

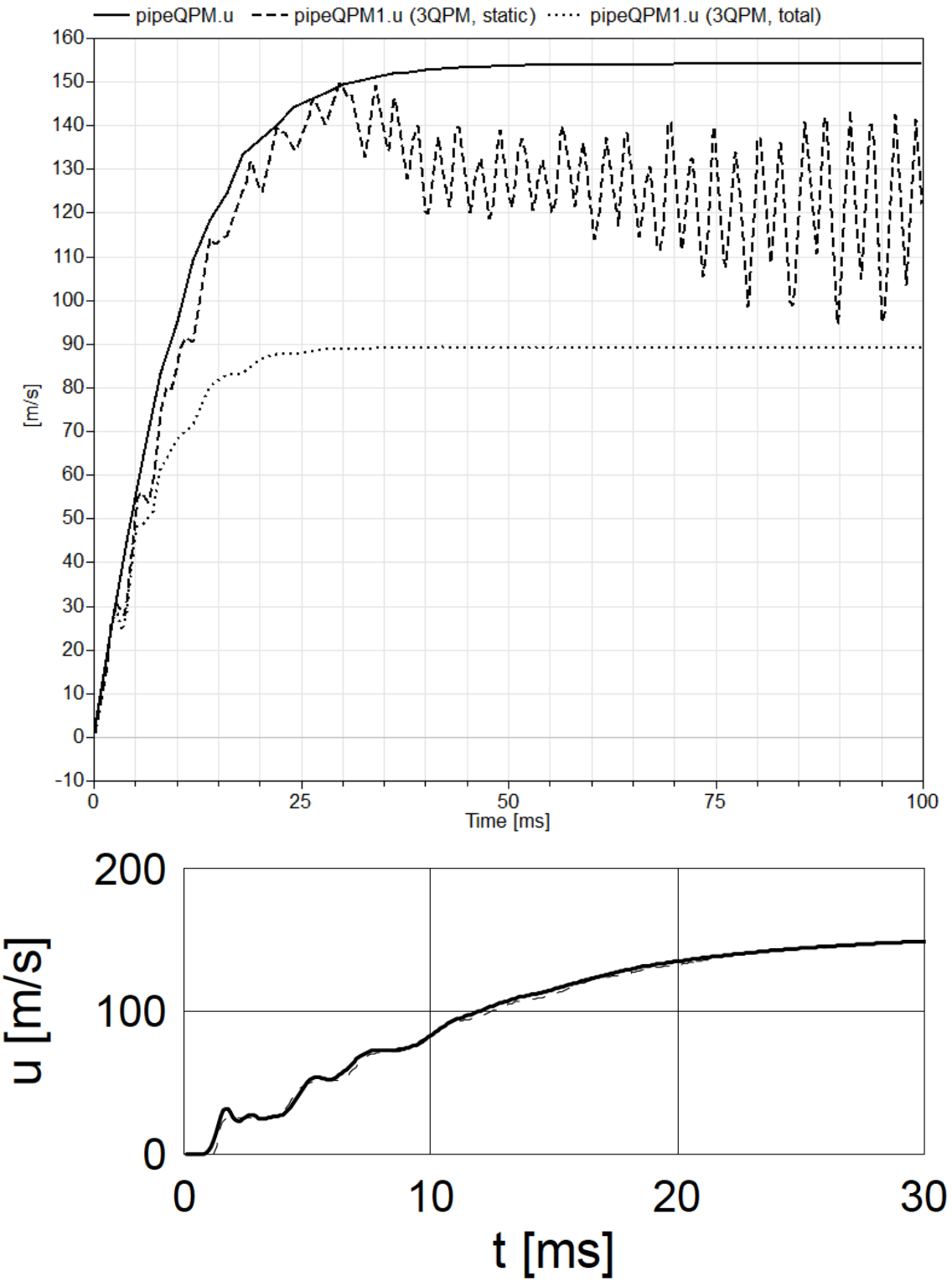

Figure 4.4: Test case 1: The upper figure shows the resulting velocity when simulating with one QPM as the solid line, with three QPMs using static boundary conditions at the inner boundaries as a dashed line, and with three QPMs with total boundary conditions at all boundaries as a dotted line. The lower figure is the result provided for this case in [2]. Note the different time scales.

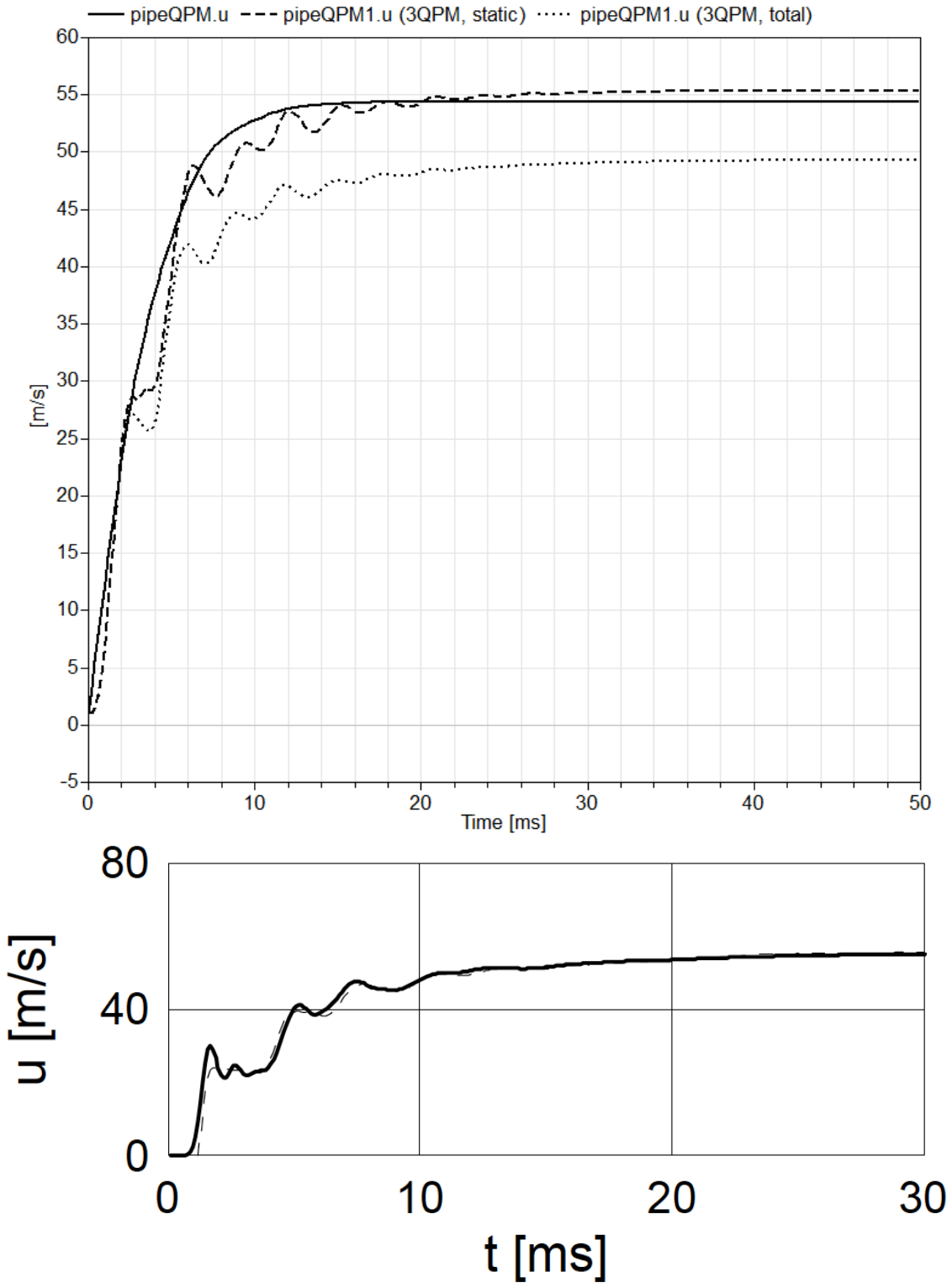

Figure 4.5: Test case 2: The upper figure shows the resulting velocity when simulating with one QPM as the solid line, with three QPMs using static boundary conditions at the inner boundaries as a dashed line, and with three QPMs with total boundary conditions at all boundaries as a dotted line. The lower figure is the result provided for this case in [2]. Note the different time scales.

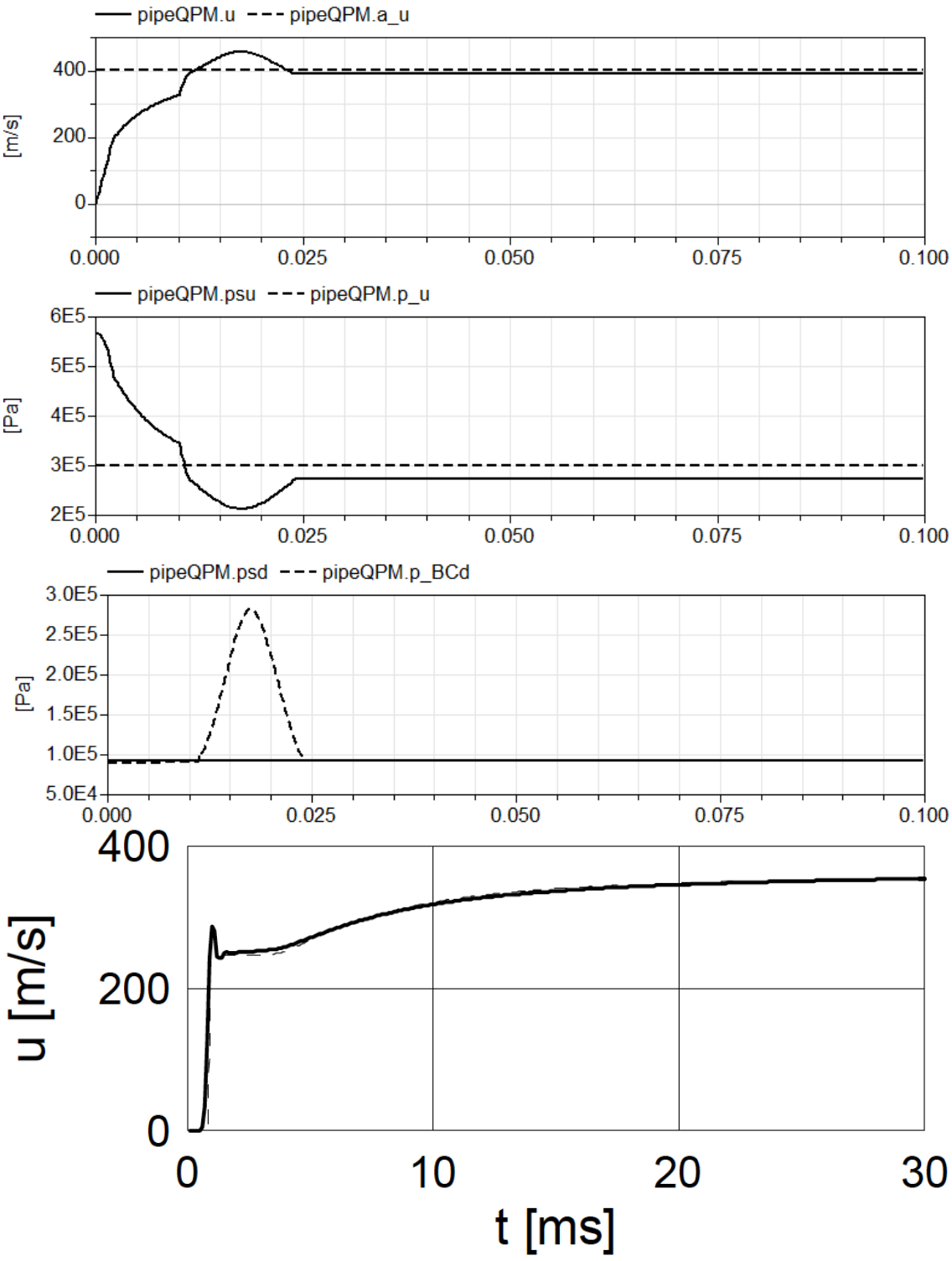

Figure 4.6: Test case 4: The upper figure shows the resulting velocity when simulating with one QPM. The dashed line is the velocity of sound upstream. The middle figures show the sonic limits and corresponding pressures, when $p_u > p_{s,u}$ the inflow is sonic and when $p_{BC,d} > p_{s,d}$ the outflow is sonic. The lower figure is the result provided for this case in [2]. Note the different time scales.

