Master's thesis

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

Jonas Tveit Hinna

# Modelica model of transient pipe flow in hydraulic laboratory systems using the method of characteristics

Master's thesis in Energy and Environmental Engineering
Supervisor: Pål-Tore Selbo Storli
June 2021

NTNU
Norwegian University of Science and Technology

WATERPOWER LABORATORY NTNU

Jonas Tveit Hinna

# Modelica model of transient pipe flow in hydraulic laboratory systems using the method of characteristics

Master's thesis in Energy and Environmental Engineering
Supervisor: Pål-Tore Selbo Storli
June 2021

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

NTNU

Kunnskap for en bedre verden

# Master`s Agreement / Main Thesis Agreement

| | |
|---|---|
| **Faculty** | **Faculty of Engineering** |
| **Institute** | **Department of Energy and Process Engineering** |
| **Programme Code** | **MTENERG** |
| **Course Code** | **TEP4900** |

## Personal Information

| | |
|---|---|
| **Surname, First Name** | **Hinna, Jonas Tveit** |
| **Date of Birth** | **22.07.1996** |
| **Email** | **jonasthi@stud.ntnu.no** |

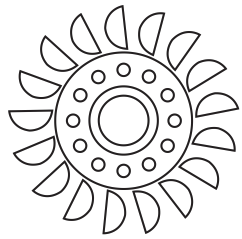
## Supervision and Co-authors

| | |
|---|---|
| **Supervisor** | **Pål-Tore Selbo Storli** |
| **Co-supervisors (if applicable)** | |
| **Co-authors (if applicable)** | |

## The Master`s thesis

| | |
|---|---|
| **Starting Date** | **11.01.2021** |
| **Submission Deadline** | **11.06.2021** |
| **Thesis Working Title** | **Modelica model of the Waterpower Laboratory** |
| **Problem Description** | **Laboratory measurements of properties of hydraulic turbine models are very useful in many aspects. However, when dynamical properties are of interest the measurements are not as useful because the hydraulic system in a lab significantly differs from the real powerplant system. Furthermore, the interpretation of measurements is more difficult as one sometimes doesn't know if what is seen has origin in the hydraulic system or in the turbine model itself. The hydraulic system in the Waterpower laboratory is rather complex, at it would be useful to have a model of this which can be used to simulate the behaviour of the system. This would enable a better understanding of the turbine model characteristics as the system dynamic could be removed from the interpretation of results. Furthermore, an ongoing work on coupling the physical system to a emulsion of a grid in the NTNU Smartgrid lab will benefit from a model of the Waterpower lab being available. A correct representation of the turbine physics is also needed, made available in a mathematical model of a turbine. The existing** |

| | |
|---|---|
| | **models are known have too many simplifications, and a new model must be made. It is unlikely that this is possible within the scope of a single master thesis, but the work needs to be initiated. The student shall make elements in the Modelica language which enable a future system to be simulated which include the effect of pressure wave propagations. The more elements, the better. Furthermore, a new mathematical model of a turbine should be made, based on physics, which also enable pressure pulsations to pass through the element. The following tasks are to be considered: 1. Literature review of system dynamics, the Modelica language and turbine models 2. Make elements in Modelica which enable pressure wave propagation, starting from open ended pipes to bends, conical sections and pipes with closed ends. 3. Make a turbine model which includes all losses, dynamics and pressure propagation.** |



WATERPOWER LABORATORY

NTNU

# *Abstract*

Hydraulic laboratory systems often house complex experimental setups. Measurements at certain locations, e.g. turbine inlet, may be affected by the hydraulic system and the characteristics of the upstream flow. It is therefore desirable to have sufficient modelling tools which can be used to simulate the dynamic behaviour of such systems.

A programming library named *OpenWPL* has been created using the method of characteristics. It was created in the open-source programming language Modelica using the software OpenModelica. Simulations highlighting the utility of *OpenWPL* have been performed, and the results have been compared to benchmarking cases.

It was revealed that the method of characteristics was successfully implemented into the pipe component of the library, both with and without friction. However, it was also revealed that the boundary conditions represented an unidentified source of error when comparing the results to the benchmarking cases. These errors were most prevalent for a low resolution of steps in space. Simultaneously, it also became evident that there was an issue when connecting pipes in series within the library. It is unknown if this is related to the errors within the boundary conditions.

*OpenWPL* is currently not capable of revealing the system dynamics of hydraulic laboratory systems. Rather, the library represents a good foundation that can be further developed into a more complex and full-fledged library in the future.

# *Sammendrag*

Eksperimentelle oppsett i hydrauliske laboratoriesystemer er ofte avanserte. Målinger, f.eks. ved innløpet til turbiner, kan bli påvirket av det hydrauliske oppsettet og av karakteristikken til strømningen oppstrøms i systemet. Det er derfor ønskelig å ha tilstrekkelige modelleringsverktøy som kan brukes til å simulere de dynamiske effektene i slike systemer.

Et programmeringsbibliotek kalt *OpenWPL* har blitt bygget ved å bruke karakteristikkmetoden. Det ble laget i programmeringsspråket Modelica, som har åpen kildekode, ved hjelp av programvaren OpenModelica. Simuleringer som belyser bruksområdet til *OpenWPL* har blitt utført, og resultatene har blitt sammenlignet med referanseverdier.

Det ble funnet ut at karakteristikkmetoden ble korrekt implementert i rørkomponenten, både med og uten friksjon. Når resultatene ble sammenliknet med referanseverdiene ble det imidlertidig avdekket at grensebetingelsene i røret representerer en feilkilde i ukjent omfang. Feilen som ble oppdaget var mest utbredt ved få simuleringssteg for lengde. Samtidig er det problemer tilknyttet seriekobling av rør i biblioteket, og det er usikkert om disse problemene har rot i samme feilkilde som ved grensebetingelsene.

*OpenWPL* er foreløpig ikke i stand til å avdekke hvilke dynamiske effekter som finnes i hydrauliske laboratoriesystemer. Biblioteket representerer i stedet et godt fundament som kan videreutvikles til et mer komplekst og fullverdig bibliotek i fremtiden.

# *Preface*

This thesis was written at the Waterpower Laboratory in the spring of 2021 at the Norwegian University of Science and Technology (NTNU). It is the result of a degree at the Department of Energy and Process Engineering within the field of engineering fluid mechanics.

I wish to express my gratitude towards my supervisor at NTNU associate professor Pål-Tore Selbo Storli, as well as my fellow students at the laboratory and NTNU.

# *Contents*

**Abstract** i

**Sammendrag** iii

**Preface** v

**Contents** vii

**List of Tables** xii

**List of Figures** xiv

**List of Symbols** xv

**1 Introduction** 1

1.1 Background information . . . . . 1

1.2 Research objectives . . . . . 3

**2 Theory and software** 5

2.1 Governing equations for fluid motion . . . . . 5

2.1.1 Momentum equation . . . . . 5

2.1.1.1 Friction . . . . . 6
2.1.2 Continuity equation . . . . . 7
2.2 Method of characteristics . . . . . 8
2.2.1 Implementation into the pipe component . . . . . 11
2.2.2 Step sizes . . . . . 11
2.2.3 Adjusted equations for non-uniform cross-sectional area . 12
2.3 Initial conditions . . . . . 12
2.4 Boundary conditions . . . . . 13
2.4.1 Reservoirs . . . . . 14
2.4.2 Valves . . . . . 14
2.4.3 Pipes in series . . . . . 15
2.5 Modelica . . . . . 15
2.5.1 Modelica Fluid Library . . . . . 15
2.5.2 Connectors . . . . . 16
2.5.3 Medium . . . . . 16

**3 Model development and simulation settings 17**
3.1 OpenModelica and OMEdit . . . . . 17
3.2 *OpenWPL* . . . . . 21
3.2.1 Pipe component . . . . . 21
3.2.2 MOC component . . . . . 22
3.3 Simulation parameters and setup for instantaneous valve closure . 22
3.3.1 Single pipe without friction . . . . . 22
3.3.1.1 Change of simulation location . . . . . 24
3.3.1.2 Change of simulation spatial steps . . . . . 25
3.3.2 Single pipe with friction . . . . . 26

3.3.2.1 Change of friction coefficient . . . . . 28
3.3.3 Single pipe with a non-uniform cross-sectional area without friction . . . . . 30

**4 Results and discussion** **33**
4.1 Benchmarking case: instantaneous valve closure . . . . . 33
4.1.1 Single pipe without friction . . . . . 33
4.1.1.1 Simulation validation . . . . . 34
4.1.1.2 Change of simulation location . . . . . 36
4.1.1.3 Change of simulation space steps . . . . . 38
4.1.2 Single pipe with friction . . . . . 39
4.1.2.1 Change of friction coefficient . . . . . 41
4.1.3 Single pipe with a non-uniform cross-sectional area without friction . . . . . 44
4.2 Overall observations . . . . . 46

**5 Conclusion** **49**

**6 Further work** **51**

**References** **53**

**A Full calculation: the equation of motion for transient flow** **55**

**B Full calculation: the continuity equation for transient flow** **57**

**C Full calculation: the method of characteristics** **59**

**D Diagram view of a setup from source to sink** **63**

**E Diagram view of source to sink with intersection and blind flange sim-**

**ulation** **65**

**F Errors related to: source to sink with intersection and blind flange simulation** **67**

**G Modelica code: functions implemented into OpenWPL** **69**

**H Modelica code: pipe component** **79**

# *List of Tables*


3.1 Table of global values which are valid for all simulations. . . . . 23

3.2 Parameter settings for a single pipe without friction where N=4. Results in Figure 4.1. . . . . 23

3.3 Parameter settings for a single pipe without friction at the mid-section where $N_l = 3$. Results in Figure 4.2. . . . . 24

3.4 Parameter settings for a single pipe without friction at the inlet where $N_l = 2$. Results in Figure 4.3. . . . . 24

3.5 Parameter settings for a single pipe without friction where N=8. Results in Figure 4.4 . . . . . 25

3.6 Parameter settings for a single pipe without friction where N=12. Results in Figure 4.5. . . . . 25

3.7 Pressure drop along the pipe from inlet to the closed valve. Results in Figure 4.6. . . . . 26

3.8 Parameter settings for a single pipe with friction. Results in Figure 4.7. . . . . 27

3.9 Parameter settings for a single pipe with friction where N=4. The friction factor has been multiplied by 100 in order to enhance the visualization. Results in Figure 4.8. . . . . 28

3.10 Parameter settings for a single pipe with friction where N=8. The friction factor has been multiplied by 100 in order to enhance the visualization. Results in Figure 4.9. . . . . 28

3.11 Parameter settings for a single pipe with friction where N=12. The friction factor has been multiplied by 100 in order to enhance the visualization. Results in Figure 4.10. . . . . 29

3.12 A pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=4, where a closed valve is located. Results in Figure 4.11 . . . . . 30

3.13 A pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=8, where a closed valve is located. Results in Figure 4.12. . . . . 30

3.14 A pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=12, where a closed valve is located. Results in Figure 4.13. . . . . 31

4.1 Comparison between $T_{r,calculated} = 0.0833(s)$ and $T_{r,simulated}$ for different spatial step N. . . . . 39

# *List of Figures*

1.1 An example from the laboratory where the effects of transient flow could affect measurements downstream. . . . . 1

2.1 Graphic display of the method of characteristics. . . . . 9

2.2 Graphic display of the boundary conditions in the method of characteristics. . . . . 13

3.1 OpenModelica software and GUI as used in this thesis in diagram mode. . . . . 18

3.2 OpenModelica software and GUI as used in this thesis in text mode. This is the underlying code of the graphics in Figure 3.1. . . . . 18

3.3 Hierarchy of software used in this thesis. . . . . 19

3.4 Parameter options of the pipe component. . . . . 20

3.5 *OpenWPL* and its sub-libraries. . . . . 21

4.1 Square pressure pulse at the end of the pipe, where a closed valve is located. Simulation settings in Table 3.2. . . . . 34

4.2 Pressure pulse at the mid-section of the pipe, upstream of where a closed valve is located. Simulation settings in Table 3.3. . . . . 37

4.3 Pressure pulse at the inlet of the pipe, upstream of where a closed valve is located. Simulation settings in Table 3.4. . . . . 37

4.4 Square pressure pulse at the end of the pipe with N=8, where a closed valve is located. Simulation settings in Table 3.5. . . . . 38

4.5 Square pressure pulse at the end of the pipe with N=12, where a closed valve is located Simulation settings in Table 3.6. . . . . 39

4.6 Pressure drop along the pipe from inlet to the closed valve at $T = 1$. Simulation settings in Table 3.7. . . . . 40

4.7 Pressure surges with friction at the end of a pipe, where a closed valve is located. Simulation settings in Table 3.8. . . . . 40

4.8 Pressure surges with friction at the end of a pipe with N=4, where a closed valve is located. Simulation settings in Table 3.9. . . . . 41

4.9 Pressure surges with friction at the end of a pipe with N=8, where a closed valve is located. Simulation settings in Table 3.10. . . . . 43

4.10 Pressure surges with friction at the end of a pipe with N=12, where a closed valve is located. Simulation settings in Table 3.11. . . . . 43

4.11 Pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=4, where a closed valve is located. Simulation settings in Table 3.12. . . . . 44

4.12 Pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=8, where a closed valve is located. Simulation settings in Table 3.13. . . . . 45

4.13 Pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=12, where a closed valve is located. Simulation settings in Table 3.14. . . . . 45

D.1 A simple setup from source to sink via a single pipe. . . . . 63

E.1 A setup from source to sink with intersection and blind flange simulation. . . . . 65

F.1 Balanced *Check Model* message. . . . . 67

F.2 Imbalanced symbolic error. . . . . 67

F.3 Imbalanced translation error. . . . . 67

# *List of Symbols*

**Latin Symbols**

| | | |
|---|---|---|
| $a$ | Speed of sound in water | $\mathrm{m/s}$ |
| $A$ | Pipe area | $m^2$ |
| B | Pipeline characteristic impedance | $\mathrm{s/m^2}$ |
| C | Characteristic constant | $m$ |
| $f$ | Darcy-Weisbach friction factor | |
| $g$ | Gravitational acceleration | $\mathrm{m/s^2}$ |
| $H$ | Hydraulic head | $m$ |
| $D$ | Pipe diameter | $m$ |
| $Q$ | Volumetric flow rate | $\mathrm{m^3/s}$ |
| $N$ | Normal vector | |
| $p$ | Pressure | $Pa$ |
| $p_x$ | Pressure gradient | $Pa$ |
| R | Pipeline resistance coefficient | $\mathrm{s^2/m^3}$ |
| $t$ | Time | $s$ |
| $V$ | Velocity vector in the x-direction | $\mathrm{m/s}$ |
| $V_t$ | Acceleration vector | $\mathrm{m/s^2}$ |

| | | |
|---|---|---|
| $V_x$ | Velocity gradient vector | $^{1}/_{\mathrm{s}}$ |
| $V_{xx}$ | Laplacian of the velocity vector | $^{1}/_{\mathrm{ms}}$ |
| $\cancel{V}$ | Volume | $m^3$ |

**Greek Symbols**

| | | |
|---|---|---|
| $\alpha$ | Angle of horizontal pipe slope | $^{\circ}$ |
| $\kappa$ | Bulk modulus | $^{\mathrm{N}}/_{\mathrm{m}^2}$ |
| $\epsilon$ | Pipe surface roughness | $m$ |
| $\mu$ | Dynamic viscosity | $Pa \cdot s$ |
| $\nabla$ | Divergence | |
| $\nu$ | Kinematic viscosity | $^{\mathrm{m}^2}/_{\mathrm{s}}$ |
| $\rho$ | Density | $^{\mathrm{kg}}/_{\mathrm{m}^3}$ |

**Abbreviation**

| | |
|---|---|
| C- | Minus characteristic |
| C+ | Plus characteristic |
| CS | Control surface |
| CV | Control volume |
| MFL | Modelica Fluid Library |
| MSL | Modelica Standard Library |
| OMEdit | OpenModelica Connection Editor |
| OpenHPL | Open Hydropower Laboratory |
| OpenWPL | Open Waterpower Laboratory |
| NTNU | Norwegian University of Science and Technology |
| Re | Reynolds number |
| USN | University of South-Eastern Norway |

# Chapter I

## *Introduction*

## 1.1 Background information

The Waterpower Laboratory at NTNU is a modern research and development facility housing several testing rigs for hydropower research including; Francis, Pelton and pump turbines. Similar to other experimental setups, verification of results and accuracy of measurement equipment is of vital importance to the facility.

Laboratory measurements of properties in regards to hydraulic turbine models are useful in many aspects. However, when dynamic properties are of interest, downstream measurements may be affected by the hydraulic system and the characteristics of the upstream flow.

The system at the laboratory is highly adaptive, and thus there exists several blind flanges and closed valves due to idle experimental configurations, shown in Figure 1.1. These introduce further dynamic complexity into the system due to present transient effects.