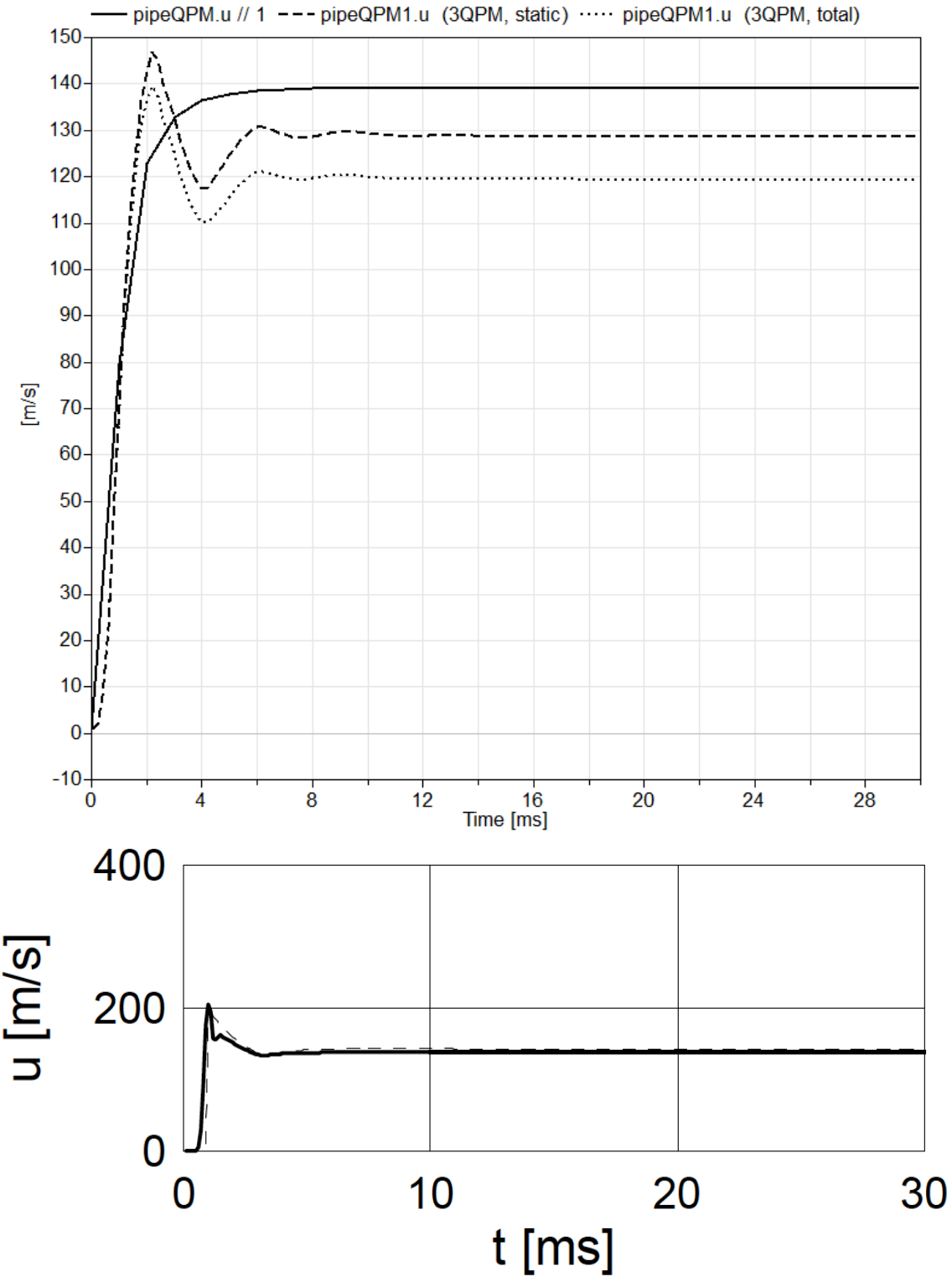

Figure 4.7: Test case 5: The upper figure shows the resulting velocity when simulating with one QPM as the solid line, with three QPMs using static boundary conditions at the inner boundaries as a dashed line, and with three QPMs with total boundary conditions at all boundaries as a dotted line. The lower figure is the result provided for this case in [2].

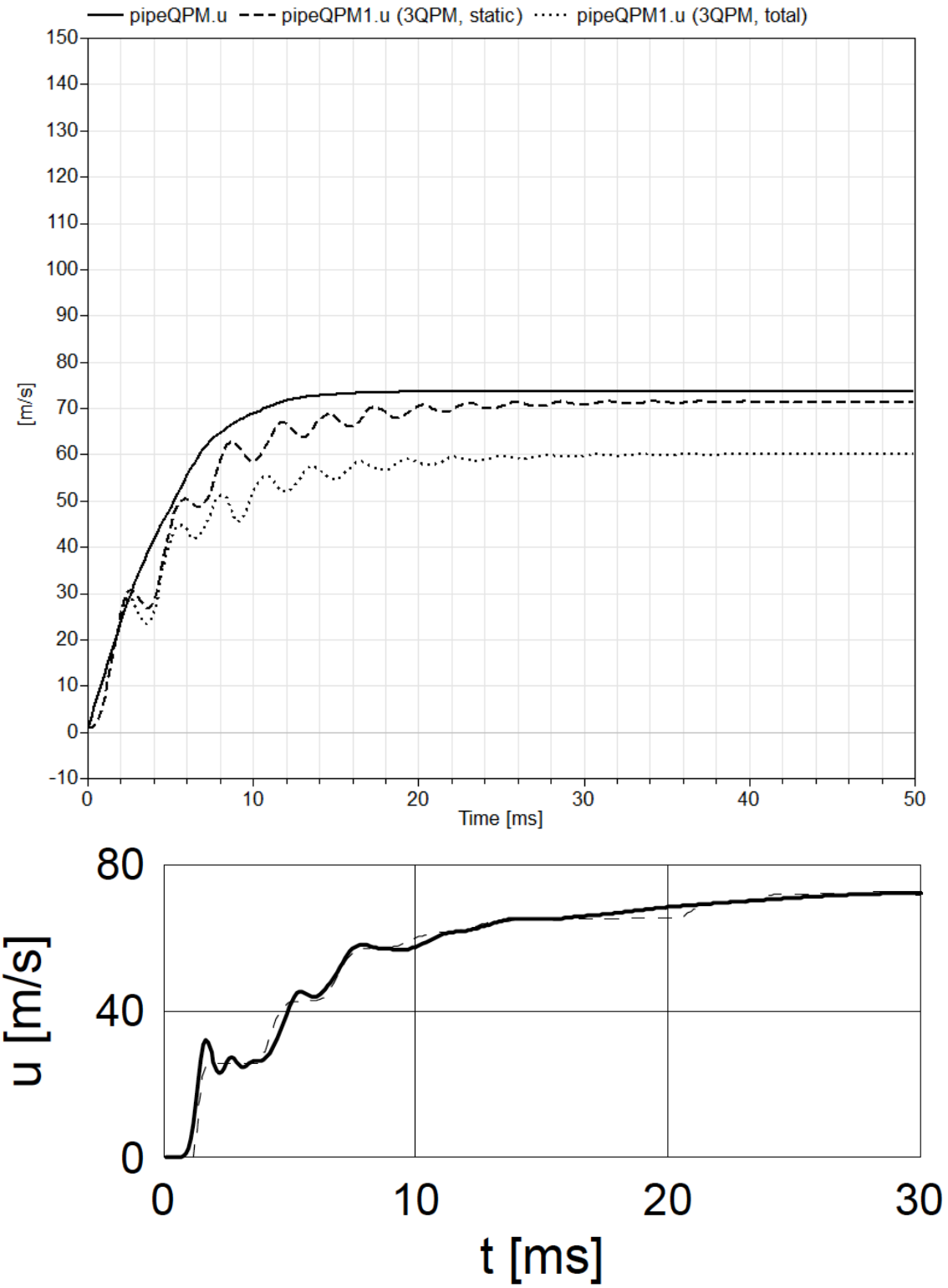

Figure 4.8: Test case 7: The upper figure shows the resulting velocity when simulating with one QPM as the solid line, with three QPMs using static boundary conditions at the inner boundaries as a dashed line, and with three QPMs with total boundary conditions at all boundaries as a dotted line. The lower figure is the result provided for this case in [2]. Note the different time scales.

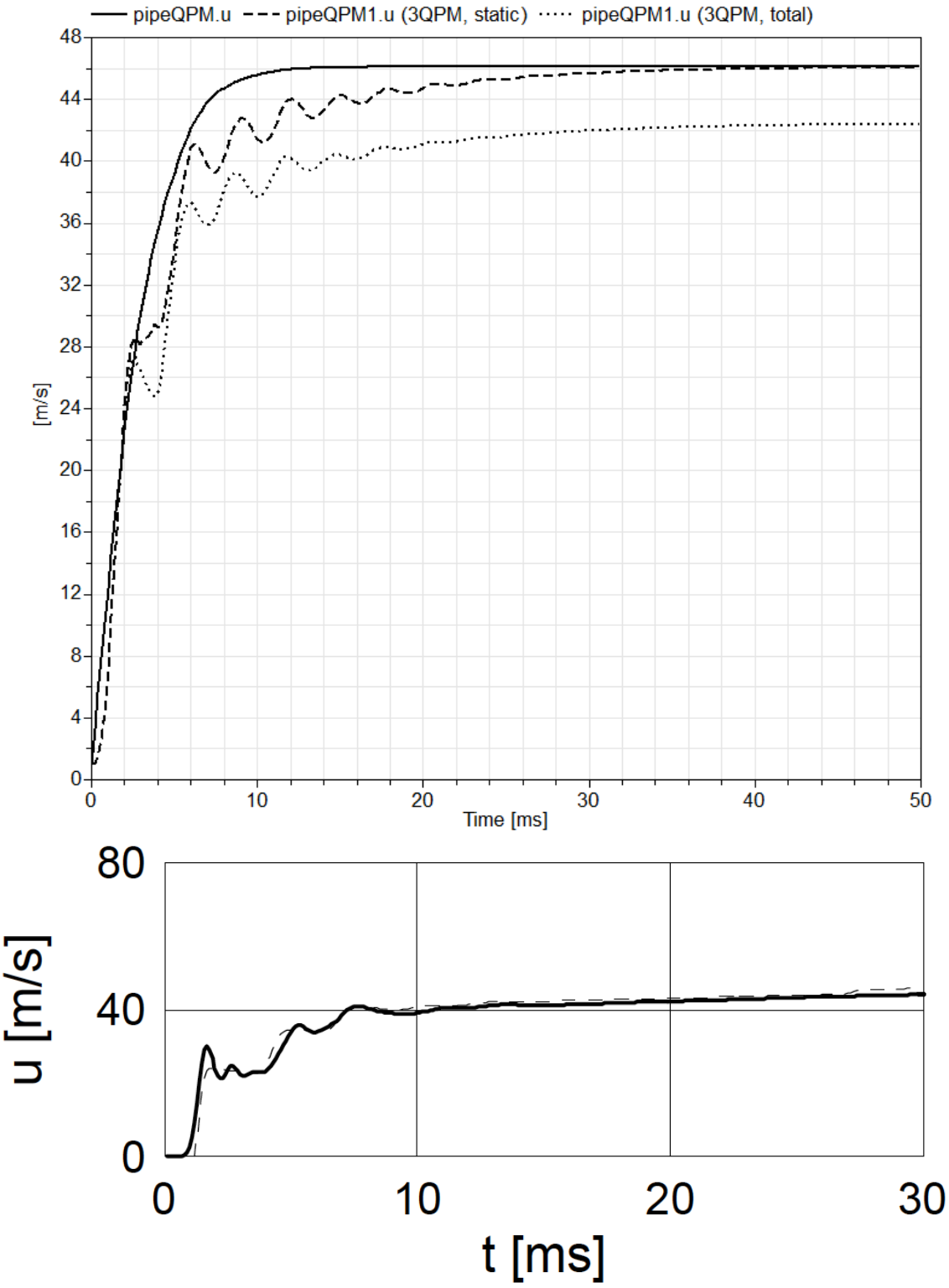

Figure 4.9: Test case 8: The upper figure shows the resulting velocity when simulating with one QPM as the solid line, with three QPMs using static boundary conditions at the inner boundaries as a dashed line, and with three QPMs with total boundary conditions at all boundaries as a dotted line. The lower figure is the result provided for this case in [2]. Note the different time scales.

Test Case	p_u [bar]	ϕ	f	u_∞ [m/s] as in [2]	u_∞ [m/s]	
1	1	1.0	0.00	148	-	(154)
2	1	1.0	0.05	55	55	(54)
3	1	1.0	0.10	40	41	(40)
4	3	1.0	0.00	375	-	(390)
5	3	1.0	0.05	137	128	(139)
6	3	1.0	0.10	105	97	(103)
7	1	0.5	0.00	72	71	(74)
8	1	0.5	0.05	44	46	(46)
9	1	0.5	0.10	35	36	(36)

Table 4.1: The different cases simulated with the steady values given in [2] and given simulating the QPM presented in this report. The steady value, in the last column, is taken in the middle of three QPM pipes simulated using static conditions at the inner boundaries and in parentheses the steady value in the middle using only one segment is given.

Test Case	p_d [bar]	ϕ	Diam. Variation	u_∞ [m/s] as in [2]	u_∞ [m/s]
10	0.9	1.0	0 %	148	154
11	0.9	1.0	-4 %	13	146
12	0.6	1.0	0 %	330	336
13	0.6	1.0	-4%	27	295
14	0.9	0.5	0 %	72	74
15	0.9	0.5	-4 %	5	71

Table 4.2: The different cases simulated with the steady values given in [2] and given by the QPM presented in this report.

inside of the boundaries. The four first of these non-linear systems of equations take 0.023 s and the last, p_{BCu}, takes 0.001 s. Using $Advanced.BreakDelayLoops = true$ there is four non-linear systems of equations. The *DynamicsSection* is again dominating and has a CPU time of 0.038 s. The first non-linear system of equations for the temperatures now takes 0.014 s to evaluate and the three non-linear systems of equations for the evaluation of $p_{1,temp}$, $p_{2,temp}$ and $p_{subOut,temp}$ takes 0.024 s.

In [2] a number of tests are also simulated with converging and diverging pipes. Only a few results will be presented and only the steady values will be given. Using a convergent pipe the upstream pressure is kept at 1 bar, $\psi = 1$ and there is no friction. The mean diameter is set to 0.03 m and results with a diameter variation of 4 % over the pipe will be presented. This means that the upstream diameter is 3.06 cm and the downstream diameter 2.94 cm. By definition $\frac{dF}{dx} = \frac{F_d - F_u}{L}$, where F_u is the area of the pipe at the upstream end and F_d is the area of the pipe at the downstream end. Then $\frac{dF}{dx} = -5.7699 \cdot 10^{-5}$ when the mean diameter is 3 cm and the diameter variation 4 %. Using this the result is shown in Table 4.2. In the divergent case it is assumed that the upstream pressure is still 1 bar, $\phi = 1$ and that the mean diameter is still 3 cm. The downstream pressure is 0.97 bar and ψ is varying between 1 and 0.5. In this case an increase of the diameter of 1 % over

the pipe is used. However, only the base case is presented here, when there is no diameter variation. In this case the steady state given in [2] is 125 m/s and the result given in simulations using the QPM presented here gives a steady value of 80 m/s. The difference is so large; one has to assume a misinterpretation of one ore more parameters has been made. Thereby, this result will conclude this section.

Chapter 5

Discussion

In the previous section comparisons with *simplePipe* from Modelon's library and with some of the results presented in [2] have been made. As always, without experimental results it is hard to draw any certain conclusions. In this section the results given shall nevertheless be discussed and analysed. Further improvements of the model will also be discussed.

5.1 Comparing with *simplePipe*

When comparing with *simplePipe* in Section 4.1 one can see that the magnitude differs substantially between the two configurations, using QPM pipes or *simplePipes*. The homentropic assumptions used in the QPM could be one explanation for this behaviour. Using only one QPM versus one *simplePipe*, the steady value is higher using *simplePipe*. In the *simplePipe* model there is no storage of energy or mass, and the flow is steady. The QPM uses the governing equations, the continuity, momentum and energy equation for non-steady flow and these give rise to the non-homentropic parts (α_1 and α_2) decreasing the steady value. This could possibly be the main cause of this difference. Put aside the magnitude difference the shape of the results are similar which could be an indication that the model captures the wave phenomena in a correct manner.

5.2 Comparing with MOC

In Section 4.2 some cases are simulated in order to compare with the results in [2], thus comparing with MOC. In most cases the steady values become close to the values in [2], these results are a bit better when only using one segment. To capture some of the initial oscillations discretisations are required and static boundary conditions should be used at all inner boundaries, where the boundary is not bordering

a reservoir.

The first case simulated is seen in Fig. 4.4. Using only one segment one gets a steady state value close to the one given in [2] by MOC. One major difference is the time it takes for the velocity to reach the steady value. MOC seems to predict a faster convergence towards steady state than the QPM presented here. When using a discretised pipe both the result using static boundary conditions and using total boundary conditions at inner boundaries are shown. Using total boundary conditions gives a stable but immensely wrong result. Using static boundary conditions will give problems. As been said before, the dynamic model used in a case with static upstream condition and fully opened downstream end, as is the case in the last two QPM pipe segments, given in Eq. 2.4.25, will when the friction is zero make the velocity grow infinitely. Obviously this is a problematic case. However, thankfully, in reality there is no engine application where the friction will actually be zero.

In the second case friction is included and both the case with one segment and the discretised case using static boundary conditions will give correct steady values, see Fig. 4.5. Again, the MOC will predict a faster reaching of steady state. There seems to be some discrepancy when comparing the initial oscillations. However, there is also some difference in the results between the MOC and the QPM used in [2]. The version of the QPM introduced in this report uses somewhat different boundary conditions and the parameter $k_0 = 1$, and the first peak present in the result from [2] is not present when using this model, nor in the MOC results, the dashed line. Around 10 ms, up to 20 ms, the QPM gives results that oscillates a bit more than MOC.

When increasing the upstream pressure and using again zero friction, only the result from one segment is presented. This case is not simulated for a discretised pipe since the results will be unstable when the friction is zero. In this case there is a clear disparity in the results. The QPM will actually give a result where the velocity for a short time overshoots the velocity of sound, see Fig 4.6. After this the velocity finds a steady value slightly below the speed of sound, about 15 m/s higher than the steady value given by the MOC. The velocity increases until around 300 m/s when the upstream pressure exceeds the sonic limit and the inflow becomes sonic. The outflow becomes sonic soon after this. In the model used in [2] and probably in the MOC used, there is no sonic limit for the inflow and the sonic limit is for the outflow in [2] $p_u > p_s = p_d$, always fulfilled in this configuration. An extra condition is used in [2] when $p_u > p_s = p_d$, when $u < u_s$ where u_s is the velocity given by the continuity equation for a flow where the velocity in the outlet throat equals the local speed of sound. This is also the velocity where the sonic outflow boundary curve is exceeding p_s and it is $u_s = \phi a_u \left(\frac{p_d}{p_u}\right)^{\frac{\kappa-1}{2\kappa}} \left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)^{\frac{2\kappa}{\kappa-1}}_{cr}$, in this case equal to a_d. These are different than the sonic limits used in this model so the disparities in the results are not that surprising. The obvious question; which is more correct, requires experimental data or CFD simulations to be answered.

In the next two cases the upstream pressure is still increased, but friction is in-

troduced. In these cases there is somewhat more difference in results when using a discretised pipe, for reasons unknown, then the steady state is around 10 m/s lower compared to the result given using a non-discretised pipe and using MOC. The shape however, with a high peak in the beginning, is similar to the result given in [2], see Fig. 4.7.

The next test is interesting since it is again zero friction. The throat at the downstream end will in this case stabilise the model enabling discretisation. In the middle segment the dynamic model will be the same as in the first test case, but the last segment will have a downstream end that is not fully opened. Again there are some similarities between QPM and MOC, see Fig. 4.8. The main difference is that QPM yet again keeps oscillating for a longer time and reaches the steady value a bit later. The same holds for the last case seen in Fig. 4.9

One major disadvantage when comparing the results of the model introduced in this report with the results in [2] is that the number and size of volumes are not noted. As seen in Fig. 3.16, the size of the volume will influence the result. Different volume sizes could maybe explain the different oscillating behaviours. It is also not stated what boundary conditions are used in the MOC simulations, but one could assume the same boundary conditions are used as in the QPM presented in [2]. Since the model introduced here is not using the same boundary conditions as in [2] this could give rise to the disagreements seen in the results.

As has been seen, most of the simulation time is spent on the non-linear systems of equations present in the model. Since these have to be solved for each pipe section used when discretising, this is a large disadvantage for the model that will probably prohibit real-time simulations. The main part of the time is laid on the non-linear system of equations solving $p_{1,temp}$, $p_{2,temp}$ and $p_{subOut,temp}$. The existence of these non-linear systems of equations has its origin in the way the model uses the slope of the boundary equations together with the property of the subsonic outflow boundary condition, not being able to be linearised. In [2] this is solved through a less general assumption about the geometry; that one of the ends is always fully opened. Two cases are considered; total conditions at the inflow and $\phi = 1$, and static inflow and $\psi = 1$. These assumptions may be enough for an engine application and in these cases no non-linear systems of equations are needed. When $\phi = 1$, the subsonic boundary equation for outflow is constant. When $\psi = 1$, the inflow condition is static, thus constant, and through the intersection the steady values for the velocity and the pressure is known and could be used to get an approximate slope. On one hand this restriction of geometry could force a discretisation in the cases when none of the ends are fully opened. On the other hand, the rescindment of the non-linear systems of equations needed to solve the linearisation problem of the subsonic outflow case will reduce the simulation cost for each segment extensively. The need of smaller time steps for shorter pipe sections will probably not exceed this gain, and it would probably be possible to simulate this in real-time.

In the last tests the diameter varied over the pipe length and the disparities in the results were vast. When simulating the diverging pipe case even without the

diameter variation the results were so diverse, there was no meaning in continuing the tests in this case. When using a converging pipe the results given in [2] showed a huge reduction of velocity even for small diameter changes. The reason behind these different results is presumably, in the diverging pipe case; a lack of information, and in the converging pipe case; a lack of details in the description leading to misinterpretations about the parameter set-ups. One motivation for this assumption is the physical meaning of the results given in [2]. Suppose the parameter set-up were correctly interpreted, the difference between the upstream and downstream diameter where 1.2 mm. That is, looking at test case 10 and 11, without any diameter variance the velocity is 148 m/s, a 1.2 mm variation of the diameter over the whole pipe length of 1 m decreases the velocity to 13 m/s. This is not reasonable. If that was the case, a pipe designed with accuracy less than a tenth of a millimetre could get a velocity variation between 150 and 10 m/s. The QPM presented here gives a much more reasonable change in velocity in accordance to the small change in diameter. To conclude, a plausible explanation to why the MOC in [2] is giving so different results is that the two cases simulated are in fact not the same at all.

5.3 Conclusion

The goal of this project was to implement a model that is able to simulate the flow in a pipe, capturing the wave phenomena, in real-time. The Quasi-Propagatory Model found was a good candidate and a slightly modified version of the, in several articles, presented QPM, has been exhibited in this work. The lumping property of the model has proven to be efficient and there is potential to simulate the model in real-time especially when using only one segment. However, to capture the wave phenomena discretisation is required and to remain stability when the length decreases in each segment the time step may need to be decreased. At the same time, when discretising the pipe the number of non-linear systems of equations needed to be solved at each time step is increased and it may altogether prohibit the possibility of simulating in real-time. One solution would be to avoid these non-linear systems of equations by loosening the generality of the model. To ensure the accuracy of the model experimental data would be optimal, but through comparisons with other models the model is found to give plausible results, have in mind the different assumptions made.

Bibliography

[1] D.E. Winterbone and R.J. Pearson. *Design Techniques for Engine Manifolds: Wave Action Methods for IC Engines*. Professional Engineering Pub. Limited, 1999.

[2] R. Cipollone and A. Sciarretta. The quasi-propagatory model: A new approach for describing transient phenomena in engine manifolds. *SAE Technical Paper 2001-01-0579, 2001, doi:10.4271/2001-01-0579*, 2001.