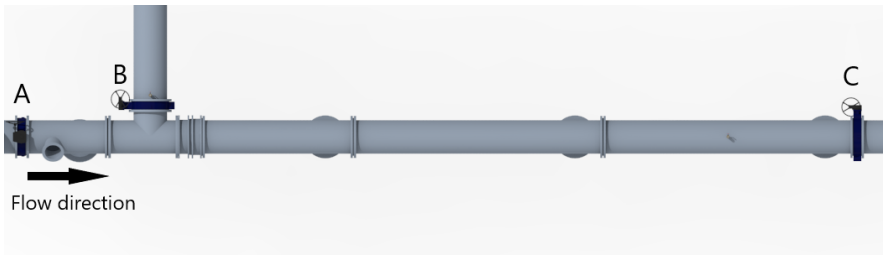



**Figure 1.1:** An example from the laboratory where the effects of transient flow could affect measurements downstream.

Figure 1.1 displays an example from the laboratory where the effects of transient flow could affect measurements downstream. Valve A and B are open, and valve C is closed. Even though the flow is going from valve A to valve B, a continuous string of water is still present in the pipes leading up to valve C. This water has a dynamic effect on the system and introduces pressure surges travelling back and forth between the valve and junction, further propagating into the system due to the compressibility of water and the pipe walls.

Induced pressure surges are affecting the measurements downstream. It is unknown to which extent the measurements are affected, especially due to the complexity of the system which includes pipe bends, intermediate tanks and other closed-off valves. A common problem at the Waterpower Laboratory is differentiating the frequencies of transient effects due to the system dynamics and the frequencies of turbine vibrations and other expected transient effects. This means that it can be difficult to filter the results of experimental testing.

The hydraulic system at the Waterpower Laboratory is rather complex, and it is therefore desirable to have sufficient modelling tools which can be used to simulate the behaviour of the system. This would enable a better understanding of the system dynamics at the laboratory, which in turn could lead to more distinct measurements and open up to new possibilities within hydropower research.

Similar modelling tools such as *OpenHPL* has been developed at the University of South-Eastern Norway (USN), albeit custom-built for full-scale hydropower systems and without the implementation of the water hammer effect. Previous project work suggests that neither *OpenHPL* nor the Modelica Standard Library is capable of modelling the complex system dynamics at the Waterpower Laboratory[1], which introduces a research gap worthwhile to investigate.

## 1.2 Research objectives

A library capable of modelling hydraulic laboratory systems with transient flow will be created using the programming language Modelica. The library will be created and utilized in the open-source simulation environment OpenModelica using the graphical user interface OpenModelica Connection Editor.

The research question which will be answered is related to the capacity of the library, henceforth known as *OpenWPL*, and whether it is capable of revealing the system dynamics of hydraulic laboratory systems.

The first research objective will be to custom-build components which are required to simulate transient flow in pipes. This thesis focuses on the method of characteristics as presented in *Fluid Transients in Systems* [2], and will include customizable pipe components which also can simulate blind flanges and closed valves.

Secondly, models of interesting sections of the hydraulic system at the Waterpower Laboratory will be created in OpenModelica, such as the case shown in Figure 1.1. This includes intersections with blind flanges, pipes with non-uniform cross-sectional areas and cases with instantaneous valve closure. The pipe component as mentioned in the first research objective will be utilized to create said models.

Lastly, the results of the simulated cases as mentioned in the second research objective will be compared to benchmarking cases. This comparison will serve as the basis for determining whether the library is capable of revealing the system dynamics of hydraulic laboratory systems, and as such, answer the main research objective. The rapport will finalize with a discussion on the results obtained, and a section on suggested further work.

The scope of this thesis is restricted by time. Neither the turbine nor the surge tank at the Waterpower Laboratory will be modelled as components due to time restrictions. The hydraulic system should ideally have been modelled in its entirety, from reservoir to sink, including all the components in-between. This will be elaborated upon in chapter 6.

Main research objective

A library named *OpenWPL* capable of modelling hydraulic laboratory systems with transient flow will be created in Modelica using the method of characteristics. Simulated results will be compared to benchmarking cases to determine whether the library is capable of revealing the system dynamics of hydraulic laboratory systems.

# Chapter II

## *Theory and software*

This chapter will introduce the governing equations of transient flow, friction, the method of characteristics and the associated boundary conditions. Finally, it will transition into the next chapter by explaining how the theory was applied in Modelica.

## 2.1 Governing equations for fluid motion

Transient flow is a term that describes unsteady flow where the properties change with respect to both time and position. It is commonly used to describe the flow of fluids in pipelines. This section will present both the equation of motion and the continuity equation in one dimension for transient flow.

### 2.1.1 Momentum equation

The momentum equation, also known as the Navier-Stokes equation, is given in Equation 2.1, and describes 1D incompressible flow in the x-direction[3]:

$$V_t + VV_x = -\frac{p_x}{\rho} + g + \nu V_{xx} \tag{2.1}$$

where $V_t$ $(\mathrm{m/s^2})$ is the acceleration vector in the x-direction, $V$ $(\mathrm{m/s})$ is the velocity vector in the x-direction, $V_x$ $(\mathrm{1/s})$ is the velocity gradient in the x-direction, $p_x$ $(\mathrm{Pa/m})$ is the pressure gradient of the fluid, $\rho$ $(\mathrm{kg/m^3})$ is the density of the fluid, $g$ $(\mathrm{m/s^2})$ is the gravitational acceleration, $\nu$ $(\mathrm{m^2/s})$ is the kinematic viscosity of the fluid and $V_{xx}$ $(\mathrm{1/ms})$ is the Laplacian of the velocity vector, i.e., scalar divergence to the velocity gradient in the x-direction.

By rearranging the terms one achieves:

$$\frac{p_x}{\rho} + VV_x + V_t - g - \nu V_{xx} = 0 \tag{2.2}$$

A version of the Navier-Stokes equation is derived for fluid flow through a conical or cylindrical tube. It is valid for any fluid, and given in Equation 2.3 which is adjusted to angled pipes[2].

The viscous term has been replaced by a friction term utilizing the Darcy-Weisbach friction factor. This assumes quasi-steady flow, i.e, flow that is changing so slowly that the related friction effects are negligible. This assumption is elaborated on in chapter 6.

$$\frac{p_x}{\rho} + VV_x + V_t + g\sin\alpha + \frac{fV|V|}{2D} = 0 \tag{2.3}$$

where $\alpha$ is the angle of a potential slope, $f$ is the Darcy-Weisbach friction factor and $D$ $(m)$ is the diameter of the pipe.

When looking at Equation 2.3 in terms of the hydraulic grade line, density is considered to be substantially constant compared with the variations in $H$ or $z$. Equation 2.3 is therefore simplified to Equation 2.4, and only valid for less compressible fluids, such as water. See Appendix A for detailed calculations.

$$gH_x + V_t + \frac{fV|V|}{2D} = 0 \tag{2.4}$$

where $H$ $(m)$ is the hydraulic grade line.

#### 2.1.1.1 Friction

Friction is the force that connects the modelled equations to the real world. It is introduced in Equation 2.4 within the term $\frac{fV|V|}{2D}$ where $f$ is the Darcy-Weisbach friction factor.

The friction factor can be directly approximated using the Haaland equation given in Equation 2.5 below. It is an approximation of the Colebrook-White equation and has the advantage of being explicitly solvable. This approximation introduces an error deviance of $2\%$ compared to the original Colebrook-White equation [3].

$$\frac{1}{\sqrt{f}} = -1.8 \log \left[ \left( \frac{\epsilon/\mathrm{D}}{3.7} \right)^{1.11} + \frac{6.9}{Re} \right] \tag{2.5}$$

where $\epsilon$ $(m)$ is the pipe surface roughness and $Re$ is the Reynolds number.

### 2.1.2 Continuity equation

Similarly, the continuity equation states that the rate at which mass flows into the control volume minus the rate at which mass flows out of the control volume is equal to the net rate of change of mass within the control volume[3]. It is given in its general differential form for a control volume (CV) and control surface (CS) in Equation 2.6.

$$\frac{d}{dt} \int_{CV} \rho \, d\forall + \int_{CS} \rho (V \cdot N) \, dA = 0 \tag{2.6}$$

where $\forall$ $(m^3)$ is the volume, $\vec{v}$ $(\mathrm{m}/\mathrm{s})$ is the velocity vector and $\vec{n}$ is the normal vector.

A version of the continuity equation is also derived for fluid flow through a conical or cylindrical tube. It is valid for any unsteady fluid flow, and given in Equation 2.7 where the total derivative is indicated by a dot [2].

$$\frac{\dot{A}}{A} + \frac{\dot{\rho}}{\rho} + V_x = 0 \tag{2.7}$$

Equation 2.8 is valid for slightly compressible fluids, and is limited to low velocities. While Equation 2.7 is a more general form of continuity, Equation 2.8 is simplified and will be used in the following sections.

$$\frac{a^2 V_x}{g} + H_t = 0 \tag{2.8}$$

where the wave velocity is given as;

$$a^2 = \frac{\mathrm{K}/\rho}{1 + (\mathrm{K}/\mathrm{A})(\Delta \mathrm{A}/\Delta \mathrm{p})} \tag{2.9}$$

where $\kappa$ $(\mathrm{N}/\mathrm{m}^2)$ is the bulk modulus.

## 2.2 Method of characteristics

The method of characteristics is a numerical method that transforms partial differential equations into total differential equations. These equations can be integrated to yield finite difference equations, which in turn can be solved numerically.

The two PDEs Equation 2.4 and Equation 2.8 repeated below can be used to model the flow in pipes. This section will explain how the method of characteristics is applied to the equations in question.

$$gH_x + V_t + \frac{fV|V|}{2D} = 0 \tag{2.4 revisited}$$

$$\frac{a^2 V_x}{g} + H_t = 0 \tag{2.8 revisited}$$

Equation 2.4 and Equation 2.8 are combined linearly as shown in Equation 2.10 below using an unknown $\lambda$:

$$L = L_1 + \lambda L_2 = 0 \tag{2.10}$$

where

$$L_1 = gH_x + V_t + \frac{f}{2D} V|V| = 0$$

$$L_2 = H_t + \frac{a^2}{g} V_x = 0$$

The linear combination yields the two following equations as shown in Equation 2.11. See Appendix C for full calculations.

$$\lambda = \pm \frac{g}{a} \tag{2.11}$$

Next, Equation 2.12 and Equation 2.13 are achieved by substitution of Equation 2.11. Either set of equations are valid as long as the other is not.

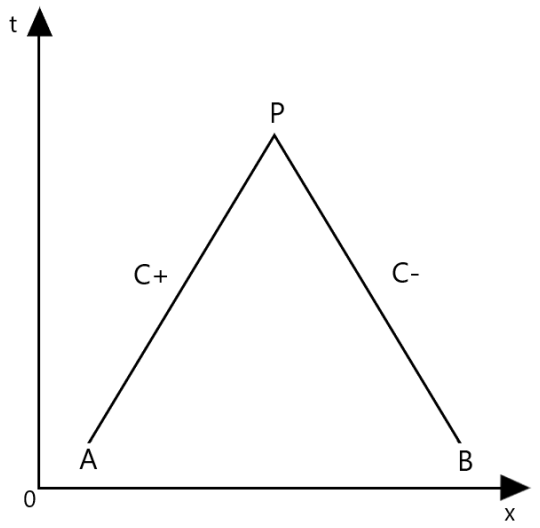



**Figure 2.1:** Graphic display of the method of characteristics.

$$C^+ : \begin{cases} \frac{g}{a}\frac{dH}{dt} + \frac{dV}{dt} + \frac{fV|V|}{2D} = 0 \\ \frac{dx}{dt} = a \end{cases} \tag{2.12}$$

$$C^- : \begin{cases} -\frac{g}{a}\frac{dH}{dt} + \frac{dV}{dt} + \frac{fV|V|}{2D} = 0 \\ \frac{dx}{dt} = -a \end{cases} \tag{2.13}$$

The finite difference equations are given below based on Equation 2.12 and Equation 2.13.

$$C^+ : H_i = C_P - B_P Q_i \tag{2.14}$$

$$C^- : H_i = C_M - B_M Q_i \tag{2.15}$$

Notice that $C_P$, $C_M$, $B_P$ and $B_M$ are known constants at the time of calculation. Furthermore, the constants are given as:

$$C_P = H_{i-1} + BQ_{i-1} \tag{2.16}$$

$$C_M = H_{i+1} - BQ_{i+1} \tag{2.17}$$

and

$$B_P = B + R|Q_{i-1}| \tag{2.18}$$

$$B_M = B + R|Q_{i+1}| \tag{2.19}$$

where $B$ is a function of physical properties, and is known as the pipeline characteristic impedance:

$$B = \frac{a}{gA} \tag{2.20}$$

Similarly, $R$ is known as the pipeline resistance coefficient:

$$R = \frac{f\Delta x}{2gDA^2} \tag{2.21}$$

where $f$ is the Darcy-Weisbach friction factor given in Equation 2.5 in subsubsection 2.1.1.1. Now, by combining Equation 2.14 and Equation 2.15 one achieves the final equations given below:

$$H_i = \frac{C_P B_M + C_M B_P}{B_P + B_M} \tag{2.22}$$

$$Q_i = \frac{C_P - C_M}{B_P + B_M} \tag{2.23}$$

Equation 2.22 and Equation 2.23 describes the hydraulic head $H$ and the volumetric flow rate $Q$. These equations are final and is valid at the point $i$, or $P$.

### 2.2.1 Implementation into the pipe component

The method of characteristics as shown in section 2.2 is applied to every point $[j, i]$ where $j = 2 : T$ and $i = 2 : N - 1$, i.e., wherever the initial and boundary conditions do not apply. The method revolves around solving $H$ and $Q$ based on the previous time and space characteristics. Both $N$ and $T$ are decided beforehand using the equations given in subsection 2.2.2.

Arrays of $Q$, $H$, $f_p$, $f_m$, $R_p$, $R_m$, $B_p$, $B_m$, $C_p$ and $C_m$ are solved in nested for-loops using the equations given in section 2.2. All equations are transformed into functions which is reused for all components. The functions are available in the *OpenWPL.Functions* package, and shown in Appendix G.

The method starts by determining a linear space for area and diameter. Next, velocity $v$, hydraulic head $H$ and volumetric flow rate $Q$ are decided based on previous values. The initial values are decided by the initial conditions, explained in section 2.3. Equation 2.22 and Equation 2.23 are the main equation used to calculate the head and volumetric flow rate.

If $friction = true$ then Equation 2.5 is used to calculate the Darcy-Weisbach friction factor. It is dependent on global values such as density, viscosity and roughness of the pipe walls, as well as local values such as velocity and diameter. If $friction = false$ then all friction values are zero; one array for the plus characteristic and one for the minus characteristic.

Finally, characteristic equations for both the plus characteristic and minus characteristic are calculated in the next step in space and time.

### 2.2.2 Step sizes

The time step $T$ and the spatial step $N$ must be decided beforehand. $T = 2$ is a requirement for the method of characteristics to be utilized, as $T = 1$ is initiated by the initial condition as shown in section 2.3.

These will be used to calculate step sizes $\Delta t$ and $\Delta x$ in time and space, respectively. Each boundary condition requires one step in space each, and the method of characteristics require at least one step to be utilized. This means that $N = 3$ is a minimum requirement. The step size in space $\Delta x$ can thus be calculated based on the pipe length $L$ and the number of steps $N$:

$$\Delta x = \frac{L}{N} \tag{2.24}$$

Next, the step size in time $\Delta t$ $(s)$ can thus be calculated based on the step size in

space $\Delta x\ (m)$ and speed of sound in water $a\ (m)$:

$$\Delta t = \frac{\Delta x}{a} \tag{2.25}$$

### 2.2.3 Adjusted equations for non-uniform cross-sectional area

The equations given in section 2.2 are only valid for a constant cross-sectional area. The equations below, however, are adapted such that the area can gradually change, making them more useful for real-life applications. The calculation of Equation 2.26 and Equation 2.27 are not shown in this thesis, but can be found in their entirety in *Simulation and analysis of FCR operation of a Francis turbine* [4].

$$Q_P = \frac{\frac{g}{a}(H_A - H_B) + \frac{1}{2}Q_B\left(\frac{1}{A_P} + \frac{1}{A_B}\right) + \frac{1}{2}Q_A\left(\frac{1}{A_P} + \frac{1}{A_A}\right)}{\frac{1}{A_P} + \frac{1}{2A_B} + \frac{1}{2A_A}} \tag{2.26}$$

$$H_P = H_A - \frac{a}{g}\left[\frac{Q_P}{A_P} - \frac{Q_A}{A_A} - \frac{1}{2}(Q_P + Q_A)\left(\frac{1}{A_P} - \frac{1}{A_A}\right)\right] \tag{2.27}$$

## 2.3 Initial conditions

Initial conditions are necessary to simulate a system with respect to time. In this thesis, initial conditions are calculated based on steady-state conditions. Subsequently, the method of characteristics uses the initial condition as a basis for further iterations, and it is therefore important to start with accurate initial values.

They can be calculated using Equation 2.28 based on the previous hydraulic head and current volumetric flow rate, e.g., using values from the upstream reservoir.

$$H = H_{previous} - f\frac{\Delta l}{D}\frac{Q|Q|}{2gA^2} \tag{2.28}$$

where $\Delta l(m)$ is the change in pipe length from the previous iteration and $Q\ (\mathrm{m}^3/\mathrm{s})$ is the volumetric flow rate.

The initial conditions aims to set the values of $H$ and $Q$ for points $[j = 1, i = 1 : N]$. The first point in space and time $[j = 1, i = 1]$ is defined by using hydraulic head $H$ and volumetric flow rate $Q$ from *port_a*. This is valid both for upstream reservoirs and upstream pipes in series.

Next, Equation 2.28 is used to calculate the values where $[j = 1, i = 2 : N]$ based on the previously obtained $H$ and current $Q$ over the displaced length $L$. This will ensure that friction makes it so that $H[j = 1, i = 1] > [j = 1, i = N]$, which is what one would expect.