[3] D. Mezher, H.and Chalet, J. Migaud, and P. Chesse. Frequency based approach for simulating pressure waves at the inlet of internal combustion engines using a parametrized model. *Applied Energy, Vol. 106, June 2013, p. 275-286*, 2013.

[4] S. Stockar, M. Canova, Y. Guezennec, A. Della Torre, G. Montenegro, and A. Onorati. Modeling wave action effects in internal combustion engine air path systems: Comparison of numerical and system dynamics approaches. *International Journal of Engine Research, August 2013, 14(4):391-408*, 2013.

[5] R. Cipollone, L. Martella, L. Scarpone, and R. Valente. New modeling to predict the fluid dynamic transient phenomena in ice ducts. *SAE Technical Paper 2008-01-2389, 2008, doi:10.4271/2008-01-2398*, 2008.

[6] R. Cipollone and A. Sciarretta. A new modelling for the air and gas dynamics in ice manifolds oriented to air-fuel ratio control. *SM - PUBLICATIONS - ICE; 32, 1;1103-144, ASME: Internal Combustion Engine Division, Spring technical conference*, 1999.

[7] R. Cipollone and A. Sciarretta. On the air dynamics in ice intake manifolds the development of a quasi-propagatory model. *Control and Systems, Mediterranean conference; 6:th, Control and Systems; 189-197*, 1998.

[8] D.E. Winterbone and R.J. Pearson. *Theory of engine manifold design: wave action methods for IC engines*. Professional Engineering Pub., 2000.

[9] R.S. Benson, J.H. Horlock, and D.E. Winterbone. *The Thermodynamics and Gas Dynamics of Internal-combustion Engines*. Number v. 1 in Oxford science publications. Clarendon Press, 1982.

[10] B. Johansson. *Förbränningsmotorer*. Institutionen för värme- och kraftteknik, Lunds tekniska högsk., 2003.

[11] V.A.W. Hillier and P. Coombes. *Hillier's Fundamentals of Motor Vehicle Technology*. Number bd. 1 in Hillier's Fundamentals of Motor Vehicle Technology. Nelson Thornes, 2004.

[12] Flow Science. Boundary conditions - pressure. `http://www.flow3d.com/cfd-101/cfd-101-boundary-conditions-pressure.html`, February 2014.

[13] A Sciaretta. *La regolazione del rapporto aria-combustibile in MCI ad accensione comandata*. PhD thesis, University of LAquila, Italy, 1999.

[14] Modelica Association. Modelicatm - a unified object-oriented language for physical systems modeling. `https://modelica.org/documents/ModelicaTutorial14.pdf`, December 2000.

[15] D. Brück, H. Elmqvist, H. Olsson, and S.E. Mattsson. Dymola for multi-engineering modeling and simulation. In *2nd International Modelica Conference, Proceedings*, 2002.

[16] *Dymola Introduction Course Volume I*. Modelon AB, 2012.

[17] *Dymola Introduction Course Volume II*. Modelon AB, 2014.

Appendix A

Boundary Conditions

In this section a number of different boundary conditions are going to be derived using homentropic and non-homentropic assumptions. From the boundary conditions the variables A, B, p_{u0} and p_{d0} used in QPM can be calculated. This is done through a linearisation of the boundary condition around u^*. Before deriving the boundary conditions identification of the variables is done. Assuming a boundary condition $p = f(u)$, the linearisation around u is $p = f(u) - f'(u^*)u^* + f'(u^*)u$. Comparing this with 2.4.13 and 2.4.14 one can identify

$$p_{u0} = f(u^*) - f'(u^*)u^* \tag{A.0.1}$$

and

$$A = -f'(u^*), \tag{A.0.2}$$

for an upstream boundary condition $p = f(u)$, and

$$p_{d0} = f(u^*) - f'(u^*)u^* \tag{A.0.3}$$

and

$$B = f'(u^*), \tag{A.0.4}$$

for a downstream boundary condition $p = f(u)$.

A.1 Homentropic Inflow

The first boundary to be modelled is a simple case; inflow through a completely open boundary from a reservoir of stagnation conditions, p_0 and a_0, assuming homentropic flow, that is, no entropy change, [9](p. 105-106). The derivation starts from the so-called ellipse of energy, the energy equation for steady flow,

$$a_u^2 = a^2 + \frac{\kappa - 1}{2}u^2. \tag{A.1.1}$$

This equation is derived in [9](p. 38-39) and is the energy equation for flow through a duct or nozzle with constant enthalpy. The Eq. A.1.1 is written in non-dimensional form as

$$\mathcal{A}_u = \mathcal{A}^2 + \frac{\kappa - 1}{2} U^2, \tag{A.1.2}$$

where $\mathcal{A}_u = \frac{a_u}{a_{ref}}$, $\mathcal{A} = \frac{a}{a_{ref}}$ and $U = \frac{u}{a_{ref}}$.

Taking $a_{ref} = a_u$, using that $a^2 = \kappa RT$ for a ideal gas and $\frac{a}{a_u} = \left(\frac{p}{p_u}\right)^{\frac{\kappa-1}{2\kappa}}$, that is the conservation of entropy, one get the expression found in [6] and [7],

$$u^2 = \frac{2\kappa R T_u}{\kappa - 1} \left(1 - \left(\frac{p}{p_u}\right)^{\frac{\kappa-1}{\kappa}}\right). \tag{A.1.3}$$

A.1.1 Partially Open Boundary

In this case the inflow through a partially open end from a reservoir of static conditions, p_u and a_u is considered. The ratio between the inlet area and the pipe area is called ψ. To get an explicit expression for this case [6] assumes that the density is constant between the throat and the inlet. The general case can be found in [9](p. 130-131).

When the flow enters the region, through the throat, there will be a zone of recirculating flow. The point in the pipe where this zone ends is called point 1 in [9]. The inlet is called point 2. The continuity equation for steady flow is

$$u_2 \rho_2 F_2 = u_1 \rho_1 F_1 \Longleftrightarrow u_1 = u_2 \left(\frac{F_2}{F_1}\right) = u_2 \psi, \tag{A.1.4}$$

assuming constant density. This simplification, together with the assumption that u_2 can be calculated as the velocity of the inflow through an open boundary, i.e. through Eq. A.1.3, as well as assuming the pressure $p = p_1 = p_2$, gives the expression for the inlet velocity ($u = u_1$) simply as

$$u^2 = \frac{2\kappa R T_u}{\kappa - 1} \psi^2 \left(1 - \left(\frac{p}{p_u}\right)^{\frac{\kappa-1}{\kappa}}\right). \tag{A.1.5}$$

Using this expression one can calculate the variables A and p_{u0} as

$$A = \frac{\kappa u^*}{\psi^2 a_u^2} p_u \left(1 - \frac{\kappa - 1}{2\psi^2} \left(\frac{u^*}{a_u}\right)^2\right)^{\frac{1}{\kappa-1}} \tag{A.1.6}$$

and

$$p_{u0} = p_u \left(1 - \frac{\kappa - 1}{2\psi^2}\left(\frac{u^*}{a_u}\right)^2\right)^{\frac{\kappa}{\kappa-1}} + Au^*. \tag{A.1.7}$$

For sonic flow $u^2 = a^2 = a_u^2\left(\frac{p}{p_u}\right)^{\frac{\kappa-1}{\kappa}}$. Using this, the critical pressure ratio, the pressure difference that will make a velocity equal to the local velocity of sound choking the inflow, can be found as

$$u^2 = a_u^2\left(\frac{p}{p_u}\right)^{\frac{\kappa-1}{\kappa}} = \frac{2\kappa RT_0}{\kappa - 1}\psi^2\left(1 - \left(\frac{p}{p_u}\right)^{\frac{\kappa-1}{\kappa}}\right) \Longleftrightarrow \frac{p_u}{p} = \left(\frac{\kappa - 1 + 2\psi^2}{2\psi^2}\right)^{\frac{\kappa}{\kappa-1}}. \tag{A.1.8}$$

The critical pressure is $p_s = p\left(\frac{\kappa-1+2\psi^2}{2\psi^2}\right)^{\frac{\kappa}{\kappa-1}}$. If the pressure upstream $p_u > p_s$ the driving pressure upstream is that much higher than the pressure in the pipe that the flow will be sonic in the throat. Then another boundary equation has to be used but this will not be presented here.

A.2 Homentropic Outflow

The derivation of the boundary condition for outflow through a nozzle assuming isentropic change can be found in [9](p. 117-120). The derivation starts with the energy equation in dimensionless form with static conditions outside the nozzle, p_d and a_d,

$$\mathcal{A}_0^2 = \mathcal{A}^2 + \frac{\kappa - 1}{2}U^2 = \mathcal{A}_d^2 + \frac{\kappa - 1}{2}U_d^2, \tag{A.2.1}$$

with $\mathcal{A}_0$ corresponding to the stagnation speed of sound in the pipe and $\mathcal{A}$ corresponding to the speed of sound at the boundary just inside of the nozzle. Here the notation is in analogy with the inflow through a open boundary case, i.e. $\mathcal{A} = \frac{a}{a_{ref}}$.

Assuming isentropic flow and using that $\frac{\rho}{\rho_{ref}} = \left(\frac{a}{a_{ref}}\right)^{\frac{2}{\kappa-1}}$ for isentropic flow, the continuity equation, $\rho uF = \rho_d u_d F_d$ can be written as

$$U\mathcal{A}^{\frac{2}{\kappa-1}} = \phi\mathcal{A}_t^{\frac{2}{\kappa-1}}U_t, \tag{A.2.2}$$

where $\phi = \frac{F_d}{F}$. Combining the dimensionless energy equation and the dimensionless continuity equation one can now derive the equation

$$U^2 = \frac{2}{\kappa - 1}\frac{\mathcal{A}^2 - \mathcal{A}_d^2}{\frac{1}{\phi^2}\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)^{\frac{4}{\kappa-1}} - 1}. \tag{A.2.3}$$

For a subsonic flow $a_{ref} = a_d$, giving the boundary equation

$$u^2 = \frac{2\kappa R T_d}{\kappa - 1} \frac{\left(\frac{p}{p_d}\right)^{\frac{\kappa-1}{\kappa}} - 1}{\frac{1}{\phi^2}\left(\frac{p}{p_d}\right)^{\frac{2}{\kappa}} - 1}. \tag{A.2.4}$$

For the sonic case when $u_d = a_d$, the critical pressure is $p_s = p_d \left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr}^{\frac{2\kappa}{\kappa-1}}$. When the pressure in the pipe $p > p_s$ the flow will be choked, the velocity in the outlet will be equal to the local speed of sound. The continuity equation can be written as

$$U\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)^{\frac{2\kappa}{\kappa-1}} = \phi U_d = \phi \mathcal{A}_d \iff \frac{U}{\mathcal{A}_d} = \phi \left(\frac{\mathcal{A}_d}{\mathcal{A}}\right)_{cr}^{\frac{2\kappa}{\kappa-1}} \iff \frac{U}{\mathcal{A}} = \phi \left(\frac{\mathcal{A}_d}{\mathcal{A}}\right)_{cr}^{\frac{\kappa+1}{\kappa-1}} \tag{A.2.5}$$

From the energy equation an expression for $\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr}$ can be derived, rewriting the dimensionless energy equation as

$$\frac{2}{\kappa - 1}\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)^2 + \left(\frac{U}{\mathcal{A}_d}\right)^2 = \frac{\kappa + 1}{\kappa - 1}. \tag{A.2.6}$$

Now, putting $\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right) = \left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr}$ and using the continuity equation, this can be written

$$\phi^2 = \left(\frac{\kappa + 1}{\kappa - 1} - \frac{2}{\kappa - 1}\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr}^2\right)\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr}^{\frac{4}{\kappa-1}}, \tag{A.2.7}$$

giving an expression for $\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr}$ to be used in the boundary equation and the sonic limit.

Using the boundary condition, Eq. A.2.5, B and p_{d0} can be written as

$$B = \frac{2\kappa}{\kappa - 1} p_d \left(\frac{\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr}^{\frac{\kappa+1}{\kappa-1}}}{a_d \phi}\right)^{\frac{2\kappa}{\kappa-1}} (u^*)^{\frac{\kappa+1}{\kappa-1}} \tag{A.2.8}$$

and

$$p_{d0} = p_u \left(u^* \frac{\left(\frac{\mathcal{A}}{\mathcal{A}_d}\right)_{cr}^{\frac{\kappa+1}{\kappa-1}}}{a_d \phi}\right)^{\frac{2\kappa}{\kappa-1}} - B u^*, \tag{A.2.9}$$

for the sonic case.