## 2.4 Boundary conditions

Boundary conditions are necessary to achieve a balanced system in terms of equations and variables. By inserting physical parameters which are known at the boundaries into equations given in section 2.2, one gets equations that are valid both on the left and the right side of the pipe.

In general, a boundary condition is derived from either knowing the hydraulic head $H$ or the volumetric flow rate $Q$ at the boundary itself, followed by calculating the missing parameter of the two using Equation 2.29 and Equation 2.30.

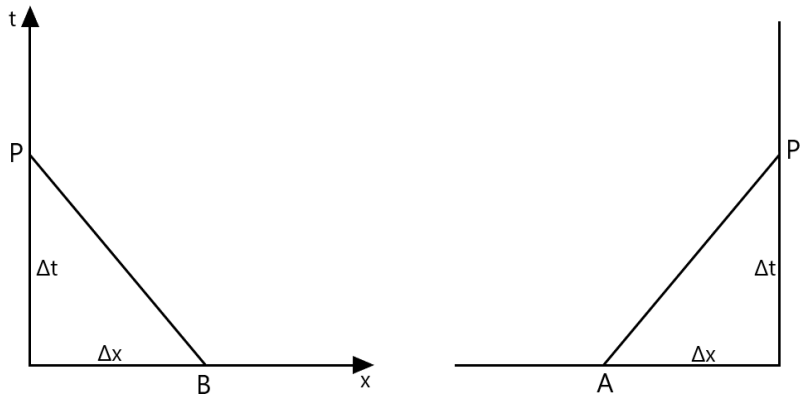



**Figure 2.2:** Graphic display of the boundary conditions in the method of characteristics.

The boundary conditions aim to set the values of either $H$ or $Q$ for points $[j = 2 : T, i = 1]$ at the left boundary and $[j = 2 : T, i = N]$ at the right boundary. They depend on which component is at either end of the pipe. If either end are connected to pipes in series, $H$ and $Q$ are simply transferred between the ports. The following subsections will describe boundary conditions as derived for different mechanical equipment at either end of the pipe.

### 2.4.1 Reservoirs

For a large upstream reservoir the change in the hydraulic grade line can be assumed constant at the connection, i.e., $\Delta P = 0$ at the boundary. By applying Equation 2.15 where $i$ refers to the point at the boundary itself, i.e., $H_i = H_{reservoir}$, $Q_i$ can be calculated by moving terms around as:

$$Q_i = \frac{H_{reservoir} - C_m}{B_m} \tag{2.29}$$

where all variables in Equation 2.29 are known, except $Q_i$, at the time of calculation. Now that both $Q_i$ and $H_i$ are known at the boundary, they can be used by the method of characteristics to calculate $Q_{i+1}$ and $H_{i+1}$. As such, Equation 2.29 will be used to calculate the boundary condition at *port_a*

Note that the hydraulic head of the reservoir is assumed constant, which is an assumption prone to errors, depending on the system simulated. In the case of the Waterpower Laboratory, the tanks are continuously filled during experimental testing by external pumps. In turn, this means that the error related to constant reservoir pressure is acceptable.

### 2.4.2 Valves

Consider a valve downstream of a pipe. If the valve is closing, we can assume that $Q_i = 0$. By applying Equation 2.14 where $i$ refers to the point at the boundary itself, i.e. $Q_i = Q_{valve}$, $H_i$ can be calculated by moving terms around as:

$$H_i = C_p \tag{2.30}$$

where, similar to the reservoir, all variables in Equation 2.30 are known, except $H_i$, at the time of calculation. Now that both $H_i$ and $Q_i$ are known at the boundary, they can be used by the method of characteristics to calculate $Q_{i-1}$ and $H_{i-1}$.

The right boundary condition at *port_b* can be defined using Equation 2.30 if there is a valve closing downstream of the pipe. This boundary condition is useful when checking the validity of the library, as one can simulate cases where certain results are expected, e.g. transient flow effect.

Blind flanges and closed valves are closely related, and the equations for a closing valve is therefore applied to the blind flanges. As previously mentioned, this can be activated as $closedValve = true$ in the pipe component parameter options. If $closedValve = false$ then the right boundary conditions will simply transfer the

last known $H$ and $Q$ to *port_b*.

### 2.4.3 Pipes in series

The boundary conditions for pipes in series are, quite simply, calculated as $H_{left,pipe} = H_{right,pipe}$ and $Q_{left,pipe} = Q_{right,pipe}$. This is only true if the diameter of either pipe is identical at the connection, which is a requirement for the model. If not, the continuity equations are not valid.

## 2.5 Modelica

Modelica is an object-oriented and equation-based programming language for modelling complex systems. Models are described by differential, algebraic and discrete equations, which is used to model the dynamic behaviour of technical systems [5].

Modelica is an open-source language that, similar to MATLAB, is built to describe mathematical models. It is often described as a modelling language, and highly resembles other object-oriented programming languages, such as $C++$.

Classes are fundamental structuring units which are translated into objects, and provide the basis for code instantiation. E.g., the library *OpenWPL* is a package class and the *pipe.mo* component is a model class[6].

Being object-oriented, components can be re-used in the same simulation. I.e., models only have to be built once and can then be utilized again different configurations. This is an efficient way of modelling systems and is highly user-friendly as the user do not need prerequisite knowledge of neither programming nor the underlying physics of the component itself.

If the same simulations would be executed in say, MATLAB, one would have to copy and paste the code which models the pipe component. This can be confusing, especially if the user is not familiar with the underlying code for the components in question. Also, lacking a user-friendly GUI, one would be forced to enter the MATLAB scripts to specify pipe length and other parameters. Using Modelica, this can be done on the canvas itself, without needing to alter the underlying code and scripts, which require knowledge of the component modelling structure.

### 2.5.1 Modelica Fluid Library

In addition to developing the free Modelica programming language, The Modelica Association also develops the open-source software Modelica Standard Library (MSL) for multi-domain models. The MSL includes a plethora of sub-libraries that can be utilized when modelling specific systems, e.g. *Modelica.Thermal* for

modelling heat transfer.

The components designed in this thesis have been programmed in such a way that they can interact with the majority of components in the *Modelica.Fluid* library. It is a comprehensive library that includes many functions and interfaces which can be reused and modified. An advantage of *Modelica.Fluid* in terms of hydraulic modelling is the possibility of backwards and compressible flow, which is integrated into several components.These components can interact with one another by drawing blue lines between them. How this is possible is described in subsection 2.5.2 below.

### 2.5.2 Connectors

A major design element in modelling with Modelica is the use of connectors. Fluid connectors are ports represented by nodes where flow enters or leaves, e.g. the point of entry of pipes. Ports are similar to nodes in electrical engineering, and as such, the laws of conservation must be valid at all time. *OpenWPL* utilizes the standard connector *FluidPort* represented by the model *PartialTwoPort* within the MFL. In turn, this leads to possible interaction between components in *OpenWPL* and MFL.

When utilizing the connector *FluidPort* in components, they are automatically equipped with a graphical node on either side of the component. This node is interactive in diagram mode and can be utilized by drawing a blue line between either set of components. When the line has been drawn, code that connects either node to each other is automatically created in the text mode. In the case of the pipe component created in this thesis, the nodes apply continuity in regards to volumetric flow rate $Q$ and hydraulic head $H$. This means that the volumetric flow rate and hydraulic head at the outlet of the *Source* component is transferred to the inlet of the pipe component.

### 2.5.3 Medium

The MFL relies on the package *Modelica.Media.Water* to provide standardized settings for the fluid medium, more specifically incompressible water with constants properties. This makes it possible to easily change the medium of the modelled system without changing fluid properties in each component. It is a feature of the MFL which is required to be present, also in the custom made pipe component, to be compatible with the standardized components.

Certain parameters, however, are required to be set manually. Such parameters can be overwritten in the component *moc* and *system* as described in subsection 3.2.2 and section 3.3.

# Chapter III

## *Model development and simulation settings*

The research objectives of this thesis will be solved by performing transient simulations of pipe sections. A library specifically designed for the Waterpower Laboratory named *OpenWPL* will be created using the programming language Modelica to perform the necessary simulations. A key feature of *OpenWPL* will be a pipe component named *pipe.mo*, henceforth known as the pipe component, which will act as the centrepiece of transient simulations.

## 3.1 OpenModelica and OMEdit

The simulation environment used in this thesis is OpenModelica utilizing the graphical user interface OpenModelica Connection Editor (OMEdit). To explain how simulations were performed, this subsection will use a simulation in OpenModelica and OMEdit as an example.

The GUI can be seen from Figure 3.1 which displays a model named *Pipe_test_v12* on the central canvas. The libraries are located on the left hand side, along with sub-packages such as *Tests* and *Components*. In the case shown in Figure 3.1 and Figure 3.2 the model *Pipe_test_v12* is located under the sub-package *Tests*. There are five components on the canvas:

- A *OpenTank* component named *Source*.
- A *Pipe* component named *pipe*.
- A *OpenTank* component named *Sink*.
- A *MOC* component named *moc*.
- A *System* component named *system*.

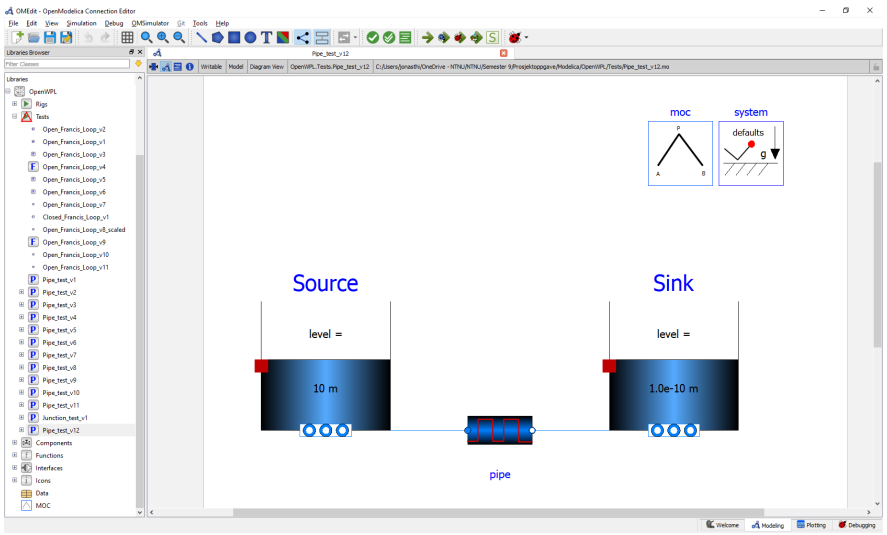
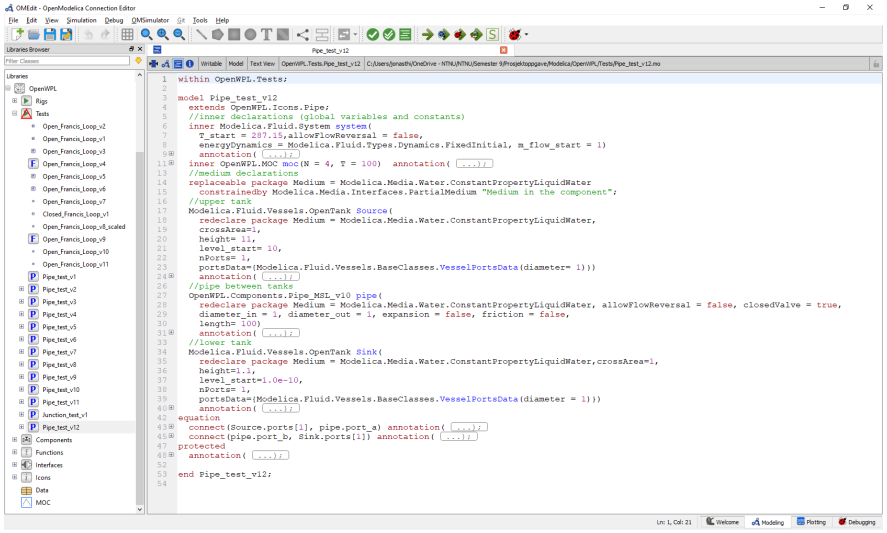



**Figure 3.1:** OpenModelica software and GUI as used in this thesis in diagram mode.



```
within OpenWPL.Tests;

model Pipe_test_v12
  extends OpenWPL.Icons.Pipe;
  //inner declarations (global variables and constants)
  inner Modelica.Fluid.System system(
    T_start = 287.15,allowFlowReversal = false,
    energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial, m_flow_start = 1)
    annotation( ...);
  inner OpenWPL.MOC moc(N = 4, T = 100)  annotation( ...);
  //medium declarations
  replaceable package Medium = Modelica.Media.Water.ConstantPropertyLiquidWater
    constrainedby Modelica.Media.Interfaces.PartialMedium "Medium in the component";
  //upper tank
  Modelica.Fluid.Vessels.OpenTank Source(
    redeclare package Medium = Modelica.Media.Water.ConstantPropertyLiquidWater,
    crossArea=1,
    height= 11,
    level_start= 10,
    nPorts= 1,
    portsData={Modelica.Fluid.Vessels.BaseClasses.VesselPortsData(diameter= 1)})
    annotation( ...);
  //pipe between tanks
  OpenWPL.Components.Pipe_MSL_v10 pipe(
    redeclare package Medium = Modelica.Media.Water.ConstantPropertyLiquidWater, allowFlowReversal = false, closedValve = true,
    diameter_in = 1, diameter_out = 1, expansion = false, friction = false,
    length= 100)
    annotation( ...);
  //lower tank
  Modelica.Fluid.Vessels.OpenTank Sink(
    redeclare package Medium = Modelica.Media.Water.ConstantPropertyLiquidWater,crossArea=1,
    height=1.1,
    level_start=1.0e-10,
    nPorts= 1,
    portsData={Modelica.Fluid.Vessels.BaseClasses.VesselPortsData(diameter = 1)})
    annotation( ...);
equation
  connect(Source.ports[1], pipe.port_a) annotation( ...);
  connect(pipe.port_b, Sink.ports[1]) annotation( ...);
protected
  annotation( ...);

end Pipe_test_v12;
```

**Figure 3.2:** OpenModelica software and GUI as used in this thesis in text mode. This is the underlying code of the graphics in Figure 3.1.

The overall hierarchy of software used in this thesis is shown in the figure below.

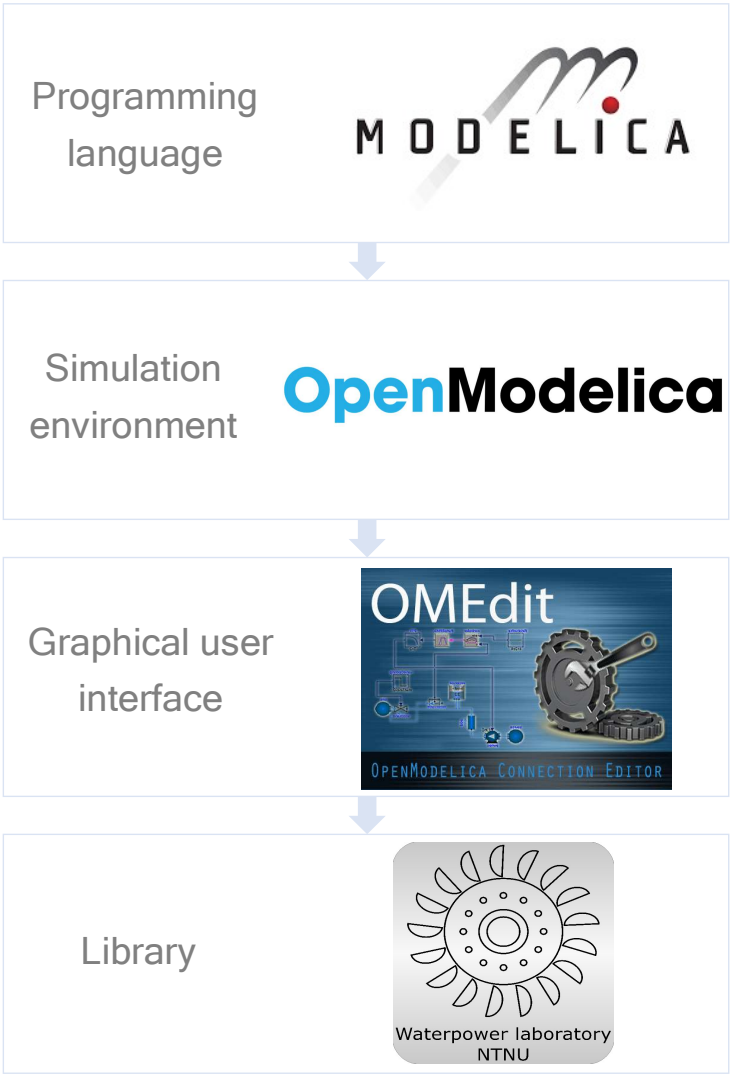



**Figure 3.3:** Hierarchy of software used in this thesis.

Figure 3.2 displays the same model as in Figure 3.1, except in text mode, e.g., when you double click a component on the canvas and set the length to $100(m)$, OpenModelica will automatically add $length = 100$ in the code. Text mode displays the underlying lines of code automatically generated when each component was dragged from the library onto the canvas. Similarly, the code was also altered when each model was double-clicked to set the parameters.

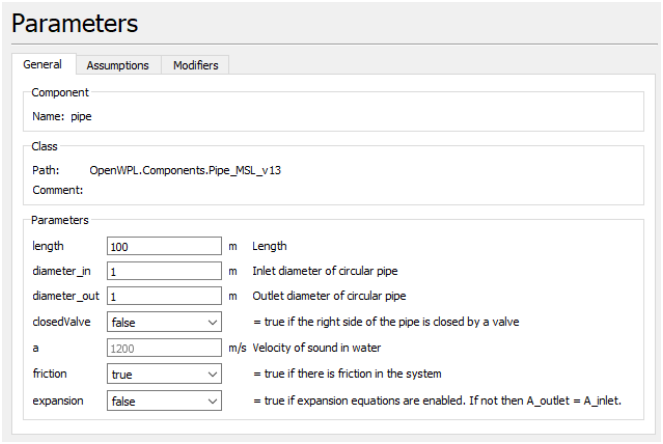



**Figure 3.4:** Parameter options of the pipe component.

Each component represents a model which contains separate lines of predefined code, and each component can be double-clicked to configure its parameters as shown in Figure 3.4. This is a simple way to alter parameters of pipes without the need to alter code, simply accessed by double-clicking the component on the canvas. Notice how each physical element is connected with blue lines in Figure 3.1, and be aware that also *Pipe_test_v12* is of the model class, i.e., the model itself which includes the whole canvas. Observe that both components named *Source* and *Sink* are of the same model named *OpenTank*. While *Source* and *Sink* can be thought of as local names only applied in the current model named *Pipe_test_v12*, they are both coded as a *OpenTank* model. This means that the component has only been coded once, but applied twice using two different configurations. This is the main principle and forte of Modelica.

In the case of Figure 3.1 and Figure 3.2 the *OpenTank* model is a component available in the Modelica Fluid Library, while *Pipe* is a component from the *OpenWPL* libray. E.g. they are instantiated (or called) using the naming convention *Modelica.Fluid.Vessels.OpenTank* and *OpenWPL.Components.Pipe* respectively.

## 3.2 *OpenWPL*

*OpenWPL* is composed of several sub-packages as seen in Figure 3.5. *Rigs* and *Tests* are packages containing complete and incomplete experimental setups, respectively. *Components* mainly contains versions of the pipe component and *Functions* contains all necessary functions as seen in Appendix G. *Interfaces* contain definitions for the ports and *Icons* merely contain icons for other models. *Data* and *MOC* on the other hand are not packages, but models. They are imported into other models in order to set global parameters, as explained in subsection 3.2.2. *Data* was imported from the *OpenHPL* library, but is currently not in use.

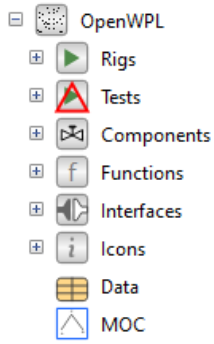



**Figure 3.5:** *OpenWPL* and its sub-libraries.

The pipe component and the MOC component are the most important models in *OpenWPL* and were built using the approach as seen in chapter 2. Whenever a new version of the pipe component was created, it was put into the sub-package *Components*. Then, a new test rig from source to sink was created to test the pipe components and their features, as seen in Figure D.1 in Appendix D.

### 3.2.1 Pipe component

The pipe component is extended upon *Modelica.Fluid.Interfaces.PartialTwoPort* which defines an interface for components with two ports. I.e., everything that is coded in *Modelica.Fluid.Interfaces.PartialTwoPort* is also included in the pipe component in *OpenWPL*. This enables compatibility with the rest of the fluid library and allows flow reversal as defined in the system-wide settings. The component may transport fluids as defined by the medium.

The interface defines *port_a* and *port_b* which corresponds to the left and right side of the pipe, respectively. They encapsulate the method of characteristics, and acts as boundaries, simultaneously providing the pipe with access points that can

connect to other components. The creation of the pipe component is explained in section 2.2, starting with the nested for-loops iterating in-between the boundary conditions, followed by initial conditions and finally the boundary conditions at *port_a* and *port_b*. Data is stored in arrays and vectors depending on whether they change in respect to both time and space, or merely space, respectively. The vectors are of size $N$ and the arrays of size $N \cdot T$.

### 3.2.2 MOC component

A model named *moc* has been created to make global simulation settings related to the method of characteristics available to all components in the simulated model. Most notably, the number of time steps $T$ and the number of intervals $N$ are set in this component, which ensures that every component uses the same values during the simulations.

For future versions of the library, it would be beneficial to add further compatibility options in regards to the method of characteristics in this model. More specifically, the *moc* component should decide the minimum amount of intervals $N$ for all pipes in the model, based on the pipe with the smallest physical length. Currently, this option is disabled, thus N must be manually set by the user.

## 3.3 Simulation parameters and setup for instantaneous valve closure

The system is simulated in OMEdit using the global parameters as shown in Table 3.1. Similar to the *moc* component mentioned in subsection 3.2.2, a MFL specific component named $system$ is also used during the simulations. Several of the parameter settings shown in Table 3.1 are set in this component.

The following simulations are all initialized by initial conditions occurring at time step $T = 1$, followed by an instantaneous valve closure occurring at $T = 2$. This action induces dynamic pressure surges which will be presented and discussed in section 4.1.

### 3.3.1 Single pipe without friction

This is the first simulation performed in this thesis. The settings of the simulations are shown in Table 3.2, in addition to the parameter settings in Table 3.1 which are valid for all simulations. The setup can also be seen in diagram mode in Appendix D.

This simulation checks if the method of characteristics is implemented correctly. It assumes that the outlet of the pipe component is instantaneously closed by a valve, while friction is disabled. One would expect to see a square pressure pulse, which would confirm the functionality of the model [7]. The results can be found in Figure 4.1 in subsection 4.1.1.

The following sections will present additional simulation settings for different configurations. All results are discussed in chapter 4.

**Table 3.1:** Table of global values which are valid for all simulations.

| Parameter | Value | Unit |
|---|---|---|
| Temperature | 14 | $°C$ |
| Pipe roughness | 0.000025 | $m$ |
| Fluid density | 999.2 | $\mathrm{kg}/\mathrm{m}^3$ |
| Dynamic viscosity | 0.0011684 | $Pa \cdot s$ |
| Kinematic viscosity | 0.0000016934 | $\mathrm{m}^2/\mathrm{s}$ |
| Gravity | 9.81 | $\mathrm{m}/\mathrm{s}^2$ |

**Table 3.2:** Parameter settings for a single pipe without friction where N=4. Results in Figure 4.1.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 4 | |
| Simulation location | $N_l$ | 4 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1 | $m$ |
| Velocity of sound | a | 1200 | $\mathrm{m}/\mathrm{s}$ |
| Valve | closedValve | true | |
| Friction | friction | false | |
| Friction coefficient | f | f | |
| Expansion | expansion | false | |

#### 3.3.1.1 Change of simulation location

**Table 3.3:** Parameter settings for a single pipe without friction at the mid-section where $N_l = 3$. Results in Figure 4.2.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 4 | |
| Simulation location | $N_l$ | 3 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1 | $m$ |
| Velocity of sound | a | 1200 | $\mathrm{m}/\mathrm{s}$ |
| Valve | closedValve | true | |
| Friction | friction | false | |
| Friction coefficient | f | f | |
| Expansion | expansion | false | |

**Table 3.4:** Parameter settings for a single pipe without friction at the inlet where $N_l = 2$. Results in Figure 4.3.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 4 | |
| Simulation location | $N_l$ | 2 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1 | $m$ |
| Velocity of sound | a | 1200 | $\mathrm{m}/\mathrm{s}$ |
| Valve | closedValve | true | |
| Friction | friction | false | |
| Friction coefficient | f | f | |
| Expansion | expansion | false | |

#### 3.3.1.2 Change of simulation spatial steps

**Table 3.5:** Parameter settings for a single pipe without friction where N=8. Results in Figure 4.4

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 8 | |
| Simulation location | $N_l$ | 8 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1 | $m$ |
| Velocity of sound | a | 1200 | $^{\mathrm{m}}/_{\mathrm{s}}$ |
| Valve | closedValve | true | |
| Friction | friction | false | |
| Friction coefficient | f | f | |
| Expansion | expansion | false | |

**Table 3.6:** Parameter settings for a single pipe without friction where N=12. Results in Figure 4.5.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 12 | |
| Simulation location | $N_l$ | 12 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1 | $m$ |
| Velocity of sound | a | 1200 | $^{\mathrm{m}}/_{\mathrm{s}}$ |
| Valve | closedValve | true | |
| Friction | friction | false | |
| Friction coefficient | f | f | |
| Expansion | expansion | false | |

### 3.3.2 Single pipe with friction

The next case which will be simulated is similar to the previous case, except with friction. It would be feasible to expect that the overall hydraulic head is decreasing over time. The setup can also be seen in diagram mode in Appendix D, and the result can be found in subsection 4.1.2.

Even though the library is built for small-scale hydropower laboratory systems with, say pipe lengths of $L = 2(m)$, a pipe length of $L = 50(m)$ is used instead for all simulations. The reason is because friction is a function of length, and that it is easier to visualize the effect friction has on the results for longer pipes.

**Table 3.7:** Pressure drop along the pipe from inlet to the closed valve. Results in Figure 4.6.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 1 | |
| Number of intervals | N | 4 | |
| Simulation location | $N_l$ | 1-4 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1 | $m$ |
| Velocity of sound | a | 1200 | $^{\mathrm{m}}/_{\mathrm{s}}$ |
| Valve | closedValve | true | |
| Friction | friction | true | |
| Friction coefficient | f | f | |
| Expansion | expansion | false | |

**Table 3.8:** Parameter settings for a single pipe with friction. Results in Figure 4.7.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 4 | |
| Simulation location | $N_l$ | 4 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1 | $m$ |
| Velocity of sound | a | 1200 | $\mathrm{m}/\mathrm{s}$ |
| Valve | closedValve | true | |
| Friction | friction | true | |
| Friction coefficient | f | f | |
| Expansion | expansion | false | |

#### 3.3.2.1 Change of friction coefficient

**Table 3.9:** Parameter settings for a single pipe with friction where N=4. The friction factor has been multiplied by 100 in order to enhance the visualization. Results in Figure 4.8.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 4 | |
| Simulation location | $N_l$ | 4 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1 | $m$ |
| Velocity of sound | a | 1200 | $^{\mathrm{m}}/_{\mathrm{s}}$ |
| Valve | closedValve | true | |
| Friction | friction | true | |
| Friction coefficient | f | 100f | |
| Expansion | expansion | false | |

**Table 3.10:** Parameter settings for a single pipe with friction where N=8. The friction factor has been multiplied by 100 in order to enhance the visualization. Results in Figure 4.9.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 8 | |
| Simulation location | $N_l$ | 8 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1 | $m$ |
| Velocity of sound | a | 1200 | $^{\mathrm{m}}/_{\mathrm{s}}$ |
| Valve | closedValve | true | |
| Friction | friction | true | |
| Friction coefficient | f | 100f | |
| Expansion | expansion | false | |

**Table 3.11:** Parameter settings for a single pipe with friction where N=12. The friction factor has been multiplied by 100 in order to enhance the visualization. Results in Figure 4.10.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 12 | |
| Simulation location | $N_l$ | 12 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1 | $m$ |
| Velocity of sound | a | 1200 | $^{\mathrm{m}}/_{\mathrm{s}}$ |
| Valve | closedValve | true | |
| Friction | friction | true | |
| Friction coefficient | f | 100f | |
| Expansion | expansion | false | |

### 3.3.3 Single pipe with a non-uniform cross-sectional area without friction

The setup can also be seen in diagram mode in Appendix D.

**Table 3.12:** A pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=4, where a closed valve is located. Results in Figure 4.11

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 4 | |
| Simulation location | $N_l$ | 4 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1.1 | $m$ |
| Velocity of sound | a | 1200 | $^{\mathrm{m}}/_{\mathrm{s}}$ |
| Valve | closedValve | true | |
| Friction | friction | false | |
| Friction coefficient | f | f | |
| Expansion | expansion | false | |

**Table 3.13:** A pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=8, where a closed valve is located. Results in Figure 4.12.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 8 | |
| Simulation location | $N_l$ | 8 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1.1 | $m$ |
| Velocity of sound | a | 1200 | $^{\mathrm{m}}/_{\mathrm{s}}$ |
| Valve | closedValve | true | |
| Friction | friction | false | |
| Friction coefficient | f | f | |
| Expansion | expansion | false | |

**Table 3.14:** A pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=12, where a closed valve is located. Results in Figure 4.13.

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Pipe length | length | 50 | $m$ |
| Time steps | T | 100 | |
| Number of intervals | N | 12 | |
| Simulation location | $N_l$ | 12 | |
| Inlet diameter | diameter_in | 1 | $m$ |
| Outlet diameter | diameter_out | 1.1 | $m$ |
| Velocity of sound | a | 1200 | $\mathrm{m}/\mathrm{s}$ |
| Valve | closedValve | true | |
| Friction | friction | false | |
| Friction coefficient | f | f | |
| Expansion | expansion | false | |

# Chapter IV

## *Results and discussion*

This section will present results and affiliated discussions. Simulation parameters and setting are presented in section 3.3, and each simulation result will be discussed consecutively.

Simulation data have been extracted from OpenModelica and imported into MATLAB for post-processing. Note that due to an unknown bug when extracting data from OpenModelica to MATLAB, the first step in space have been excluded. This only affects the visual results related to the graphs, not the numerical results.

## 4.1 Benchmarking case: instantaneous valve closure

The following simulations are cases of instantaneous valve closure. As previously mentioned, the following simulations are all initialized by initial conditions occurring at time step $T = 1$.

### 4.1.1 Single pipe without friction

This case is simulated to check if the method of characteristics is implemented correctly, and will be compared to the theoretical benchmarking case with an identical setup.

By closing the downstream valve instantaneously at $T = 2$, one would expect to observe a square pressure pulse propagating back and forth from the valve to the reservoir. This is known as water hammer and can be seen from the point of view of the valve from Figure 4.1, which indeed confirms that the method is implemented correctly into the pipe component.

All graphs display the pressure at the valve for $T = 100$ time steps. Observe that the peak hydraulic head is constant which is a result of friction being excluded in this particular simulation.

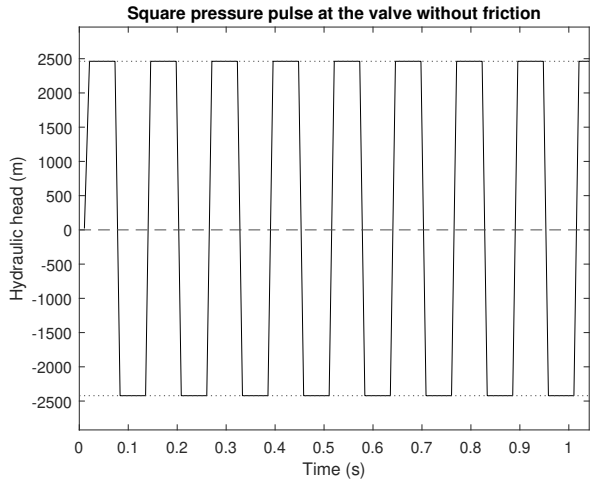



**Figure 4.1:** Square pressure pulse at the end of the pipe, where a closed valve is located. Simulation settings in Table 3.2.

All figures also display absolute pressure, and the negative pressure is therefore only valid as a numerical representation. Negative absolute pressure is nonphysical in real systems, but the dynamic behaviour is nonetheless correct in terms of change in pressure.

The initial pressure is $H = 20.3034(m)$ due to the upstream water in the reservoir, and the point of equilibrium between the upper and lower amplitude is therefore not zero. The peak surge is $H_{peak} = 2462.16(m)$ and the minimum surge is $H_{min} = -2421.55(m)$. The vertical lines are slightly sloped due to the pressure being represented as an increase from one time step to the next. The latter is most noticeable in Figure 4.3 in subsubsection 4.1.1.2.

#### 4.1.1.1 Simulation validation

To check the validity of simulations one can calculate the expected hydraulic head $H$ and reflection time $T_r$ using the Joukowsky equation and reflection time equation as shown below.

Consider Equation 2.8 where the change in velocity $\Delta v$ occurs over the characteristic change in time $\Delta t$, which is a necessary assumption when using the method of characteristics, such that:

$$\Delta H = -\frac{a^2}{g}\frac{dv}{dx}dt \tag{4.1}$$

Insert $\frac{dx}{dt} = a$ as seen in Equation 2.12, and observe that:

$$\Delta H = -\frac{a\Delta v}{g} \tag{4.2}$$

where $H(m)$ is the hydraulic head, $a(\mathrm{m/s})$ is the speed of sound, $v(\mathrm{m/s})$ is the velocity and $g(\mathrm{m/s^2})$ is the gravitational acceleration[7]. Equation 4.2 is known as the Joukowsky equation which calculates peak transient pressure when a valve is closed instantaneously. It is used to check if the hydraulic head in the simulations concurs with the calculated value.

Apply Equation 4.2 to the simulation settings given in Table 3.2, and compare with the amplitude of Figure 4.1:

$$h_{calculated} = -\frac{1200.00 \cdot 19.95}{9.81} = 2440.90(m) \tag{4.3}$$

$$h_{simulated} = -2441.86(m) \tag{4.4}$$

This means that there is an $0.039\%$ increased change in hydraulic the head from the calculated values to the simulated values. This is a negligible error which is most likely related to numerical round-off, considering the fact that Equation 4.2 is based on the same equations as the method of characteristics.