For the subsonic case the boundary function cannot be linearised in the same way since p cannot be explicitly expressed as a function $f(u)$.

A.3 Non-homentropic Inflow

In an engine manifold one can maybe assume that the flow is homentropic at the outflow since there will not be that many vortices decreasing the energy of the flow if the geometry is changing gently. In the inflow case there may be more vortices. Benson has derived boundary equations for non-homentropic inflow in [9]. The flow is assumed to change from the stagnation pressure, p_u isentropically to the pressure in the throat, p_t and then adiabatically to the pressure in the pipe, p. In the subsonic case the pressure p_t is assumed to be equal to the pressure in the pipe.

For subsonic flow the boundary equation is derived using the continuity equation, the energy equation and the equation for the speed of sound, $a^2 = \frac{\kappa p}{\rho}$. Using these equations together with the assumption that $p_t = p$ and that the flow expands isentropically from outside the pipe into the throat, one eventually reach the equation for the subsonic inflow. Starting by writting the continuity equation for $p_t = p$ as, $u_t = \frac{1}{\psi}\frac{\rho}{\rho_t}$ and using that the equation for the speed of sound can be written as

$$\frac{\rho}{\rho_t} = \frac{p}{p_t}\left(\frac{a_t}{a}\right)^2 = \left(\frac{a_t}{a}\right)^2 \tag{A.3.1}$$

the continuity equation becomes

$$u_t = \frac{1}{\psi}\frac{a_t}{a}. \tag{A.3.2}$$

The ellipse of energy is again used, now in the form

$$a_t^2 = a_u^2 - \frac{\kappa - 1}{2}u_t^2. \tag{A.3.3}$$

Combining these equations, the following expression is found;

$$\psi\frac{a_u}{a_t}\left(\frac{2}{\kappa - 1}\left(\left(\frac{a_u}{a_t}\right)^2 - 1\right)\right)^{\frac{1}{2}} = \frac{\left(\frac{u}{a_u}\right)}{\left(\frac{a}{a_u}\right)^2}. \tag{A.3.4}$$

The energy equation can also be written in the form

$$\left(\frac{a}{a_u}\right)^2 = 1 - \frac{\kappa - 1}{2}\left(\frac{u}{a_u}\right)^2, \tag{A.3.5}$$

and

$$\frac{a_u}{a_t} = \left(\frac{p_u}{p_t}\right)^{\frac{\kappa-1}{2\kappa}} = \left(\frac{p_u}{p}\right)^{\frac{\kappa-1}{2\kappa}}. \tag{A.3.6}$$

Inserting these two expressions in Eq. A.3.4 gives the boundary equation

$$\psi\left(\frac{p_u}{p}\right)^{\frac{\kappa-1}{2\kappa}}\left(\frac{2}{\kappa - 1}\left(\left(\frac{p_u}{p}\right)^{\frac{\kappa-1}{\kappa}} - 1\right)\right)^{\frac{1}{2}} = \frac{\frac{u}{a_u}}{1 - \frac{\kappa-1}{2}\left(\frac{u}{a_u}\right)^2}. \tag{A.3.7}$$

From this equation one can find

$$p = p_u \left(\frac{\sqrt{g(u)^2 + 4g(u)} - g(u)}{2} \right)^{\frac{\kappa}{\kappa-1}} = f(u), \tag{A.3.8}$$

with

$$g(u) = \frac{2\psi^2}{\kappa - 1} \left(\frac{a_u^2}{u^2} - \kappa + 1 + \left(\frac{\kappa - 1}{2} \right)^2 \frac{u^2}{a_u^2} \right)^2 \tag{A.3.9}$$

that can be linearised around u^*.

This will give

$$A = -p_u \frac{\kappa}{\kappa - 1} \left(\frac{\sqrt{g(u^*)^2 + 4g(u^*)} - g(u^*)}{2} \right)^{\frac{1}{\kappa-1}} \frac{g'(u^*)}{2} \left(\frac{g(u^*) + 2}{\sqrt{g(u^*)^2 + 4g(u^*)}} - 1 \right), \tag{A.3.10}$$

and p_{u0} can be calculated as $f(u^*)$.

At the sonic boundary $u_t = a_t$ and $p = p_t$. The energy equation is then, using $u_t = a_t$,

$$a_u^2 = a_t + \frac{\kappa - 1}{2} a_t^2 = \frac{\kappa + 1}{2} a_t^2 \iff \frac{a_u}{a_t} = \left(\frac{\kappa + 1}{2} \right)^{\frac{1}{2}}. \tag{A.3.11}$$

Again using $\frac{a_u}{a_t} = \left(\frac{p_u}{p} \right)^{\frac{\kappa-1}{2\kappa}}$ the critical pressure is derived as $p_s = p \left(\frac{\kappa+1}{2} \right)^{\frac{\kappa}{\kappa-1}}$.

When the flow is sonic in the throat the assumption that $p_t = p$ no longer holds. Using $u_t = a_t$ the continuity can now be written, using $\frac{\rho}{\rho_t} = \frac{p}{p_t} \left(\frac{a_t}{a} \right)^2$, as

$$u = \psi \left(\frac{a}{a_t} \right)^2 \left(\frac{p_t}{p} \right) a_t \iff \frac{u}{a_u} = \psi \left(\frac{a}{a_t} \right)^2 \left(\frac{a_u}{a_t} \right) \left(\frac{p_t}{p_u} \right) \left(\frac{p_u}{p} \right). \tag{A.3.12}$$

Using the form of the energy equation given in Eq. A.3.5 and Eq. A.3.11, equivalent to $\frac{p_t}{p_u} = \left(\frac{2}{\kappa+1} \right)^{\frac{\kappa}{\kappa-1}}$ due to the isentropic change of state, and substituting these expressions into the continuity equation the boundary equation for choked flow in the throat is derived as

$$p = p_u \psi \left(\frac{2}{\kappa + 1} \right)^{\frac{\kappa+1}{2(\kappa-1)}} \left(\frac{1 - \frac{\kappa-1}{2} \left(\frac{u}{a_u} \right)^2}{\frac{u}{a_u}} \right) = f(u). \tag{A.3.13}$$

From this p_{u0} can be calculated as $f(u^*)$, and A as $-f'(u^*)$;

$$A = -p_u \psi \left(\frac{2}{\kappa + 1} \right)^{\frac{\kappa+1}{2(\kappa-1)}} \left(-\frac{a_u}{(u^*)^2} - \frac{\kappa - 1}{2a_u} \right). \tag{A.3.14}$$

Appendix B

Derivations

For readers interested in details derivations or verifications of some equations are given here.

B.1 Eq. 2.1.1 – Continuity

The derivation of the continuity equation for non-steady flow, Eq. 2.1.1, is given as in [9] (p. 62).

When deriving the governing equations a pipe-section is considered, with an area change $\frac{dF}{dx}$ over the control-volume that is of length dx. The continuity expresses how the net outflow rate of the volume,

$$\left(\rho + \frac{\partial \rho}{\partial x}dx\right)\left(u + \frac{\partial u}{\partial x}dx\right)\left(F + \frac{dF}{dx}dx\right) - \rho u F,$$

is equal to the rate of decrease in mass inside of the volume,

$$-\frac{\partial}{\partial t}(\rho F dx).$$

The first expression can be expanded and simplified, including only first order terms, as

$$dx\left(\rho u\frac{dF}{dx} + \rho F\frac{\partial u}{\partial x} + uF\frac{\partial \rho}{\partial x}\right) = dx\frac{\partial(\rho u F)}{\partial x}$$

Using this and $-\frac{\partial}{\partial t}(\rho F dx) = -dxF\frac{\partial \rho}{\partial t}$, the continuity equation for non-steady flow is given as

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + \rho\frac{\partial u}{\partial x} + \frac{\rho u}{F}\frac{dF}{dx} = 0.$$

B.2 Eq. 2.1.2 – Momentum

The derivation of the conservation of momentum, Eq. 2.1.2, is given in [9] (p. 63) and same control volume as in the previous section is used. In the momentum equation the sum of the pressure forces,

$$pF - \left(p + \frac{\partial p}{\partial x}dx\right)\left(F + \frac{dF}{dx}dx\right) + p\frac{dF}{dx}dx \approx -\frac{\partial}{\partial x}(pF)dx + p\frac{dF}{dx}dx,$$

and the shear forces,

$$-f\frac{\rho u^2}{2}\pi D dx,$$

is equal to the sum of the rate of change of momentum,

$$\frac{\partial}{\partial t}(\rho F u dx),$$

and the net outflow of momentum from the control surface,

$$\left(\rho + \frac{\partial \rho}{\partial x}dx\right)\left(u + \frac{\partial u}{\partial x}dx\right)^2\left(F + \frac{dF}{dx}dx\right) - \rho u^2 F \approx \frac{\partial}{\partial x}(\rho u^2 F dx).$$

The momentum equation can then be written as, expanding the derivatives,

$$u\left(\frac{\partial \rho}{\partial t} + \rho\frac{\partial u}{\partial x} + \frac{\rho u}{F}\frac{dF}{dx} + u\frac{\partial \rho}{\partial x}\right) + f\frac{\rho u^2}{2}\frac{4}{D} + \frac{\partial p}{\partial x} + \rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} = 0.$$

Identifying the continuity equation this expression simplifies to

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{1}{\rho}\frac{\partial p}{\partial x} + G = 0,$$

with $G = \frac{1}{2}u|u|f\frac{4}{D}$.

B.3 Eq. 2.1.3 – Energy

The energy equation is derived in [9] (p.64), starting from the first law of thermodynamics for the same control volume used above. The first law of thermodynamics states that, with an external shear work equal to zero, the heat transfer rate

$$\dot{Q} = q\rho F dx$$

is equal to the sum of the rate of change of total energy, E, inside the volume

$$\frac{\partial E}{\partial t} = \frac{\partial}{\partial t}\left(\rho F dx\left(\frac{p}{(\kappa - 1)\rho} + \frac{u^2}{2}\right)\right),$$

and the net outflow of stagnation enthalpy,

$$\frac{\partial}{\partial x}\left(\rho F u\left(\frac{p}{(\kappa-1)\rho}+\frac{u^2}{2}+\frac{p}{\rho}\right)\right)dx.$$

The energy equation is then

$$q\rho F dx=\frac{\partial}{\partial t}\left(\rho F dx\left(\frac{p}{(\kappa-1)\rho}+\frac{u^2}{2}\right)\right)+\frac{\partial}{\partial x}\left(\rho F u\left(\frac{p}{(\kappa-1)\rho}+\frac{u^2}{2}+\frac{p}{\rho}\right)\right)dx.$$

The version of the energy equation used in this work is the equation above simplified using the continuity and momentum equation. The full derivation with all steps can be found in [9].

B.4 Eq. 2.4.5 and 2.4.6 – Successions

In Chapter 2 the succession of the velocity states in the homentropic case is given as Eq. 2.4.5 and Eq. 2.4.6. Looking at Fig. 2.2 the slope of the characteristic between a state u_{2j} and the subsequent state u_{2j+1} is

$$C=\frac{p_{2j+1}-p_{2j}}{u_{2j+1}-u_{2j}}.$$

Further the slopes of the approximated boundary equations are

$$A=\frac{p_{2j+1}-p_\infty}{u_\infty-u_{2j+1}}$$

and

$$B=\frac{p_{2j}-p_\infty}{u_{2j}-u_\infty}.$$

Combining the equations for the different slopes gives the succession from u_{2j} to u_{2j+1} as

$$u_{2j+1}=\frac{A+B}{C+A}u_\infty+\frac{C-B}{C+A}u_{2j}\Longleftrightarrow u_{2j+1}=\left(1-\frac{C-B}{C+A}\right)u_\infty+\frac{C-B}{C+A}u_{2j},$$

which is equivalent to the Eq. 2.4.6. In the same manner the Eq. 2.4.5 can be derived starting with

$$-C=\frac{p_{2j+1}-p_{2j+2}}{u_{2j+1}-u_{2j+2}}.$$

Note that A is the slope with a minus sign by definition.