Next, consider the commonly known velocity equation which relates velocity to distance over time, $v = \frac{l}{t}$, and apply to the simulated setup. Equation 4.5 can then determine the reflection time in a pipe at an instantaneous valve closure. It is used to check if the time in the simulation concurs with the calculated value [7]:

$$T_r = \frac{2L}{a} \tag{4.5}$$

where $T_r\ (s)$ is the reflection time, $L\ (m)$ is the length of the pipe and $a\ (\mathrm{m/s})$ is the speed of sound in water. Observe that the equation is multiplied by two which represents the wave travelling back and forth in the pipe, known as the half-period.

Apply Equation 4.5 to the simulation settings given in Table 3.2, and compare with the half-period of Figure 4.1 in subsection 4.1.1:

$$T_{r,calculated} = \frac{2 \cdot 50}{1200} = \frac{1}{12} = 0.0833 \tag{4.6}$$

$$T_{r,simulated} = 0.0677 \tag{4.7}$$

This means that there is an $18.72\%$ decreased change in reflection time from the calculated values to the simulated values.

The minuscule $0.039\%$ change in the hydraulic head is negligible, however, the major change in reflection time can be a significant source of error. The latter is vulnerable to errors of the steady-state boundary conditions due to the small amount of spatial steps $N$. Both boundary conditions occupy one step each, namely the first and last. When $N = 4$, which is the setting used in the current simulation, only two steps are left which are simulated using the full method of characteristics, namely the second and third. It is therefore of interest to see if the increase of spatial steps $N$ leads to a decrease of change in the reflection time $T_r$. This will be elaborated on in subsubsection 4.1.1.3.

#### 4.1.1.2 Change of simulation location

Figure 4.1 displays the classical representation of a frictionless water hammer [8]. It acts as the main benchmarking case due to its basic dynamic behaviour, which is highly relatable to the governing equations.

In order to check if the method of characteristics is implemented correctly along the whole length of the pipe, the pressure pulse has been plotted at different sections. Figure 4.2 and Figure 4.3 shows the water hammer effect from the same simulation from the point of view of the mid-section and inlet of the same pipe, respectively.

The figures display dynamic behaviour in accordance with what is to be expected [8]. Observe how the middle section of the pipe experiences a shorter surge peak in Figure 4.2, as well as a brief horizontal distribution at $H = 20.3034(m)$.

Similarly, observe how the inlet of the pipe experiences a very brief surge peak in Figure 4.3, as well as a wider horizontal distribution at $H = 20.3034(m)$. The general tendency concerning the surge peak is that it is occurring further away in time, and appears shorter, for an increased distance from the valve. This can be explained by the fact that it takes some time before the surge is propagating from the closed valve up to the pipe inlet, and then back again.

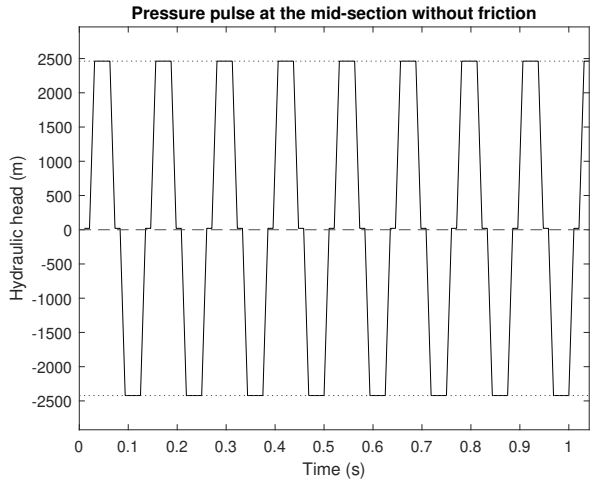



**Figure 4.2:** Pressure pulse at the mid-section of the pipe, upstream of where a closed valve is located. Simulation settings in Table 3.3.

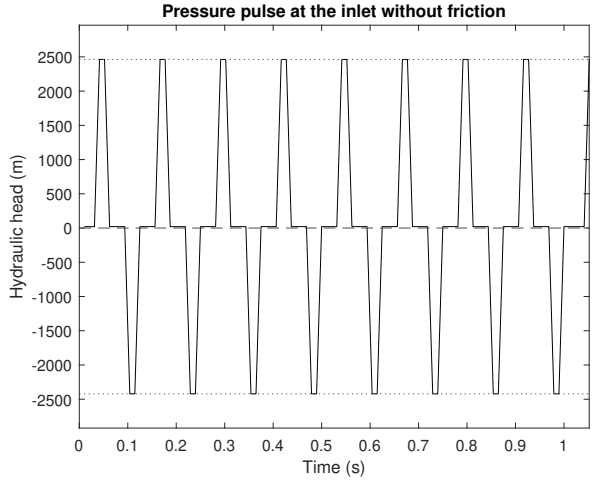



**Figure 4.3:** Pressure pulse at the inlet of the pipe, upstream of where a closed valve is located. Simulation settings in Table 3.4.

#### 4.1.1.3 Change of simulation space steps

This section will be designated to increasing the amount of spatial steps up from $N = 4$, in order to see if the change in calculated and simulated reflection time $T_r$ will diminish. Figure 4.4 and Figure 4.5 displays the square pressure surge as seen in Figure 4.1, except with $N = 8$ and $N = 12$, respectively.

First, observe how the x-axis gets smaller. This is a result of Equation 2.24 where $\Delta x$ decreases for an increasing $N$. Hence, $\Delta t$ must also decrease due to a decreasing $\Delta x$ as seen in Equation 2.25. Since the simulations are based on a static $T = 100$, the length of the x-axis is therefore reduced. This is an expected result which further strengthens the validity of the component.

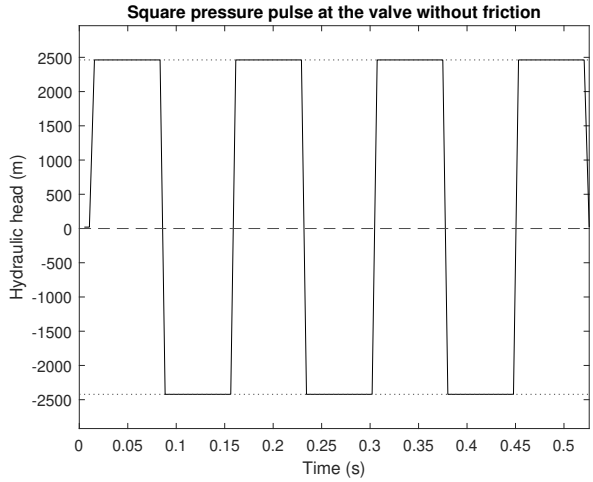



**Figure 4.4:** Square pressure pulse at the end of the pipe with N=8, where a closed valve is located. Simulation settings in Table 3.5.

Next, observe from Table 4.1 that the decrease from $T_{r,calculated}$ to $T_{r,simulated}$ diminishes for a increasing $N$. This confirms the hypothesis that the model is more accurate from an increasing amount of spatial steps. As previously mentioned, this is likely due to the decreasing impact of the steady-state boundary conditions. However, an increase of $N$ will also lead to a longer simulation time, which is undesirable. Nonetheless, Table 4.1 clearly shows that the simulations should run on longer spatial steps $N$, to increase the accuracy of the model.

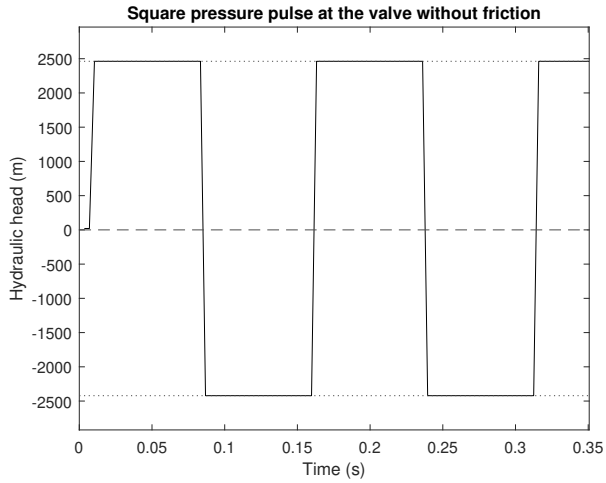



**Figure 4.5:** Square pressure pulse at the end of the pipe with N=12, where a closed valve is located Simulation settings in Table 3.6.

**Table 4.1:** Comparison between $T_{r,calculated} = 0.0833(s)$ and $T_{r,simulated}$ for different spatial step N.

| $N$ | $T_{r,simulated}$ | Decrease from $T_{r,calculated}$ |
|---|---|---|
| 4 | $0.0677(s)$ | $18.72\%$ |
| 8 | $0.0729(s)$ | $12.48\%$ |
| 12 | $0.0764(s)$ | $8.28\%$ |

### 4.1.2 Single pipe with friction

This section will simulate cases to check if friction is implemented correctly. First, it is desirable to see first-hand that the hydraulic head $H$ is decreasing along the pipe due to friction. Figure 4.6 displays a linear pressure drop acting in the initial stages of the simulation at $T = 1$. Observe how the x-axis represents the length of the pipe, contradictory to every other simulation where the point of view has thus far been stationary.

Note that even though all cases in this thesis are set in a pipe where a valve is closed instantaneously, this has not occurred in the case shown in Figure 4.6. Instead, the figure displays how the initial conditions have correctly implemented friction using Equation 2.28.

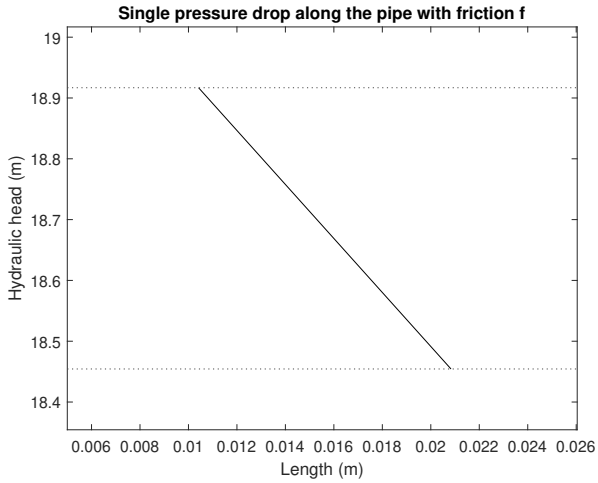



**Figure 4.6:** Pressure drop along the pipe from inlet to the closed valve at $T = 1$. Simulation settings in Table 3.7.

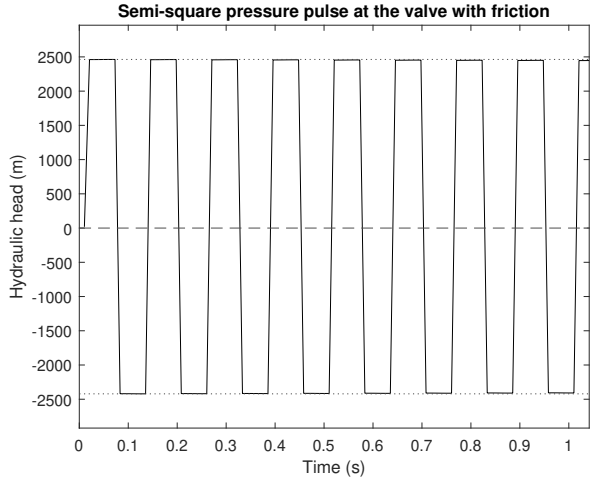



**Figure 4.7:** Pressure surges with friction at the end of a pipe, where a closed valve is located. Simulation settings in Table 3.8.

Now, by closing the downstream valve instantaneously at again time step $T = 2$, one would expect to observe a semi-square pressure pulse propagating back and forth from the valve to the reservoir.

At first glance, Figure 4.7 seems interchangeable from the frictionless square pressure pulse in Figure 4.1. At the very least, this means that the core of the method of characteristics is not corrupted by the inclusion of friction.

It is hypothesized that friction will impact the simulated results insignificantly. To this effect, follow-up simulations have been performed to enhance the visual effect of friction, where the friction factor has been multiplied by one hundred, i.e., $f^* = 100 \cdot f$ where $f$ is the correct friction factor. These simulations are discussed in the following section.

#### 4.1.2.1 Change of friction coefficient

The simulation results in this section have an increased friction factor to enhance the visual effects while assessing the validity of the library.

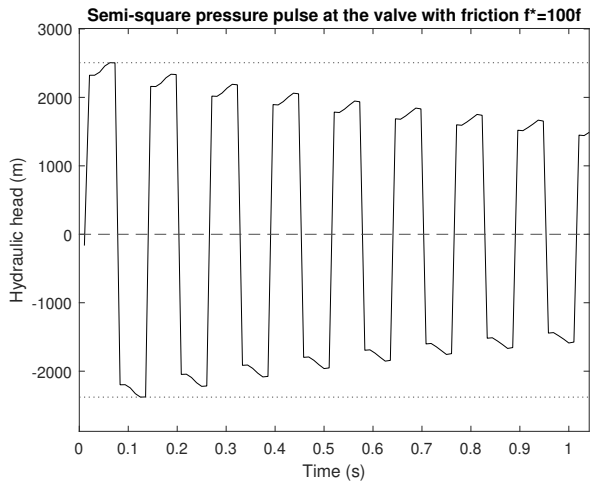



**Figure 4.8:** Pressure surges with friction at the end of a pipe with N=4, where a closed valve is located. Simulation settings in Table 3.9.

Figure 4.8 displays a change in values and behaviour. The pressure pulse is still semi-square, and overall the hydraulic head is decreasing for increasing time, as expected. In practice, this means that the hydraulic head is less than what it was when the valve originally closed down, merely due to friction. It can be deduced

that the hydraulic head would reach zero for a long period of time, naturally without an inflow of water from the reservoir after valve closure.

Observe how the hydraulic head is increasing and decreasing piece-wise linearly for each semi-square pulse on the positive and negative side of the y-axis respectively. These slopes are results of the pressure surge moving upwards in the pipe, all the while the hydraulic head is larger further upstream of the pipe.

The pressure increase at the valve is due to the friction in the upstream pipe disappearing when the flow is stopped by the initiated pressure rise propagating upstream. This phenomenon is also known as line-packing.

It was hypothesized that the increase of hydraulic head at each semi-square pulse would be linear, which is not the case seen from Figure 4.8. The head appears to start and finish off with a section of non-changing head, while the section in-between is increasing linearly. This could be a numerical error or an error with the boundary conditions. Note that the method of characteristics is implemented piece-wise, and not continuously at the boundaries. More specifically, it is only the hydraulic head $H$ and volumetric flow rate $Q$ which is transferred between boundaries, and not the characteristics themselves. If implemented without simplifications, the plus characteristic should be implemented at the left boundary and the minus characteristic at the right boundary. This could also be a source of error which could lead to faulty boundary conditions as seen in Figure 4.8. Other possible sources of error could be that the boundary conditions are calculated based on the steady-state inlet parameters.

Figure 4.9 and Figure 4.10 clearly displays that the hypothesized boundary condition errors which act as horizontal pressure surges are less prevalent in simulations with an increasing amount of spatial steps $N$. The slopes are smoother due to more spatial steps, but still retain the horizontal head development at the beginning and end of the slope. This confirms that the method is more accurate for a larger number of $N$ when friction is included.

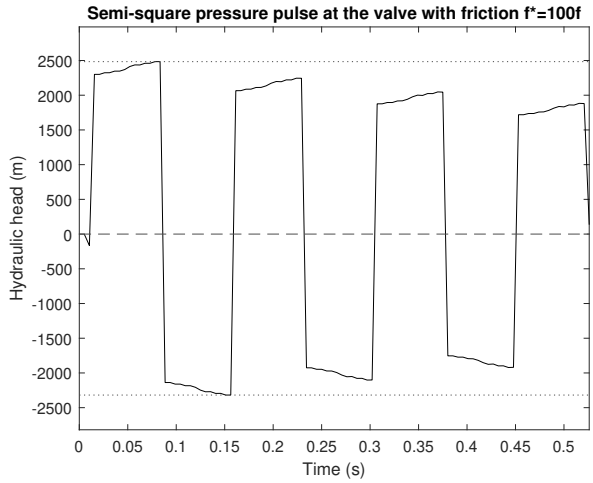



**Figure 4.9:** Pressure surges with friction at the end of a pipe with N=8, where a closed valve is located. Simulation settings in Table 3.10.

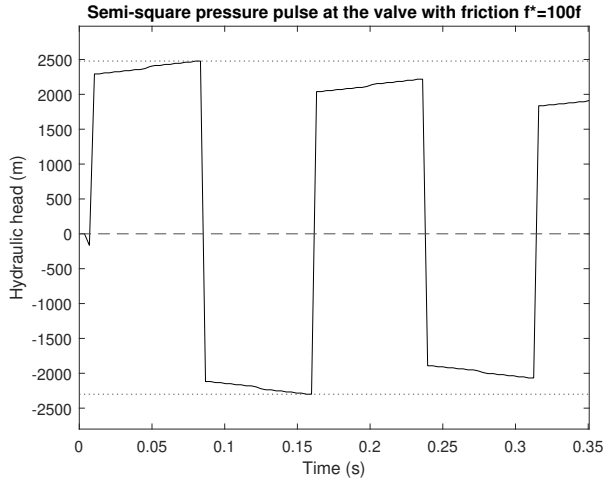



**Figure 4.10:** Pressure surges with friction at the end of a pipe with N=12, where a closed valve is located. Simulation settings in Table 3.11.

### 4.1.3 Single pipe with a non-uniform cross-sectional area without friction

This section utilizes Equation 2.26 and Equation 2.27 in order to simulate a frictionless pipe with a non-uniform cross-sectional area. It has previously been observed that the accuracy of simulation results vastly depends on space step $N$. In addition, it is easier to understand the behaviour of the pipe component when varying $N$, due to its unique synergy with the boundary conditions.

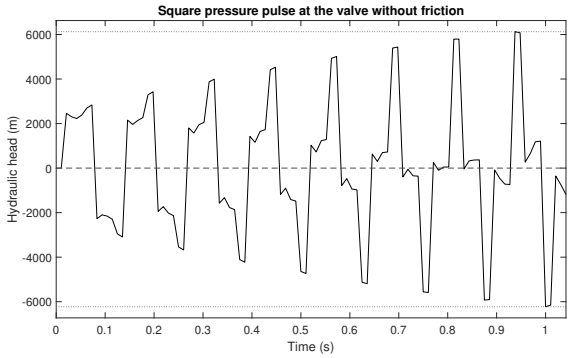



**Figure 4.11:** Pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=4, where a closed valve is located. Simulation settings in Table 3.12.

First of all, the results from these simulations are faulty and nonphysical. The first indicator of this is the change in dynamic behaviour over time. Due to the absence of friction, all pressure pulses should display the same dynamic behaviour, which is not the case.

Secondly, the peak hydraulic head appears to increase over time. Simultaneously, the initial values of each pulse appear to decrease over time. Both of these values should be non-changing. As with all simulations, Figure 4.13 where $N = 12$ appear to be the most stable, even though the dynamic behaviour here is also non-physical. These simulations show that the library is currently incapable of simulating pipes with non-uniform cross-sectional area.

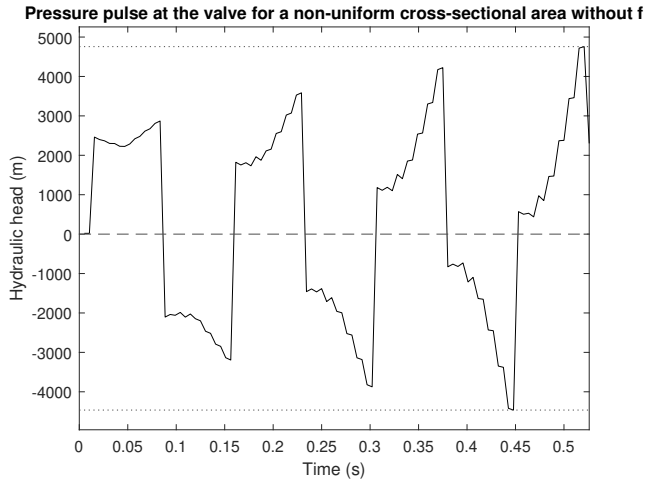



**Figure 4.12:** Pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=8, where a closed valve is located. Simulation settings in Table 3.13.

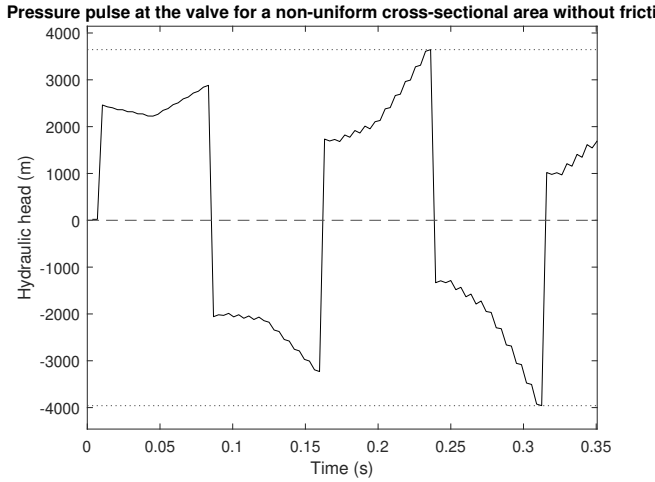



**Figure 4.13:** Pressure pulse at the end of a single pipe with a non-uniform cross-sectional area with N=12, where a closed valve is located. Simulation settings in Table 3.14.

## 4.2 Overall observations

The results obtained has thus far been expected, except for the pipes with non-uniform cross-sectional areas in subsection 4.1.3. The graphs display typical dynamic behaviour, and can in most cases easily be compared to benchmarking cases. The classical representation of the frictionless water hammer effect as seen in Figure 4.1 is the simplest example of this.

The simulation results were validated in subsubsection 4.1.1.1 and were in accordance with the Joukowsky equation and the reflection time equation, with some exceptions. Naturally, the results are more precise when simulating with a larger spatial step $N$, which revealed that the boundary conditions represent a source of error. When comparing results for $N \in [4, 8, 12]$ it was revealed that the difference between simulated reflection time $T_{r,simulated}$ and calculated reflection time $T_{r,calculated}$ significantly diminishes for an increasing $N$, at the cost of an increased simulation real-time. Depending on the simulation which would be performed with the library, it would be beneficial for the user to strive after a minimum value of $N = 8$, and as large as possible, within the acceptable simulation time span available to the user.

Simulations are limited to the simple geometry available in the parameter settings of the pipe model. It has been revealed that neither the option for non-uniform cross-sectional area nor the option for connecting pipes in series is currently not functional. These are major limitations to the library which reduces the capabilities for revealing the system dynamics of hydraulic laboratory systems.

Despite limitations and errors within the pipe component, a full library utilizing the method of characteristics for pipe flow has nonetheless been created. The first research objective related to the pipe component itself, including blind flanges, is also completed. The pipes are easily customizable using the user-friendly graphical user interface in OpenModelica.

The second research objective has only partially been completed. The initial case as shown in Figure 1.1 in section 1.1 was not achieved. However, the method of characteristics has successfully been implemented into the pipe component, which enables the library to be applied to other sections of interest in hydraulic laboratory systems.

The last research objective related to the comparison between simulated cases and benchmarking cases has been fulfilled in its entirety. Each case has contributed to a greater understanding of the capabilities, as well as illuminating the advantages and disadvantages, of the library.

Errors related to the simulation results, especially concerning the boundary conditions, decreases the scope of applicability of the library. Additionally, due to Modelica-related errors, pipes are not able to be coupled in series. It is unknown if the latter is related to the boundary conditions, which introduces an element of uncertainty.

Simulations have been within the scope of the research objectives. Due to the time restriction, these errors have not been fully ascertained. Suggestions for further work, especially related to known errors and incomplete simulations, is presented in chapter 6.

# Chapter V

## *Conclusion*

A library named *OpenWPL* has been created using the method of characteristics. It was created in the open-source programming language Modelica using the software OpenModelica. As such, the first research question has been satisfied.

The simulated results have been compared to benchmarking cases of instantaneous valve closure. The comparison revealed that the pipe component is capable of simulating the classic representation of a frictionless water hammer along the whole length of the pipe as a square pressure pulse. Hence the second and third research questions have been investigated.

The simulations were validated using the Joukowsky equation and the reflection time equation, and investigations revealed that there are non-negligible errors related to the boundary conditions. Also, friction was successfully implemented, although significantly more accurate for higher resolution in space.

In conclusion, *OpenWPL* is currently not capable of revealing the system dynamics of hydraulic laboratory systems. Rather, the library represents a good foundation that can be further developed into a more complex and full-fledged library in the future.

# Chapter VI

## *Further work*

The following chapter lists the most prominent suggestions for further work on the library which would increase its preciseness and applicability. All suggestions in this chapter were not carried through due to the restriction of time.

Most notably, it would be beneficial to fully implement the functions of connecting pipes in series and the implementation of pipes with non-uniform cross-sectional areas.

The former would enable the possibility of simulating the case as shown in Figure E.1 in Appendix E. The problem at hand, according to the error message as shown in OpenModelica, is related to the imbalance of equations and variables during the pre-optimization of the model. Although the system is balanced when using OpenModelica's built-in function *Check Model* as seen in Figure F.1, the system is deemed under-determined during simulation as seen in Figure F.2 and Figure F.3, all in Appendix F. This may be a result of imbalanced equations within some part of the pipe model, or perhaps external functions that are used within. Further work on this issue could potentially lead to the currently biggest advancement of the library, namely the possibility to connect pipes in series.

The latter, i.e. the problems with the implementation of pipes with non-uniform cross-sectional areas, is most likely related to faulty boundary conditions. It is currently not clear as to why, or even if, the boundary conditions are not correctly implemented. The only observation which relates to this effect is that several simulations, e.g. Table 4.1, Figure 4.8 and Figure 4.12, display certain strange elements where the boundary conditions could be the origin. A possible solution to this problem could be to fully implement the method of characteristics at the boundaries as well, i.e., obtain a plus characteristic at *port_a* at the left boundary and transfer the minus characteristic at *port_b* at the right boundary.

There are several other suggestions for further work which could prove beneficial to the library, however, this chapter will finish with only a few mentions. First of all, it would be beneficial if there was full compatibility between *OpenWPL* and the MFL. This would enable the use of many useful components available within the Modelica Fluid Library. Another step along the way would then be to fully develop the MOC component. Currently, the values of $N$ and $T$ must be set manually, but, if several pipes would be coupled in series it would be useful that both $N$ and $T$ would be computed automatically based on the shortest pipe and simulation time set in Modelica, respectively.

Secondly, it would prove useful to design other hydropower-related components neither available in the MFL nor OpenWPL, such as surge shafts and turbines. This would enable the user of the library to simulate real-case scenarios of hydraulic laboratory systems.

Lastly, the transient effects in the pipe component are simulated using a quasi-steady friction model, which was an assumption used to implement friction in Equation 2.3. Ideally, a model simulating unsteady friction should be included, in addition to the quasi-steady simplification implemented in the pipe component.

# *References*

[1] Hinna, J. T., 2020, "Modelica model of the Waterpower Laboratory," Project work, Norwegian University of Science and Technology, Trondheim.

[2] Wylie, E. B. and Streeter, V. L., 1993, *Fluid Transients in Systems*, Prentice Hall, Englewood Cliffs, NJ.

[3] Çengel, Y. A. and Cimbala, J. M., 2014, *Fluid Mechanics: Fundamentals and applications*, 3rd ed., McGraw-Hill Education, Boston.

[4] Holm Aftret, A., 2017, "Simulation and analysis of FCR operation of a Francis turbine," Master's thesis, Norwegian University of Science and Technology, Trondheim, accessed 2021-03-05, https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2454894

[5] Otter, M., 2013, "Modelica Overview," accessed 2020-11-13, https://modelica.org/education/educational-material/lecture-material/english/ModelicaOverview.pdf/view.html

[6] 2021, "Modelica Language Specification," accessed 2021-05-14, https://specification.modelica.org/master/MLS.html

[7] Nielsen, T., 1990, "Dynamisk dimensjonering av vannkraftverk," Tech. Rep. STF67 A 90038, SINTEF, ISBN: 82-595-5952-8.

[8] Storli, P.-T., 2010, "Transient friction in pressurized pipes; the wate rhammer phenomenon," phd, Norwegian University of Science and Technology, Trondheim.

# Appendix - A

## *Full calculation: the equation of motion for transient flow*

This appendix will show calculations in regards to the simplification from Equation 2.3 to Equation 2.4 [2].

$$\frac{p_x}{\rho} + VV_x + V_t + g\sin\alpha + \frac{fV|V|}{2D} = 0 \qquad \text{(2.3 revisited)}$$

Consider Equation 2.3 for steady flow, i.e. $V_t = 0$, as a special case of unsteady flow.

$$\frac{p_x}{\rho} + VV_x + g\sin\alpha + \frac{fV|V|}{2D} = 0 \qquad \text{(2.3a)}$$

Assume also constant fluid density and pipe area, i.e. $V_x = 0$.

$$\Delta p = -\rho g \Delta x \sin\alpha - \frac{\rho f \Delta x V|V|}{2D} = 0 \qquad \text{(2.3b)}$$

The latter equation is equivalent to the Darcy-Weisbach equation, and it is therefore consistent to exclude the term $VV_x$, which is a common simplification for unsteady flows. This leads to

$$\frac{p_x}{\rho} + g\sin\alpha + \frac{fV|V|}{2D} = 0 \qquad \text{(2.3c)}$$

where the pressure term may be replaced with the piezometric head $H$ and elevation $z$ at position $x$.

$$\frac{\rho g(H_x - \sin\alpha)}{\rho} + g\sin\alpha + \frac{fV|V|}{2D} = 0 \tag{2.3d}$$

Which subsequently leads to

$$gH_x + V_t + \frac{fV|V|}{2D} = 0 \tag{2.4 revisited}$$

# Appendix - B

## *Full calculation: the continuity equation for transient flow*

This appendix will show calculations in regards to the simplification from Equation 2.7 to Equation 2.8 [2].

$$\frac{\dot{A}}{A} + \frac{\dot{\rho}}{\rho} + V_x = 0 \qquad \text{(2.7 revisited)}$$

The second term in the latter equation can be replaced using the definition of bulk modulus of elasticity for a fluid.

$$\frac{\dot{\rho}}{\rho} = \frac{\dot{p}}{K} \qquad \text{(2.7a)}$$

Similarly the first term can be expressed as such

$$\dot{A} = \frac{dA}{dp}\dot{p} \qquad \text{(2.7b)}$$

Substituting Equation 2.7a and Equation 2.7b into Equation 2.7 yields

$$V_x = \frac{\dot{p}}{K}\left(1 + \frac{K}{A}\frac{dA}{dp}\right) = 0 \qquad \text{(2.7c)}$$

which can be re-written into

$$\rho a^2 V_x + \dot{p} = 0 \tag{2.7d}$$

where the wave velocity is given as

$$a^2 = \frac{{}^{\mathrm{K}}/_{\rho}}{1 + ({}^{\mathrm{K}}/_{\mathrm{A}})({}^{\Delta \mathrm{A}}/_{\Delta \mathrm{p}})} \tag{2.9 revisited}$$

Consider Equation 2.7d for steady flow where $p_t = 0$.

$$\rho a^2 V_x + V p_x = 0 \tag{2.7e}$$

Next, consider $V_x = 0$ where density and tube area variations are not considered in steady flow of compressible fluids. In turn, this means that $p_x = 0$. Substitute Equation 2.3 into Equation 2.7d, and eliminate $V_x$.

$$p_x \left(1 - \left(\frac{V}{a}\right)^2\right) + \frac{V}{a^2} p_t + \rho V_t + \rho g \sin\alpha + \rho \frac{f V^2}{2D} = 0 \tag{2.7f}$$

The latter equation can be simplified by removing the term $({}^{\mathrm{V}}/_{\mathrm{a}})^2$ for low Mach numbers.

$$\rho a^2 V_x + p_t = 0 \tag{2.7g}$$

Finally, by replacing pressure using $p_t = \rho g H_t$, yields

$$\frac{a^2 V_x}{g} + H_t = 0 \tag{2.8 revisited}$$

# Appendix - C

## *Full calculation: the method of characteristics*

This appendix will shown the calculation of the method of characteristics which starts with Equation 2.10, and yields Equation 2.22 and Equation 2.23.

$$L = L_1 + \lambda L_2 = 0 \qquad \text{(2.10 revisited)}$$

Substitution of Equation 2.4 and Equation 2.8 into Equation 2.10 yields:

$$L = L_1 + \lambda L_2 = \lambda \left( H_x \frac{g}{\lambda} + H_t \right) + \left( V_x \lambda \frac{a^2}{g} + V_t \right) + \frac{fV|V|}{2D} = 0 \qquad \text{(2.10a)}$$

Variables $V$ and $H$ are functions of both $x$ and $t$ which, for any two real and distinct values of $\lambda$, means that:

$$\frac{dH}{dt} = H_t \frac{dx}{dt} + H_t \qquad \text{(2.10b)}$$

$$\frac{dV}{dt} = V_x \frac{dx}{dt} + V_t \qquad \text{(2.10c)}$$

Observe that if

$$\frac{dx}{dt} = \frac{g}{\lambda} = \frac{\lambda a^2}{g} \qquad \text{(2.10d)}$$

then Equation 2.10a becomes

$$\lambda\frac{dH}{dt}+\frac{dV}{dt}+\frac{fV|V|}{2D}=0 \tag{2.10e}$$

which yields

$$\lambda=\pm\frac{g}{a} \tag{2.11 revisited}$$

Now substitute Equation 2.11 into Equation 2.10d which yields

$$\frac{dx}{dt}=\pm a \tag{2.11a}$$

2.11a refers to the change in position in relation to the change in time by the wave velocity $a$. The substitution of the set of equations given in 2.11a into Equation 2.10d yields

$$C^{+}:\begin{cases}\dfrac{g}{a}\dfrac{dH}{dt}+\dfrac{dV}{dt}+\dfrac{fV|V|}{2D}=0\\[2ex]\dfrac{dx}{dt}=a\end{cases} \tag{2.12 revisited}$$

$$C^{-}:\begin{cases}-\dfrac{g}{a}\dfrac{dH}{dt}+\dfrac{dV}{dt}+\dfrac{fV|V|}{2D}=0\\[2ex]\dfrac{dx}{dt}=-a\end{cases} \tag{2.13 revisited}$$