B.5 Eq. 2.4.9 and 2.4.12 – Dynamic Model

The successions can be expressed using only the steady value, derived using $u_0 = 0$ and Eq. 2.4.5 and Eq. 2.4.6. The even states for $\lambda > 0$ are easily shown to be

$$u_{2j} = \left(1 - \lambda^j\right) u_\infty.$$

The time between each of these states is $\frac{2L}{c}$, where c is the velocity of the wave. A time dependent expression corresponding to the discrete model is then

$$u(t) = \left(1 - \lambda^{\frac{tc}{2L}}\right) u_\infty \Longleftrightarrow u(t) = \left(1 - e^{\frac{-t}{\tau}}\right) u_\infty,$$

where $\tau = \frac{-2L}{cln\lambda}$. At time-points $t = \frac{2L}{c} j$ the discrete model will be fulfilled, $u\left(t = \frac{2L}{c} j\right) = u_{2j}$.

In the same manner the successions can be given for $\lambda > 0$ for the odd states as

$$u_{2j+1} = \left(1 - \xi\lambda^j\right) u_\infty.$$

It holds that $u\left(t = \frac{2L}{c}\left(j + \frac{1}{2}\right)\right) = u_{2j+1}$ and the time dependent model is then for this case

$$u(t) = \left(1 - \frac{\xi}{\sqrt{\lambda}} e^{\frac{-t}{\tau}}\right) u_\infty.$$

For $\lambda < 0$ the expressions are

$$u_{2j} = \left(1 - (-|\lambda|)^j\right) u_\infty$$

and

$$u_{2j+1} = \left(1 - \xi(-|\lambda|)^j\right) u_\infty.$$

The time dependent expressions are

$$u(t) = \left(1 - e^{\frac{-t}{\tau}} cos(\omega t)\right) u_\infty,$$

and

$$u(t) = \left(1 - \frac{\xi}{\sqrt{\lambda}} e^{\frac{-t}{\tau}} sin(\omega t)\right) u_\infty.$$

The cosine and sine function ensures that there will be a correct sign inside the parentheses through $\omega = \frac{\pi c}{2L}$ and will become ± 1 for $t = \frac{2L}{c} j$ respectively $t = \frac{2L}{c}\left(j + \frac{1}{2}\right)$.

That the dynamical models, 2.4.9 and 2.4.12, holds can easily be shown differentiating the time dependent expressions and substituting into the equations.

B.6 Eq. 2.4.15 and 2.4.16 – Successions

The successions in the non-homentropic case are somewhat different from the ones in the homentropic case. First of all, here the successions are from one state at the upstream boundary to the next at the upstream boundary, and vice versa, corresponding in the homentropic case to a succession of step-size two.

The derivation starts with four expressions Eq. 2.4.13 and Eq. 2.4.14

$$\begin{aligned} p_{u,j} &= p_{u0} - Au_{u,j}, \\ p_{d,j} &= p_{d0} + Bk_0 u_{d,j}. \end{aligned}$$

that are the approximated boundary conditions, and

$$\frac{1}{C}\frac{dp}{dt} \pm \frac{du}{dt} + \Delta_1 + \Delta_2 \pm \Delta_3 = 0 \tag{B.6.1}$$

that are combinations of the governing equations, as described in Chapter 2. Starting by integrating the upper-sign version of Eq. 2.3.8 over the time it takes for a forward going wave to pass the whole pipe, that is, a wave starting at the upstream end at time t_1 at state $(u_{u,j}, p_{u,j})$, ending at the downstream end at state $(u_{d,j}, p_{d,j})$ at time t_2. This gives

$$p(t_2) - p(t_1) + Cu(t_2) - Cu(t_1) + C(\Delta_1 + \Delta_2 + \Delta_3)(t_2 - t_1) = 0.$$

Using that $t_2 - t_1 = \frac{L}{a+u}$, one gets the expression

$$p_{d,j} - p_{u,j} + Cu_{d,j} - Cu_{u,j} + \alpha_1 = 0, \tag{B.6.2}$$

where $\alpha_1 = \frac{CL}{a+u}(\Delta_1 + \Delta_2 + \Delta_3)$.

The lower-sign of Eq. 2.3.8 is integrated over a backward going wave, from $(u_{d,j}, p_{d,j})$ at time t_2 to $(u_{u,j+1}, p_{u,j+1})$ at time t_3, giving

$$p(t_3) - p(t_3) - Cu(t_3) + Cu(t_2) + C(\Delta_1 + \Delta_2 - \Delta_3)(t_3 - t_2) = 0$$

or

$$p_{u,j+1} - p_{d,j} - Cu_{u,j+1} + Cu_{d,j} + \alpha_2 = 0, \tag{B.6.3}$$

where $\alpha_2 = \frac{CL}{a-u}(\Delta_1 + \Delta_2 - \Delta_3)$

Now, the successions at the upstream and downstream end can be found by combining Eq. 2.4.13, Eq. 2.4.14, Eq. B.6.2 and Eq. B.6.3. First the expression for the upstream end is sought, starting with the expression

$$\begin{aligned} &(C+B)(C+A)u_{u,j+1} - (C-B)(C-A)u_{u,j} - 2C(p_{u0} - p_{d0}) \\ &= C(Cu_{u,j+1} - Cu_{u,j} + Au_{u,j+1} + Au_{u,j} - 2(p_{u0} - p_{d0})) \\ &\quad + B(Cu_{u,j+1} + Cu_{u,j} + Au_{u,j+1} - Au_{u,j}). \end{aligned} \tag{B.6.4}$$

Now, two expressions combining Eq. 2.4.13 and Eq. 2.4.14 are given as

$$p_{u,j} - p_{d,j} = p_{u0} - p_{d0} - Au_{u,j} - Bu_{d,j}$$

and

$$p_{d,j} - p_{u,j+1} = -p_{u0} + p_{d0} + Au_{u,j+1} + Bu_{d,j}.$$

These can be combined in the following ways

$$\begin{aligned}(p_{u,j} - p_{d,j}) - (p_{d,j} - p_{u,j+1}) &= 2(p_{u0} - p_{d0}) - Au_{u,j} - Au_{u,j+1} - 2Bu_{d,j} \\ &\iff \\ Au_{u,j} + Au_{u,j+1} - 2(p_{u0} - p_{d0}) &= -2Bu_{d,j} - (p_{u,j} - p_{d,j}) + (p_{d,j} - p_{u,j+1}),\end{aligned}$$

and

$$(p_{u,j} - p_{d,j}) + (p_{d,j} - p_{u,j+1}) = Au_{u,j+1} - Au_{u,j}.$$

Now, these expressions can be inserted into Eq. B.6.4 giving

$$\begin{aligned}&(C+B)(C+A)u_{u,j+1} - (C-B)(C-A)u_{u,j} - 2C(p_{u0} - p_{d0}) \\ &= C(Cu_{u,j+1} - Cu_{u,j} - (p_{u,j} - p_{d,j}) + (p_{d,j} - p_{u,j+1})) \\ &\quad + B(Cu_{u,j+1} + Cu_{u,j} + (p_{u,j} - p_{d,j}) + (p_{d,j} - p_{u,j+1}) - 2Cu_{d,j}) \\ \iff& \\ u_{u,i+1} &= \frac{C-B}{C+B}\frac{C-A}{C+A}u_{u,i} + \frac{2}{C+B}\frac{C}{C+A}(p_{u0} - p_{d0}) - \frac{1}{C+A}\left(\alpha_1\frac{C-B}{C+B} - \alpha_2\right).\end{aligned}$$

In the same manner one can get the succession for the downstream end as

$$u_{d,i+1} = \frac{C-B}{C+B}\frac{C-A}{C+A}u_{d,i} + \frac{2}{C+B}\frac{C}{C+A}(p_{u0} - p_{d0}) - \frac{1}{C+B}\left(\alpha_1 - \alpha_2\frac{C-A}{C+A}\right).$$

To get the steady values upstream and downstream set $u_{\cdot,i+1} = u_{\cdot,i} = u_{\cdot,\infty}$ and solve for $u_{\cdot,\infty}$.

Appendix C

Setup of Tests

Here most of the setups of the simulated configurations discussed in 3.2 are presented. In the following explicit Euler was used, the length of the pipe was set to 1 m, the diameter to 5 cm and the friction factor was set to 0.05. The media used was from Modelon's library the ideal fluid *Fast Air, fixed composition, linear cp, 20-600 °C*. The temperature where 25 °C in both ends and there was no heat-transfer.

Table C.1: Table over the setup in configurations simulated and shown in the figures indicated under Case.

Case	p_u [bar]	ψ	p_d [bar]	ϕ	time-step [ms]	Run-time [ms]	CPU-time [ms]
Fig. 3.4	ramp: 1.5 to 2	0.7 0.3	1.5	0.7	1.0	50	15 18
Fig. 3.5	$1.7 + 0.1sin(\pi t)$	0.7	1.5	0.7	1.0	3000	463
Fig. 3.6	$1.7 + 0.1sin(10\pi t)$	0.7	1.5	0.7	1.0	3000	505
Fig. 3.7	2	0.7	1.5	$0.5 + 0.5sin(\pi t)$	0.2	3000	2430
Fig. 3.8	$1.4 + 0.2sin(\pi t)$	0.7	1.6	0.7	1.0	3000	600
Fig. 3.9	$1.5 + 0.2sin(\pi t)$	0.7	1.5	0.7	1.0	3000	542
Fig. 3.10	$1.5 + 0.2sin(10\pi t)$	0.7	1.5	0.7	1.0	3000	542
Fig. 3.13	$1.5 + 0.2sin(10\pi t + 10)$	$0.7 + 0.2sin(5\pi t + 5)$	$1.5 - 0.1sin(\pi t + 90)$	$0.5 + sin(\pi t + 45)$	0.2	3000	2370