These equations are known as the characteristic equations. In order to obtain the finite difference equations one must integrate along the characteristic lines. Now multiply the first part of Equation 2.12 by $a\frac{dt}{g}=\frac{dx}{g}$, and integrate along the positive characteristic line from point $A$ to point $P$:

$$\int_{H_A}^{H_P}dH+\frac{a}{gA}\int_{Q_A}^{Q_P}dQ+\frac{f}{2gDA^2}\int_{x_A}^{x_P}Q|Q|\,dx=0 \tag{2.12a}$$

Now using integration by parts:

$$\int_{x_A}^{x_P} Q^2\,dx = Q^2x\Big|_{X_A}^{X_P} - \int_{x_A}^{x_P} x\,dQ^2 = Q^2x\Big|_{x_A}^{x_P} - 2\int_{x_A}^{x_P} xQ\,dQ$$
$$\approx Q_P^2 x_P - Q_A^2 x_A - 2\left[\frac{x_P Q_P + x_A + Q_A}{2}(Q_P - Q_A)\right]$$
$$\approx Q_P|Q_A|(x_P - x_A) \tag{2.12b}$$

The same calculations can be performed for Equation 2.13. By completing the integration for Equation 2.12a, and doing the same for the minus characteristic, one achieves:

$$H_P - H_A + \frac{a}{gA}(Q_P - Q_A) + \frac{f\Delta x}{2gDA^2}Q_P|Q_A| = 0 \tag{2.12c}$$

$$H_P - H_B + \frac{a}{gA}(Q_P - Q_B) + \frac{f\Delta x}{2gDA^2}Q_P|Q_B| = 0 \tag{2.13c}$$

Solving for $H_P$ yields:

$$C^+ : H_P = H_A - B(Q_P - Q_A) - RQ_P|Q_A| \tag{2.12d}$$

$$C^+ : H_P = H_B - B(Q_P - Q_B) + RQ_P|Q_B| \tag{2.13d}$$

where it is now obvious that Equation 2.20 and Equation 2.21 are coefficients of Equation 2.12d and Equation 2.13d:

$$B = \frac{a}{gA} \tag{2.20 revisited}$$

$$R = \frac{f\Delta x}{2gDA^2} \tag{2.21 revisited}$$

As seen in section 2.2, Equation 2.12d and Equation 2.13d can be simplified to:

$$C^+ : H_i = C_P - B_P Q_i \tag{2.14 revisited}$$

$$C^- : H_i = C_M - B_M Q_i \qquad \text{(2.15 revisited)}$$

with constants

$$C_P = H_{i-1} + BQ_{i-1} \qquad \text{(2.17 revisited)}$$

$$C_M = H_{i+1} - BQ_{i+1} \qquad \text{(2.16 revisited)}$$

and

$$B_P = B + R|Q_{i-1}| \qquad \text{(2.18 revisited)}$$

$$B_M = B + R|Q_{i+1}| \qquad \text{(2.19 revisited)}$$

Finally Equation 2.22 is found by eliminating $Q_i$ from Equation 2.14 and Equation 2.15. Equation 2.23 is found by inserting $H_i$ from Equation 2.22 into either Equation 2.14 or Equation 2.15:

$$H_i = \frac{C_P B_M + C_M B_P}{B_P + B_M} \qquad \text{(2.22 revisited)}$$

$$Q_i = \frac{C_P - C_M}{B_P + B_M} \qquad \text{(2.23 revisited)}$$

# Appendix - D

## *Diagram view of a setup from source to sink*

This setup is used in several cases when simulating the instantaneous valve close.

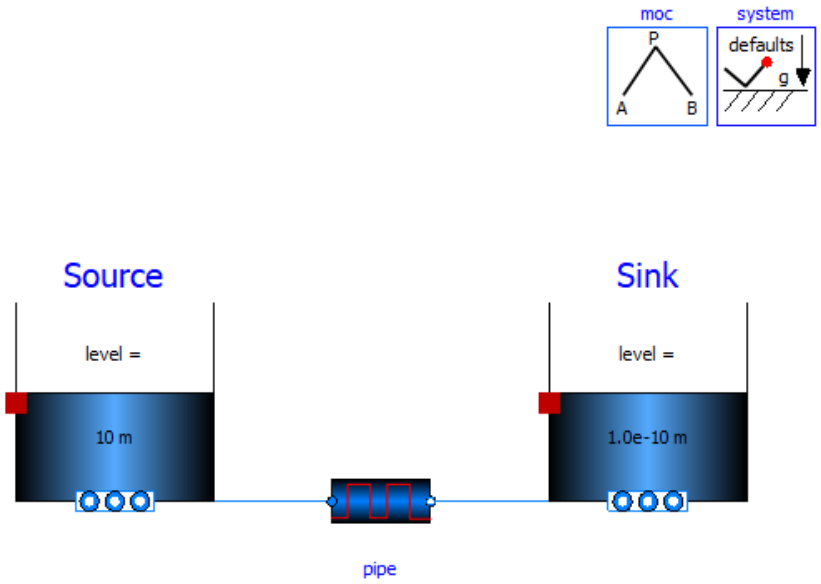



**Figure D.1:** A simple setup from source to sink via a single pipe.

# Appendix - E

## *Diagram view of source to sink with intersection and blind flange simulation*

This setup did not compile. The intention was to observe the system dynamics in pipe 1 and pipe 2 when the valve at pipe 3 closed.

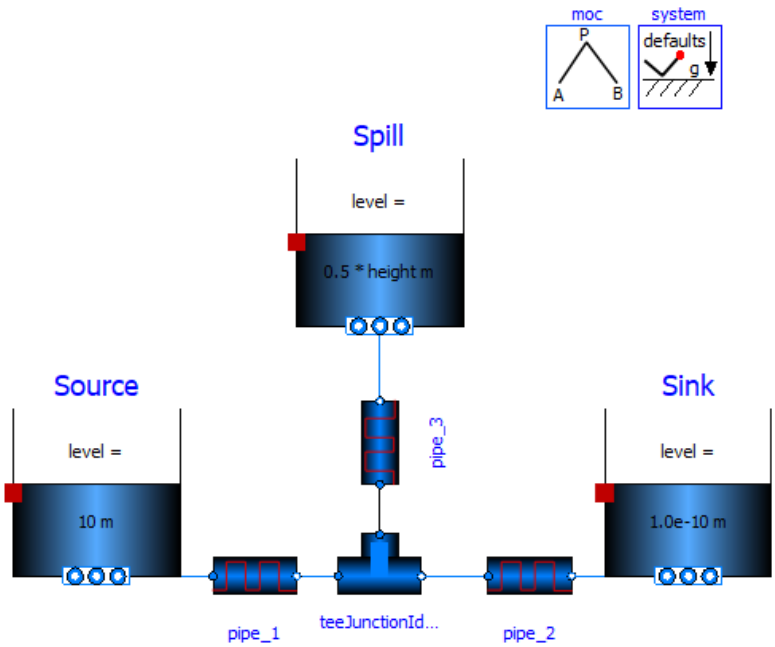



**Figure E.1:** A setup from source to sink with intersection and blind flange simulation.

# Appendix - F

## *Errors related to: source to sink with intersection and blind flange simulation*

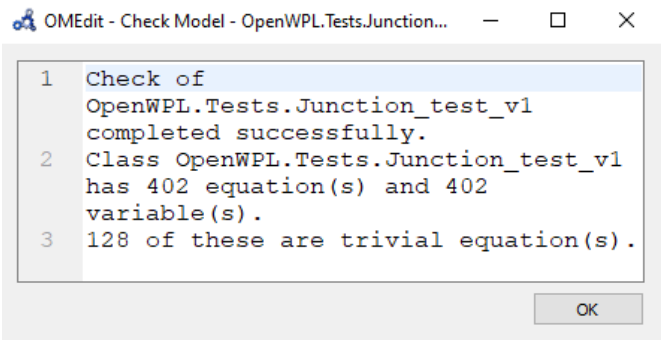



**Figure F.1:** Balanced *Check Model* message.

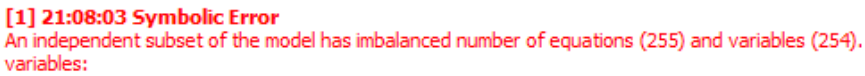



**Figure F.2:** Imbalanced symbolic error.

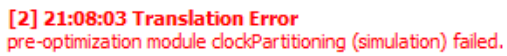



**Figure F.3:** Imbalanced translation error.

# Appendix - G

## *Modelica code: functions implemented into OpenWPL*

```
within OpenWPL;

package Functions "External functions"
  extends Modelica.Icons.FunctionsPackage;
  annotation (
    version="1.0.0",
    versionDate="08.06.21",
    Protection(access = Access.packageDuplicate),
    preferredView="info",
    Documentation(info="<html>
<p> A package of external functions which is used in
   the library. Most functions are based on the
   method of characteristics. </p>
</html>"),
  Icon);

  function Re
    extends Modelica.Icons.Function;
    input Modelica.SIunits.Velocity v "Flow velocity"
       ;
    input Modelica.SIunits.Diameter D "Pipe diameter"
       ;
    input Modelica.SIunits.Density rho "Density";
    input Modelica.SIunits.DynamicViscosity mu "
       Dynamic viscosity of water";
    output Modelica.SIunits.ReynoldsNumber Re_D "
       Reynolds number";
  algorithm
```

```
  Re_D := (rho*abs(v)*D)/mu;
end Re;

function f_Haaland
  extends Modelica.Icons.Function;
  input Modelica.SIunits.Velocity v "Flow velocity"
     ;
  input Modelica.SIunits.Diameter D "Pipe diameter"
     ;
  input Modelica.SIunits.Density rho "Density";
  input Modelica.SIunits.DynamicViscosity mu "
     Dynamic viscosity of water";
  input Modelica.SIunits.Height eps "Pipe roughness
      height";
  output Real f "Friction factor";
protected
  Modelica.SIunits.ReynoldsNumber Re_D "Reynolds
     number";
algorithm
  Re_D := Re(v, D, rho, mu);
  f :=  100*(1/(-1.8*log(((eps/D)/3.7)^(1.11)+(6.9/
     Re_D))))^2;
end f_Haaland;

function B_Impedance
  extends Modelica.Icons.Function;
  input Modelica.SIunits.Acceleration g "
     Graviational accelleration";
  input Modelica.SIunits.VelocityOfSound a "
     Velocity of sound in water";
  input Modelica.SIunits.Area A "Area";
  output Real B;
algorithm
  B := a/(g*A);
end B_Impedance;

function R_Resistance
  extends Modelica.Icons.Function;
  input Real f "Friction factor";
  input Modelica.SIunits.Acceleration g "
     Graviational accelleration";
```

```
  input Modelica.SIunits.Diameter D "Pipe diameter"
     ;
  input Modelica.SIunits.Area A "Pipe area";
  input Modelica.SIunits.Distance dx;
  output Real R;
algorithm
  R := f * dx / (2 * g * D * A ^ 2);
end R_Resistance;

function N_parts
  extends Modelica.Icons.Function;

  input Modelica.SIunits.Length L;
  input Modelica.SIunits.Length dx;
  output Real N;
algorithm
  N := L/dx;
end N_parts;

function T_step
  extends Modelica.Icons.Function;

  input Modelica.SIunits.Length dx;
  input Modelica.SIunits.Velocity a;
  output Modelica.SIunits.Time dt;
algorithm
  dt := dx/a;
end T_step;

function C_plus
  extends Modelica.Icons.Function;

  /*
  input Real Hb;
  input Real B;
  input Real Qb;
  input Real R;
  output Real Cp;
  */

  input Real Ha "Hydraulic head at point A";
```

```
    input Real B "Impedance";
    input Real Qa "Volumetric flow at point A";
    output Real Cp "Plus characteristic given in
       equation (3-23) in Wylie & Streeter";
  algorithm
//Cp := Hb - B*Qb + R*Qb*abs(Qb);
    Cp := Ha + B*Qa;
  end C_plus;

  function C_minus
    extends Modelica.Icons.Function;

    /*
    input Real Ha;
    input Real B;
    input Real Qa;
    input Real R;
    output Real Cm;
    */

    input Real Hb "Hydraulic head at point B";
    input Real B "Impedance";
    input Real Qb "Volumetric flow at point B";
    output Real Cm "Minus characteristic given in
       equation (3-24) in Wylie & Streeter";
  algorithm
//Cm := Ha + B*Qa - R*Qa*abs(Qa);
    Cm := Hb - B*Qb;
  end C_minus;

  function H_head
    extends Modelica.Icons.Function;
    input Real Cp "Plus characteristic given in
       equation (3-24) in Wylie & Streeter";
    input Real Cm "Minus characteristic given in
       equation (3-24) in Wylie & Streeter";
    input Real Bm "Impedance at Cm given in equation
       (3-24) in Wylie & Streeter";
    input Real Bp "Impedance at Cp given in equation
       (3-23) in Wylie & Streeter";
```

```
  output Real H "Hydraulic head given in equation
      (2-25) in Wylie & Streeter";
algorithm
  H := (Cp*Bm+Cm*Bp)/(Bp+Bm);
end H_head;

function Q_flow
  extends Modelica.Icons.Function;
  input Real Cp "Plus characteristic given in
      equation (3-24) in Wylie & Streeter";
  input Real Cm "Minus characteristic given in
      equation (3-24) in Wylie & Streeter";
  input Real Bm "Impedance at Cm given in equation
      (3-24) in Wylie & Streeter";
  input Real Bp "Impedance at Cp given in equation
      (3-23) in Wylie & Streeter";
  output Real Q "Volumetric flow given in equation
      (2-26) in Wylie & Streeter";
algorithm
  Q := (Cp-Cm)/(Bp+Bm);
end Q_flow;

function v_Velocity
  extends Modelica.Icons.Function;
  input Real H "Hydraulic head";
  input Modelica.SIunits.Acceleration g "
      Graviational accelleration";
  output Modelica.SIunits.Velocity v "Flow velocity
      ";
algorithm
  v := sqrt(2*g*abs(H));
end v_Velocity;

function H_expansion
  extends Modelica.Icons.Function;
  input Real Ha;
  input Modelica.SIunits.Acceleration g "
      Graviational accelleration";
  input Modelica.SIunits.VelocityOfSound a "
      Velocity of sound in water";
  input Modelica.SIunits.Area Aa "Area at point A";
```

```
  input Modelica.SIunits.Area A "Area at point P";
  input Real Qa "Volumetric flow at point A";
  input Real Q "Volumetric flow at point P";
  output Real H "Hydraulic head";
algorithm
  H := Ha - (a/g)*(Q/A-Qa/Aa-((Q+Qa)/2)*(1/A - 1/Aa
     ));
end H_expansion;

function Q_expansion
  extends Modelica.Icons.Function;
  input Real Ha;
  input Real Hb;
  input Modelica.SIunits.Acceleration g "
     Graviational accelleration";
  input Modelica.SIunits.VelocityOfSound a "
     Velocity of sound in water";
  input Modelica.SIunits.Area Ab "Area at point B";
  input Modelica.SIunits.Area Aa "Area at point A";
  input Modelica.SIunits.Area A "Area at point P";
  input Real Qa "Volumetric flow at point A";
  input Real Qb "Volumetric flow at point B";
  output Real Q "Hydraulic head";
algorithm
Q := ((g/a)*(Ha-Hb) + Qb*(1/(2*A) + 1/(2*Ab)) + Qa
    *(1/(2*A) + 1/(2*Aa)))/(1/A + 1/(2*Ab) + 1/(2*Aa
    ));
end Q_expansion;

function H_initial
extends Modelica.Icons.Function;
  input Real f "Friction factor";
  input Real H_res "Head at reservoir";
  input Real Q "Volumetric flow rate";
  input Modelica.SIunits.Length L;
  input Modelica.SIunits.Acceleration g "
     Graviational accelleration";
  input Modelica.SIunits.Diameter D "Pipe diameter"
     ;
  input Modelica.SIunits.Area A "Pipe area";
  output Real H;
```

```
algorithm
  H := H_res - f*(L/D)*abs(Q)*Q/(2*g*A^2);
end H_initial;

function v_Bernoulli
  extends Modelica.Icons.Function;
  input Modelica.SIunits.Height H_in "H. head at
     reservoir";
  input Modelica.SIunits.Height H_out "H. head at
     sink";
  input Modelica.SIunits.Diameter D "Pipe diameter"
     ;
  input Modelica.SIunits.Acceleration g "
     Graviational accelleration";
  input Modelica.SIunits.Distance L "Length of pipe
     ";
  input Real k "loss coefficient";
  input Real f "Friction factor";
  output Real v "Flow velocity";
algorithm
  v :=  sqrt((H_in-H_out)/((2*g)*(k+f*L/D)));
end v_Bernoulli;

function f_Bernoulli
  extends Modelica.Icons.Function;
  input Modelica.SIunits.Height H_in "H. head at
     reservoir";
  input Modelica.SIunits.Height H_out "H. head at
     sink";
  input Modelica.SIunits.Diameter D "Pipe diameter"
     ;
  input Modelica.SIunits.Acceleration g "
     Graviational accelleration";
  input Modelica.SIunits.Distance L "Length of pipe
     ";
  input Real v "Friction factor";
  output Real f "Flow velocity";
algorithm
  f :=  (2*D*g*(H_in-H_out))/(L*V^2);
end f_Bernoulli;
```

```
function H_f
extends Modelica.Icons.Function;
  input Modelica.SIunits.Diameter D "Pipe diameter"
     ;
  input Modelica.SIunits.Acceleration g "
     Graviational accelleration";
  input Modelica.SIunits.Distance L "Length of pipe
     ";
  input Modelica.SIunits.Velocity V;
  input Real f "Friction factor";
  output Real H;
algorithm
  H := f*(L/D)*((V^2)/(2*g));
end H_f;

function v_iterate
  extends Modelica.Icons.Function;
  input Modelica.SIunits.Diameter D "Pipe diameter"
     ;
  input Modelica.SIunits.Density rho "Density";
  input Modelica.SIunits.DynamicViscosity mu "
     Dynamic viscosity of water";
  input Modelica.SIunits.Height eps "Pipe roughness
      height";
  input Modelica.SIunits.Length L "Length";
  input Modelica.SIunits.Acceleration g "
     Graviational accelleration";
  input Real v_guess "Guessed v using frictionless
     Bernoulli";
  input Real hf_guess "Change in h due to guessed
     friction";
  input Real dh "Real dh";
  output Real v_out "Flow velocity";
protected
  Real f_guess;
algorithm
  while hf_guess>=dh loop
    v_guess :=v_guess*0.95;
    f_guess :=100*OpenWPL.Functions.f_Haaland(
       v_guess,D,rho,mu,eps);
```

```
      hf_guess:=OpenWPL.Functions.H_f(f_guess,L,D,
          v_guess,g);
    end while;
    v_out := v_guess;
  end v_iterate;

end Functions;
```