Appendix D

Modelica Code

```
model QPM_pipe
  replaceable package Medium =
    Modelon.Media.PreDefined.IdealGases.FastDryAir constrainedby
    Modelon.Media.Interfaces.GenericMedium "Fluid model"
  annotation(choicesAllMatching);

  parameter Real timeStep = 0.0001 "Time step size";
  Real x( start = 0) "Where is the wave";
  Real xMod "x mod 2L";

 parameter Boolean useEventLambda = false
    "true if one want to have an event each time lambda changes
     sign";

  // Geometry
  parameter Modelica.SIunits.Length L = 1 "Length of pipe";
  parameter Modelica.SIunits.Diameter D = 0.05 "Diameter of pipe";
  parameter Modelica.SIunits.Area F = Modelica.Constants.pi*D^2/4
    "Pipe area upstream";
  parameter Real dFdx = 0 "Change of F along x";

  Real phi_real(max=1, min=0) "Throat area downstream / Area pipe";
  Real psi_real(max=1, min=0) "Throat area upstream / Area pipe";
  Real phi "Limited area ratio phi";
  Real psi "Limited area ratio psi";

 parameter Boolean static_upstream = false;
 parameter Boolean static_downstream = false;

  // Initially
  parameter Modelica.SIunits.Pressure p_init = 1e5
    "Initial pressure";

  // Medium
  Medium.ThermodynamicState state_u
    "State at upstream end non-reversed flow" ;
  Medium.ThermodynamicState state_u_rev
```

```
    "State at upstream end reversed flow";
Medium.ThermodynamicState state_d
  "State at downstream end non-reversed flow";
Medium.ThermodynamicState state_d_rev
  "State at downstream end reversed flow";

Modelica.SIunits.SpecificHeatCapacity Cp_u
  "Specific heat upstream";
Modelica.SIunits.SpecificHeatCapacity Cv_u
  "Specific heat upstream";
Real k_u "Ratio of specific heats upstream";

Modelica.SIunits.SpecificHeatCapacity Cp_d
  "Specific heat downstream";
Modelica.SIunits.SpecificHeatCapacity Cv_d
  "Specific heat downstream";
Real k_d "Ratio of specific heats downstream";

Modelica.SIunits.Pressure p_u(start = p_init)
  "Pressure upstream";
Modelica.SIunits.Pressure p_d(start = p_init)
  "Pressure downstream";
Modelica.SIunits.Velocity a_u(start = 400)
  "Speed of sound upstream";
Modelica.SIunits.Velocity a_d(start = 400)
  "Speed of sound downstr.";
Modelica.SIunits.Density rho_u "Density upstream";
Modelica.SIunits.Density rho_d "Density downstream";
Modelica.SIunits.MassFlowRate m_flow_middle;

Real G "Friction force/unit mass";
parameter Real f = 0.05 "Friction factor";
Real q "Heat flow rate /unit mass/unit time";
Modelica.SIunits.HeatFlowRate Q_flow "Net heat energy transfer";

// QPM-variables
Real lambda(start = 1) "(C-B)/(C+B)*(C-A)/(C+A)";
Real A(start = 1) "Appr. slope of upstream b.c.";
Real p_u0(start = p_init) "Pressure at u=0, upstream b.c";
Real B(start = 1) "Appr. slope of downstream b.c.";
Real p_d0(start = p_init) "Pressure at u=0, downstream b.c";
Real C "dp/du, slope in state diagram";
Modelica.SIunits.Velocity c(start = 400) "Velocity of wave";
Modelica.SIunits.Velocity u_inf(start = 400) "Steady value of u";

Modelica.SIunits.Velocity u(start = 1) "Midpoint velocity";
Modelica.SIunits.Acceleration u_temp "Midpoint acceleration";

Real u_uinf(start = 300) "Steady state upstream end";
Real u_dinf(start = 300) "Steady state downstream end";
Real alpha_1(start = 1) "Non-homentropic term";
Real alpha_2(start = 1) "Non-homentropic term";
Real D1(start = 1) "Non-homentropic term";
Real D2(start = 1) "Non-homentropic term";
Real D3(start = 1) "Non-homentropic term";
Real u_int(start = 300) "Homentropic steady state";
```

```
  Real tau(start = 2*timeStep) "Time constant";
  Real omega(start = 3.14/(2*timeStep)) "Pulsation constant";

  Modelica.SIunits.Pressure p_BCu(start=p_init)
    "Pressure inside pipe at upstream end";
  Modelica.SIunits.Pressure p_BCd(start=p_init)
    "Pressure inside pipe at downstream end";

  Real p_cr(start = 1) "Critical pressure ratio for outflow";
  Real p_cr_rev(start = 1) "Cr. pr. ratio for outflow if rev. flow";
  parameter Real dp = 1e3 "spliceFunction parameter";

  Real p1(start=p_init,nominal = 1e5,min = 0) "Linearisation var.";
  Real p2(start=p_init,nominal = 1e5, min= 0) "Linearisation var.";
  Real p_subOut(start=p_init, nominal = 1e5, min = 0) "Lin. var.";
  Real du(start=0.1) "Linearisation variable";
  Real p1_temp(start = 1) "Linearisation help variable";
  Real p2_temp(start = 1) "Linearisation help variable";
  Real p_subOut_temp(start = 1) "Linearisation help variable";

  Real psu "Critical pressure upstream end, non-reversed flow";
  Real psu_rev "Critical pressure upstream end, reversed flow";
  Real psd "Critical pressure downstream end, non-reversed flow";
  Real psd_rev "Critical pressure downstream end, reversed flow";


  Modelon.ThermoFluid.Interfaces.FlowPort port_u(
    redeclare package Medium=Medium)
    a;
  Modelon.ThermoFluid.Interfaces.FlowPort port_d(
    redeclare package Medium=Medium)
    a;

  Modelica.Blocks.Interfaces.RealInput port_psi
    a;
  Modelica.Blocks.Interfaces.RealInput port_phi
    a;
  Modelon.ThermoFluid.Interfaces.FlowHeatPort heatPort
    a;

equation
  //Avoid fully closed and fully open ends
  phi = min(max(phi_real,0.01),0.99);
  psi = min(max(psi_real,0.01),0.99);

  // Dynamic model
  if p_u>p_d and static_upstream and phi==1 or
    p_d>p_u and static_downstream and psi==1 then
    der(u)=(p_u-p_d)*c/(C*L)-G;
    der(u_temp)=0;
  else
      if useEventLambda then
        if (lambda > 0) then
            der(u) = (u_inf - u) / tau;
            der(u_temp) = -u_temp/tau;
```

```
        else
            der(u_temp)= -2/tau * der(u) +
            (1/tau^2 + omega^2)*(u_inf - u);
            der(u)=u_temp;
        end if;
       else
        der(u) = noEvent(
        if lambda > 0 then
            (u_inf - u) / tau
        else
            u_temp);
        der(u_temp) = noEvent(
        if lambda > 0 then
            -u_temp/tau
        else
            -2/tau*der(u) + (1/tau^2+omega^2)*(u_inf-u));
     end if;
end if;

 tau = -2*L/(c * log(min(0.999,abs(lambda))));
omega = Modelica.Constants.pi * c /(2*L);

lambda = (C-abs(B))/(C+abs(B))*(C-abs(A))/(C+abs(A));
C = k_u*p_u/a_u;

// Follow the wave
der(x) = c;
xMod = noEvent(mod(x,2*L));
c = noEvent(
  if xMod > 0 and xMod < L then
      a_u+u
  else
      a_u-u);

// Help variables
u_uinf=u_int-(C+abs(B))/max(0.001,(abs(A)+abs(B)))*
0.5/C*(alpha_1*(C-abs(B))/max(0.001,(C+abs(B)))-alpha_2);
u_dinf=u_int-(C+abs(A))/max(0.001,(abs(A)+abs(B)))*
0.5/C*(alpha_1-alpha_2*(C-abs(A))/(C+abs(A)));

D1 = -(k_u-1)*(q+abs(u)*G)/a_u;
D2 = (a_u*abs(u)/F)*dFdx;
D3 = G;

alpha_1 = (D1+D2+D3)/(a_u+abs(u))*C*L;
alpha_2 = -1*(-D1-D2+D3)/(a_u-abs(u))*C*L;

u_int = if p_u > p_d then
  (p_u0-p_d0)/max(0.001,(abs(A)+abs(B)))
else
  -1* (p_u0-p_d0)/max(0.001,(abs(A)+abs(B)));

u_inf = if p_u > p_d then
  0.5*(u_uinf+u_dinf)
else
 - 0.5*(u_uinf+u_dinf);
```

```
// Critical pressure (used in outflow B.C)
 p_cr = 1 + (k_d-1)/4*sqrt(1-phi^2);
 p_cr_rev = 1 +  (k_u-1)/4*sqrt(1-psi^2);

// Boundary conditions
psu = delay(p_BCu,timeStep)*((k_u + 1)/2)^(k_u/(k_u - 1));
psu_rev =  p_u*p_cr_rev^(2*k_u/(k_u-1));
psd = p_d*p_cr^(2*k_d/(k_d-1));
psd_rev = delay(p_BCd,timeStep)*((k_d + 1)/2)^(k_d/(k_d - 1));



// Subsonic outflow
 du = max(abs(u)/100,1);
 max(abs(u),0.1) + du = (
   if p_u > p_d  then
      sign(p1_temp)*(-1)*(min(abs(p1_temp),(psd-p_d))-
      max(abs(p1_temp),(psd-p_d))+abs(p1_temp)-(psd-p_d))/
      (2*max((max(abs(p1_temp),(psd-p_d))-
      min(abs(p1_temp),(psd-p_d))),1e-8))*
      sqrt(abs(2*a_d^2/(k_d - 1)*max(1e-8,(
      max(1, (abs(p1_temp)+p_d)/p_d)^((k_d - 1)/k_d) - 1)/
      (1/phi^2*max(1, (abs(p1_temp)+p_d)/p_d)^(2/k_d) - 1)))) +
      sign(p1_temp)*(-1)*(min(abs(p1_temp),(psd-p_d))-
      max(abs(p1_temp),(psd-p_d))-abs(p1_temp)+(psd-p_d))/
      (2*max((max(abs(p1_temp),(psd-p_d))-
      min(abs(p1_temp),(psd-p_d))),1e-4))*
      a_d*((abs(p1_temp)+p_d)/p_d)^((k_d-1)/(2*k_d))*
      psi*p_cr^((-1*(k_d+1))/(k_d-1))
  else
      sign(p1_temp)*(-1)*(min(abs(p1_temp),(psu_rev-p_u))-
      max(abs(p1_temp),(psu_rev-p_u))+abs(p1_temp)-(psu_rev-p_u))/
      (2*max((max(abs(p1_temp),(psu_rev-p_u))-
      min(abs(p1_temp),(psu_rev-p_u))),1e-8))*
      sqrt(abs(2*a_u^2/(k_u - 1)*max(1e-8,(
      max(1,(abs(p1_temp)+p_u)/p_u)^((k_u - 1)/k_u)-1)/
      (1/psi^2*max(1, (abs(p1_temp)+p_u)/p_u)^(2/k_u)-1)))) +
      sign(p1_temp)*(-1)*(min(abs(p1_temp),(psu_rev-p_u))-
      max(abs(p1_temp),(psu_rev-p_u))-abs(p1_temp)+(psu_rev-p_u))/
      (2*max((max(abs(p1_temp),(psu_rev-p_u))-
      min(abs(p1_temp),(psu_rev-p_u))),1e-4))*
      a_u*((abs(p1_temp)+p_u)/p_u)^((k_u-1)/(2*k_u))*
      psi*p_cr_rev^((-1*(k_u+1))/(k_u-1)));

  max( abs(u),0.1) - du = (
    if p_u > p_d  then
      sign(p2_temp)*(-1)*(min(abs(p2_temp),(psd-p_d))-
      max(abs(p2_temp),(psd-p_d))+abs(p2_temp)-(psd-p_d))/
      (2*max((max(abs(p2_temp),(psd-p_d))-
      min(abs(p2_temp),(psd-p_d))),1e-8))*
      sqrt(abs(2*a_d^2/(k_d - 1)*max(1e-8,(
      max(1, (abs(p2_temp)+p_d)/p_d)^((k_d - 1)/k_d) - 1)/
      (1/phi^2*max(1, (abs(p2_temp)+p_d)/p_d)^(2/k_d)-1)))) +
      sign(p2_temp)*(-1)*(min(abs(p2_temp),(psd-p_d))-
      max(abs(p2_temp),(psd-p_d))-abs(p2_temp)+(psd-p_d))/
```

```
        (2*max((max(abs(p2_temp),(psd-p_d))-
        min(abs(p2_temp),(psd-p_d))),1e-4))*
        a_d*((abs(p2_temp)+p_d)/p_d)^((k_d-1)/(2*k_d))*
        phi*p_cr^((-1*(k_d+1))/(k_d-1))
    else
        sign(p2_temp)*(-1)*(min(abs(p2_temp),(psu_rev-p_u))-
        max(abs(p2_temp),(psu_rev-p_u))+abs(p2_temp)-(psu_rev-p_u))/
        (2*max((max(abs(p2_temp),(psu_rev-p_u))-
        min(abs(p2_temp),(psu_rev-p_u))),1e-8))*
        sqrt(abs(2*a_u^2/(k_u - 1)*
        max(1e-8,(max(1, (abs(p2_temp)+p_u)/p_u)^((k_u - 1)/k_u)-1)/
        (1/psi^2*max(1, (abs(p2_temp)+p_u)/p_u)^(2/k_u) - 1)))) +
        sign(p2_temp)*(-1)*(min(abs(p2_temp),(psu_rev-p_u))-
        max(abs(p2_temp),(psu_rev-p_u))-abs(p2_temp)+(psu_rev-p_u))/
        (2*max((max(abs(p2_temp),(psu_rev-p_u))-
        min(abs(p2_temp),(psu_rev-p_u))),1e-4))*
        a_u*((abs(p2_temp)+p_u)/p_u)^((k_u-1)/(2*k_u))*
        psi*p_cr_rev^((-1*(k_u+1))/(k_u-1)));

    max(abs(u),0.1) = (
    if p_u > p_d  then
        sign(p_subOut_temp)*(-1)*(min(abs(p_subOut_temp),(psd-p_d))
        max(abs(p_subOut_temp),(psd-p_d))+abs(p_subOut_temp)-
        (psd-p_d))/(2*max((max(abs(p_subOut_temp),(psd-p_d))-
        min(abs(p_subOut_temp),(psd-p_d))),1e-8))*
        sqrt(abs(2*a_d^2/(k_d - 1)*max(1e-8,(
        max(1,(abs(p_subOut_temp)+p_d)/p_d)^((k_d-1)/k_d)-1)/
        (1/phi^2*max(1, (abs(p_subOut_temp)+p_d)/p_d)^(2/k_d)-1))))+
        sign(p_subOut_temp)*(-1)*(min(abs(p_subOut_temp),(psd-p_d))-
        max(abs(p_subOut_temp),(psd-p_d))-abs(p_subOut_temp)+
        (psd-p_d))/(2*max((max(abs(p_subOut_temp),(psd-p_d))-
        min(abs(p_subOut_temp),(psd-p_d)))1e-4))*
        a_d*((abs(p_subOut_temp)+p_d)/p_d)^((k_d-1)/(2*k_d))*
        phi*p_cr^((-1*(k_d+1))/(k_d-1))
    else
        sign(p_subOut_temp)*(-1)*
        (min(abs(p_subOut_temp),(psu_rev-p_u))-
        max(abs(p_subOut_temp),(psu_rev-p_u))+
        abs(p_subOut_temp)-(psu_rev-p_u))
        /(2*max((max(abs(p_subOut_temp),(psu_rev-p_u))-
        min(abs(p_subOut_temp),(psu_rev-p_u))),1e-8))*
        sqrt(abs(2*a_u^2/(k_u - 1)*max(1e-8,(
        max(1, (abs(p_subOut_temp)+p_u)/p_u)^((k_u - 1)/k_u) - 1)/
        (1/psi^2*max(1,(abs(p_subOut_temp)+p_u)/p_u)^(2/k_u)-1))))+
        sign(p_subOut_temp)*(-1)*
        (min(abs(p_subOut_temp),(psu_rev-p_u))-
        max(abs(p_subOut_temp),(psu_rev-p_u))-
        abs(p_subOut_temp)+(psu_rev-p_u))
        /(2*max((max(abs(p_subOut_temp),(psu_rev-p_u))-
        min(abs(p_subOut_temp),(psu_rev-p_u)))1e-4))*
        a_u*((abs(p_subOut_temp)+p_u)/p_u)^((k_u-1)/(2*k_u))*
        psi*p_cr_rev^((-1*(k_u+1))/(k_u-1)));

    p1 = (
    if p_u > p_d then
        abs(p1_temp) + p_d
```

```
    else
        abs(p1_temp) + p_u);
    p2 = (
    if p_u > p_d then
        abs(p2_temp) + p_d
    else
        abs(p2_temp) + p_u);
    p_subOut = (
    if p_u > p_d then
        abs(p_subOut_temp) + p_d
    else
        abs(p_subOut_temp)+p_u);

  A = (
  if p_u > p_d then
 if static_upstream then
    0
    else
    spliceFunction(
    -1*p_u*k_u/(k_u-1)*(0.5*(sqrt(functions.g(abs(u),k_u,a_u,psi)^2+
        4*functions.g(abs(u),k_u,a_u,psi)) -
        functions.g(abs(u),k_u,a_u,psi)))^(1/(k_u - 1))*
        0.5*functions.dgdu(abs(u),k_u,a_u,psi)*
        ((functions.g(abs(u),k_u,a_u,psi) + 2)/
        sqrt(functions.g(abs(u),k_u,a_u,psi)^2 +
        4*functions.g(abs(u),k_u,a_u,psi)) - 1),
    -1*p_u*psi*(2/(k_u + 1))^((k_u + 1)/(2*k_u - 2))*
        (-a_u/(abs(u)^2) - (k_u - 1)/(2*a_u)),
    psu-p_u,
    dp)
  else
    spliceFunction(
    -1* (p1-p2)/(2*du),
    -1*2*k_u/(k_u-1)*(p_cr_rev^((k_u+1)/(k_u-1))/
        (a_u*psi))^(2*k_u/(k_u-1))*abs(u)^((k_u+1)/(k_u-1))*p_u,
    psu_rev-delay(p_BCu,timeStep),
    dp));

  p_u0 = (
  if p_u > p_d then
if static_upstream then
    p_u
    else
    spliceFunction(
    A*abs(u)+p_u*(0.5*(sqrt(functions.g(abs(u),k_u,a_u,psi)^2 +
        4*functions.g(abs(u),k_u,a_u,psi)) -
        functions.g(abs(u),k_u,a_u,psi)))^(k_u/(k_u - 1)),
    A*abs(u)+p_u*psi*(2/(k_u + 1))^((k_u+1)/(2*k_u-2))*(a_u/abs(u)-
        (k_u - 1)/2*abs(u)/a_u),
    psu- p_u,
    dp)
  else
    spliceFunction(
    A*abs(u) + p_subOut,
    A*abs(u) + (p_cr_rev^((k_u+1)/(k_u-1))*abs(u)/
        (a_u*psi))^(2*k_u/(k_u-1))*p_u,
```

```
    psu_rev-delay(p_BCu,timeStep),
    dp));

  B = (
  if p_u > p_d then
    spliceFunction(
    (p1-p2)/(2*du),
    2*k_d/(k_d-1)*(p_cr^((k_d+1)/(k_d-1))/(a_d*phi))^
        (2*k_d/(k_d-1))*abs(u)^((k_d+1)/(k_d-1))*p_d,
    psd-delay(p_BCd,timeStep),
    dp)
  else
if static_downstream then
    0 else
    spliceFunction(
    p_d*k_d/(k_d - 1)*(0.5*(sqrt(functions.g(abs(u),k_d,a_d,phi)^2 +
        4*functions.g(abs(u),k_d,a_d,phi)) -
        functions.g(abs(u),k_d,a_d,phi)))^(1/(k_d - 1))*
        0.5*functions.dgdu(abs(u),k_d,a_d,phi)*
        ((functions.g(abs(u),k_d,a_d,phi) + 2)/
        sqrt(functions.g(abs(u),k_d,a_d,phi)^2 +
        4*functions.g(abs(u),k_d,a_d,phi)) - 1),
    p_d*phi*(2/(k_d + 1))^((k_d + 1)/(2*k_d - 2))*
        (-a_d/(abs(u)^2) - (k_d - 1)/(2*a_d)),
    psd_rev-p_d,
    dp));

  p_d0 = (
  if p_u > p_d then
    spliceFunction(
    -B*abs(u) + p_subOut,
    -B*abs(u) + (p_cr^((k_d+1)/(k_d-1))*
    abs(u)/(a_d*phi))^(2*k_d/(k_d-1))*p_d,
    psd-delay(p_BCd,timeStep),
    dp)
  else
if static_downstream then
      p_d else
    spliceFunction(
    -B*abs(u) + p_d*(0.5*(sqrt(functions.g(abs(u),k_d,a_d,phi)^2 +
    4*functions.g(abs(u),k_d,a_d,phi)) -
    functions.g(abs(u),k_d,a_d,phi)))^(k_d/(k_d - 1)),
    -B*abs(u) + p_d*phi*(2/(k_d + 1))^((k_d + 1)/(2*k_d - 2))*
        (a_d/abs(u) - (k_d - 1)/2*abs(u)/a_d),
    psd_rev-p_d,
    dp));

     p_BCu = (
  if p_u > p_d then
if p_u > p_d then
    if static_upstream then
    p_u
    else
    spliceFunction(
    p_u*(0.5*(sqrt(functions.g(abs(u),k_u,a_u,psi)^2 +
        4*functions.g(abs(u),k_u,a_u,psi)) -
```

```
        functions.g(abs(u),k_u,a_u,psi)))^(k_u/(k_u - 1)),
    p_u*psi*(2/(k_u + 1))^((k_u + 1)/(2*k_u - 2))*
        (a_u/abs(u) - (k_u - 1)/2*abs(u)/a_u),
    psu-p_u,
    dp)
  else
    spliceFunction(
    p_subOut,
    (p_cr_rev^((k_u+1)/(k_u-1))*
        abs(u)/(a_u*psi))^(2*k_u/(k_u-1))*p_u,
    psu_rev-delay(p_BCu,timeStep),
    dp));

     p_BCd = (
  if p_u > p_d then
    spliceFunction(
    p_subOut,
    (p_cr^((k_d+1)/(k_d-1))*abs(u)/(a_d*phi))^(2*k_d/(k_d-1))*p_d,
    psd-delay(p_BCd,timeStep),
    dp)
  else
if static_downstream then
      p_d else
    spliceFunction(
     p_d*(0.5*(sqrt(functions.g(abs(u),k_d,a_d,phi)^2 +
        4*functions.g(abs(u),k_d,a_d,phi)) -
        functions.g(abs(u),k_d,a_d,phi)))^(k_d/(k_d - 1)),
    p_d*phi*(2/(k_d + 1))^((k_d + 1)/(2*k_d - 2))*
        (a_d/abs(u) - (k_d - 1)/2*abs(u)/a_d),
    psd_rev-p_d,
    dp));

  // Medium

  state_u_rev = Medium.setState_phX(
    p_u, port_u.h_outflow,port_u.X_outflow);
  state_u = Medium.setState_phX(
    p_u, inStream(port_u.h_outflow), inStream(port_u.X_outflow));
  state_d_rev = Medium.setState_phX(
    p_d, inStream(port_d.h_outflow), inStream(port_d.X_outflow));
  state_d = Medium.setState_phX(
    p_d, port_d.h_outflow,port_d.X_outflow);

  k_u =  Cp_u/Cv_u;
  Cp_u = if u > 0 then
    Medium.specificHeatCapacityCp(state_u)
  else
    Medium.specificHeatCapacityCp(state_u_rev);
  Cv_u = if u > 0 then
    Medium.specificHeatCapacityCv(state_u)
  else
    Medium.specificHeatCapacityCv(state_u_rev);
  a_u = if u > 0 then
    Medium.velocityOfSound(state_u)
  else
    Medium.velocityOfSound(state_u_rev);
```

```
  rho_u =if u > 0 then
    Medium.density(state_u)
  else
    Medium.density(state_u_rev);

  k_d =  Cp_d/Cv_d;
  Cp_d = if u > 0 then
    Medium.specificHeatCapacityCp(state_d)
  else
    Medium.specificHeatCapacityCp(state_d_rev);
  Cv_d = if u > 0 then
    Medium.specificHeatCapacityCv(state_d)
  else
    Medium.specificHeatCapacityCv(state_d_rev);
  a_d = if u > 0 then
    Medium.velocityOfSound(state_d)
  else
    Medium.velocityOfSound(state_d_rev);
  rho_d =if u > 0 then
    Medium.density(state_d)
  else
    Medium.density(state_d_rev);

  G = f*abs(u)*abs(u)*2/D;
  q = Q_flow/(rho_u*F*L);

  //Connect
  heatPort.T = if u > 0 then
    (Medium.temperature(state_u)+ Medium.temperature(state_d))/2
  else
    (Medium.temperature(state_u_rev)+
     Medium.temperature(state_d_rev))/2;
  heatPort.Q_flow = Q_flow;

  phi_real = port_phi;
  psi_real = port_psi;

  m_flow_middle= u *(F+0.5*L*dFdx)*(rho_d+rho_u)/2;
  port_d.m_flow = - m_flow_middle;
  port_u.m_flow = m_flow_middle;
  p_d = port_d.p;
  p_u = port_u.p;

  port_u.h_outflow = -sign(port_u.m_flow)*(port_d.m_flow *
    inStream(port_d.h_outflow)-Q_flow)/max(abs(port_u.m_flow),0.00001);
  port_d.h_outflow = -sign(port_d.m_flow)*(port_u.m_flow *
    inStream(port_u.h_outflow)-Q_flow)/max(abs(port_d.m_flow),0.00001);
  port_u.X_outflow = inStream(port_d.X_outflow);
  port_d.X_outflow = inStream(port_u.X_outflow);
  port_u.C_outflow = inStream(port_d.C_outflow);
  port_d.C_outflow = inStream(port_u.C_outflow);

  a
end QPM_pipe;
```

```
package functions

  function g
    input Real u;
    input Real k;
    input Real a;
    input Real F_change;

    output Real gu;
  algorithm
    gu :=2*F_change^2/(k - 1)*((a/max(u,0.0001))^2 -
        k+1+((k-1)/2)^2*(u/a)^2);
  end g;

  function dgdu
     input Real u;
     input Real k;
     input Real a;
     input Real F_change;

     output Real dgu;
  algorithm
    dgu := 2*F_change^2/(k-1)*(-2*a^2/(max(u,0.01)^3)+
        (k-1)^2/4*2*max(u,0.01)/(a^2));
  end dgdu;
end functions;
```