# Appendix - H

## *Modelica code: pipe component*

```
model Pipe
    /*
              Final model of a pipe using MOC to
                 simulate the waterhammer effect and
                 mass oscillations. Based on version
                 13.

              Author:   Jonas Tveit Hinna
              Date:     10.06.2021
              Version:  13
            */
    extends Modelica.Fluid.Interfaces.PartialTwoPort;
    import Modelica.Constants;
    import Modelica.SIunits;
    outer Modelica.Fluid.System system "System wide
       properties";
    outer OpenWPL.MOC moc "System wide storage of MOC
        values";
    extends OpenWPL.Icons.Waterhammer_MOC;
    // Geometry
    final parameter Real nParallel(min = 1) = 1 "
       Number of identical parallel pipes" annotation
       (
      Dialog(group = "Geometry"));
    parameter SI.Length length = 1 "Length"
       annotation(
      Dialog(tab = "General", group = "Geometry"));
```

```
final parameter Boolean isCircular = true "= true
    if cross sectional area is circular"
   annotation(
  Evaluate,
  Dialog(tab = "General", group = "Geometry"));
parameter SI.Diameter diameter_in "Inlet diameter
    of circular pipe" annotation(
  Dialog(group = "Geometry", enable = isCircular)
    );
parameter SI.Diameter diameter_out = diameter_in
   "Outlet diameter of circular pipe" annotation(
  Dialog(group = "Geometry", enable = isCircular)
    );
final parameter SI.Area A_in = Modelica.Constants
   .pi * diameter_in * diameter_in / 4 "Inlet
   cross section area" annotation(
  Dialog(tab = "General", group = "Geometry",
    enable = not isCircular));
final parameter SI.Area A_out = Modelica.
   Constants.pi * diameter_out * diameter_out / 4
    "Oulet cross section area" annotation(
  Dialog(tab = "General", group = "Geometry",
    enable = not isCircular));
final parameter Modelica.Fluid.Types.Roughness
   roughness = 2.5e-5 "Average height of surface
   asperities" annotation(
  Dialog(group = "Geometry"));
final parameter SI.Volume V = A_in * length *
   nParallel "volume size";
// Boundary condition determination
parameter Boolean closedValve = true "= true if
   the right side of the pipe is closed by a
   valve" annotation(
  Dialog(enable = true, tab = "General", group =
    "Boundary condition"));
// Static head
final parameter SI.Length height_ab = 0 "Height(
   port_b) - Height(port_a)" annotation(
  Dialog(group = "Static head"));
// Initialization
final parameter Medium.AbsolutePressure p_a_start
```

```
    = system.p_start "Start value of pressure at
    port a" annotation(
  Dialog(tab = "Initialization"));
final parameter Medium.AbsolutePressure p_b_start
    = p_a_start "Start value of pressure at port
    b" annotation(
  Dialog(tab = "Initialization"));
final parameter Medium.MassFlowRate m_flow_start
    = system.m_flow_start "Start value for mass
    flow rate" annotation(
  Evaluate = true,
  Dialog(tab = "Initialization"));
// Fluid parameters
final parameter SIunits.Density rho = 999.2 "
    Fluid density" annotation(
  Dialog(group = "Fluid properties"));
final parameter SIunits.DynamicViscosity mu =
    0.0011684 "Dynamic viscosity" annotation(
  Dialog(group = "Fluid properties"));
final parameter SIunits.KinematicViscosity nu =
    mu / rho "Kinematic viscosity" annotation(
  Dialog(group = "Fluid properties"));
parameter SIunits.VelocityOfSound a = 1200 "
    Velocity of sound in water" annotation(
  Dialog(group = "Fluid properties"));
// Method of Characteristics declarations
parameter Boolean friction = true "= true if
    there is friction in the system" annotation(
  Dialog(enable = true, tab = "General"));
final parameter Real dx = length / moc.N "Length
    step calculated in this model";
final parameter Integer N = integer(length / dx)
    "n parts";
final parameter Real dt = dx / a "Time step";
//final parameter Real dx_min = moc.dx_min;
final parameter Real dx_min = dx;
final parameter Integer T = moc.T;
// Temporary time constant
// Local array storage declarations. Global
    values are stored in the component "MOC.mo"
Real v_array[T, N];
```

```
  //array for velocity
  Real fp_array[T, N];
  //array for friction at plus characteristic
  Real fm_array[T, N];
  //array for friction at minus characteristic
  Real Rp_array[T, N];
  //array for resistance at plus characteristic
  Real Rm_array[T, N];
  //array for resistance at minus characteristic
  Real Bp_array[T, N];
  //array for impedance at plus characteristic
  Real Bm_array[T, N];
  //array for impedance at minus characteristic
  Real Cp_array[T, N];
  //array for plus characteristic
  Real Cm_array[T, N];
  //array for minus characteristic
  Real H_array[T, N];
  //array for head
  Real Q_array[T, N];
  //array for flow
  Real A_vector[N];
  //vector for area
  Real D_vector[N];
  //vector for diameter
  // Initial condition declarations
  Real Qa = port_a.m_flow / rho;
  Real Qb = Qa;
  Real Ha = port_a.p / (rho * system.g);
  //Real Hb = Ha;
  Real v = sqrt(2 * system.g * Ha);
  parameter Boolean expansion = false "= true if
      expansion equations are enabled. If not then
      A_outlet = A_inlet.";
protected
equation
//moc.time_step[1] = dx; not imlemented yet
/*
  if diameter_out == diameter_in then
    expansion = true;
  elseif not diameter_out == diameter_in then
```

```
    expansion = false;
  end if;

  Get translational warning: "In relation
     diameter_out == diameter_in, == on Real
     numbers is only allowed inside functions"
  Get an unbalanced system with one more equation
     than variables
  */
// Area and diameter vectors
  A_vector[:] = linspace(A_in, A_out, N);
  D_vector[:] = linspace(diameter_in, diameter_out,
      N);
// Initial condition loop
  for k in 1:N loop
    v_array[1, k] = OpenWPL.Functions.v_Velocity(H
       = Ha, g = system.g);
    Q_array[1, k] = v_array[1, k] * A_vector[k];
// Initial conditions for the plus characteristic
    if friction then
      fp_array[1, k] = OpenWPL.Functions.f_Haaland(
         v = v_array[1, k], D = D_vector[k], rho =
         rho, mu = mu, eps = roughness);
    elseif not friction then
      fp_array[1, k] = 0.0;
    end if;
    H_array[1, k] = OpenWPL.Functions.H_initial(f =
       fp_array[1, k], H_res = Ha, Q = Q_array[1,
       k], L = dx * k, g = system.g, D = D_vector[k
       ], A = A_vector[k]);
    Rp_array[1, k] = OpenWPL.Functions.R_Resistance
       (f = fp_array[1, k], dx = dx_min, g = system
       .g, D = D_vector[k], A = A_vector[k]);
    Bp_array[1, k] = OpenWPL.Functions.B_Impedance(
       a = a, A = A_vector[k], g = system.g) +
       Rp_array[1, k] * abs(Q_array[1, k]);
    Cp_array[1, k] = OpenWPL.Functions.C_plus(Ha =
       H_array[1, k], B = OpenWPL.Functions.
       B_Impedance(a = a, A = A_vector[k], g =
       system.g), Qa = Q_array[1, k]);
// Initial conditions for the minus characteristic
```

```
    if friction then
      fm_array[1, k] = OpenWPL.Functions.f_Haaland(
          v = v_array[1, k], D = D_vector[k], rho =
          rho, mu = mu, eps = roughness);
    elseif not friction then
      fm_array[1, k] = 0.0;
    end if;
    Rm_array[1, k] = OpenWPL.Functions.R_Resistance
        (f = fm_array[1, k], dx = dx_min, g = system
        .g, D = D_vector[k], A = A_vector[k]);
    Bm_array[1, k] = OpenWPL.Functions.B_Impedance(
        a = a, A = A_vector[k], g = system.g) +
        Rm_array[1, k] * abs(Q_array[1, k]);
    Cm_array[1, k] = OpenWPL.Functions.C_minus(Hb =
         H_array[1, k], B = OpenWPL.Functions.
        B_Impedance(a = a, A = A_vector[k], g =
        system.g), Qb = Q_array[1, k]);
  end for;
// End I.C loop
// Time iteration loop
  for j in 2:T loop
// BC left: open to reservoir with p equal to port.
    p (should be dp/dt=0)
    H_array[j, 1] = Ha;
    v_array[j, 1] = OpenWPL.Functions.v_Velocity(H
        = H_array[j, 1], g = system.g);
    Q_array[j, 1] = (H_array[j, 1] - Cm_array[j -
        1, 2]) / Bm_array[j - 1, 2];
    if friction then
      fp_array[j, 1] = OpenWPL.Functions.f_Haaland(
          v = v_array[j, 1], D = D_vector[1], rho =
          rho, mu = mu, eps = roughness);
    elseif not friction then
      fp_array[j, 1] = 0.0;
    end if;
    Rp_array[j, 1] = OpenWPL.Functions.R_Resistance
        (f = fp_array[j, 1], dx = dx_min, g = system
        .g, D = D_vector[1], A = A_vector[1]);
    Bp_array[j, 1] = OpenWPL.Functions.B_Impedance(
        a = a, A = A_vector[1], g = system.g) +
        Rp_array[j, 1] * abs(Q_array[j, 1]);
```

```
    Cp_array[j, 1] = OpenWPL.Functions.C_plus(Ha =
        H_array[j, 1], B = OpenWPL.Functions.
        B_Impedance(a = a, A = A_vector[1], g =
        system.g), Qa = Q_array[j, 1]);
    if friction then
      fm_array[j, 1] = OpenWPL.Functions.f_Haaland(
          v = v_array[j, 1], D = D_vector[1], rho =
          rho, mu = mu, eps = roughness);
    elseif not friction then
      fm_array[j, 1] = 0.0;
    end if;
    Rm_array[j, 1] = OpenWPL.Functions.R_Resistance
        (f = fm_array[j, 1], dx = dx_min, g = system
        .g, D = D_vector[1], A = A_vector[1]);
    Bm_array[j, 1] = OpenWPL.Functions.B_Impedance(
        a = a, A = A_vector[1], g = system.g) +
        Rm_array[j, 1] * abs(Q_array[j, 1]);
    Cm_array[j, 1] = OpenWPL.Functions.C_minus(Hb =
         H_array[j, 1], B = OpenWPL.Functions.
        B_Impedance(a = a, A = A_vector[1], g =
        system.g), Qb = Q_array[j, 1]);
// BC right: closed by valve with Q=0
    if closedValve then
      Q_array[j, N] = 0.0;
      v_array[j, N] = OpenWPL.Functions.v_Velocity(
          H = H_array[j, N], g = system.g);
      H_array[j, N] = Cp_array[j - 1, N - 1] -
          Bp_array[j - 1, N - 1] * Q_array[j, N];
    elseif not closedValve then
      Q_array[j, N] = Q_array[j, 1];
      H_array[j, N] = Cp_array[j - 1, N - 1] -
          Bp_array[j - 1, N - 1] * Q_array[j, N];
      v_array[j, N] = OpenWPL.Functions.v_Velocity(
          H = H_array[j, N], g = system.g);
    end if;
    if friction then
      fp_array[j, N] = OpenWPL.Functions.f_Haaland(
          v = v_array[j, N], D = D_vector[N], rho =
          rho, mu = mu, eps = roughness);
    elseif not friction then
      fp_array[j, N] = 0.0;
```

```
    end if;
    Rp_array[j, N] = OpenWPL.Functions.R_Resistance
        (f = fp_array[j, N], dx = dx_min, g = system
        .g, D = D_vector[N], A = A_vector[N]);
    Bp_array[j, N] = OpenWPL.Functions.B_Impedance(
        a = a, A = A_vector[N], g = system.g) +
        Rp_array[j, N] * abs(Q_array[j, N]);
    Cp_array[j, N] = OpenWPL.Functions.C_plus(Ha =
        H_array[j, N], B = OpenWPL.Functions.
        B_Impedance(a = a, A = A_vector[N], g =
        system.g), Qa = Q_array[j, N]);
    if friction then
      fm_array[j, N] = OpenWPL.Functions.f_Haaland(
          v = v_array[j, N], D = D_vector[N], rho =
          rho, mu = mu, eps = roughness);
    elseif not friction then
      fm_array[j, N] = 0.0;
    end if;
    Rm_array[j, N] = OpenWPL.Functions.R_Resistance
        (f = fm_array[j, N], dx = dx_min, g = system
        .g, D = D_vector[N], A = A_vector[N]);
    Bm_array[j, N] = OpenWPL.Functions.B_Impedance(
        a = a, A = A_vector[N], g = system.g) +
        Rm_array[j, N] * abs(Q_array[j, N]);
    Cm_array[j, N] = OpenWPL.Functions.C_minus(Hb =
         H_array[j, N], B = OpenWPL.Functions.
        B_Impedance(a = a, A = A_vector[N], g =
        system.g), Qb = Q_array[j, N]);
// Space iteration loop
    for i in 2:N - 1 loop
      if expansion then
        Q_array[j, i] = scalar({OpenWPL.Functions.
            Q_expansion(g = system.g, Ha = H_array[j
             - 1, i - 1], Hb = H_array[j - 1, i +
            1], a = a, Aa = A_vector[i - 1], Ab =
            A_vector[i + 1], A = A_vector[i], Qa =
            Q_array[j - 1, i - 1], Qb = Q_array[j -
            1, i + 1])});
        H_array[j, i] = scalar({OpenWPL.Functions.
            H_expansion(g = system.g, Ha = H_array[j
             - 1, i - 1], a = a, Aa = A_vector[i -
```

```
            1], A = A_vector[i], Qa = Q_array[j - 1,
             i - 1], Q = Q_array[j, i])});
        elseif not expansion then
          H_array[j, i] = scalar({OpenWPL.Functions.
              H_head(Cp = Cp_array[j - 1, i - 1], Cm =
               Cm_array[j - 1, i + 1], Bm = Bm_array[j
               - 1, i + 1], Bp = Bp_array[j - 1, i -
              1])});
          Q_array[j, i] = scalar({OpenWPL.Functions.
              Q_flow(Cp = Cp_array[j - 1, i - 1], Cm =
               Cm_array[j - 1, i + 1], Bm = Bm_array[j
               - 1, i + 1], Bp = Bp_array[j - 1, i -
              1])});
        end if;
        v_array[j, i] = scalar(OpenWPL.Functions.
            v_Velocity(H = H_array[j, i], g = system.g
            ));
        if friction then
          fp_array[j, i] = OpenWPL.Functions.
              f_Haaland(v = v_array[j, i], D =
              D_vector[i], rho = rho, mu = mu, eps =
              roughness);
        elseif not friction then
          fp_array[j, i] = 0.0;
  // if testing for a square pressure pulse
        end if;
        Rp_array[j, i] = OpenWPL.Functions.
            R_Resistance(f = fp_array[j, i], dx =
            dx_min, g = system.g, D = D_vector[i], A =
             A_vector[i]);
        Bp_array[j, i] = OpenWPL.Functions.
            B_Impedance(a = a, A = A_vector[i], g =
            system.g) + Rp_array[j, i] * abs(Q_array[j
            , i]);
        Cp_array[j, i] = OpenWPL.Functions.C_plus(Ha
            = H_array[j, i], B = OpenWPL.Functions.
            B_Impedance(a = a, A = A_vector[i], g =
            system.g), Qa = Q_array[j, i]);
        if friction then
          fm_array[j, i] = OpenWPL.Functions.
              f_Haaland(v = v_array[j, i], D =
```

```
            D_vector[i], rho = rho, mu = mu, eps =
            roughness);
        elseif not friction then
          fm_array[j, i] = 0.0;
// if testing for a square pressure pulse
        end if;
        Rm_array[j, i] = OpenWPL.Functions.
            R_Resistance(f = fm_array[j, i], dx =
            dx_min, g = system.g, D = D_vector[i], A =
             A_vector[i]);
        Bm_array[j, i] = OpenWPL.Functions.
            B_Impedance(a = a, A = A_vector[i], g =
            system.g) + Rm_array[j, i] * abs(Q_array[j
            , i]);
        Cm_array[j, i] = OpenWPL.Functions.C_minus(Hb
             = H_array[j, i], B = OpenWPL.Functions.
            B_Impedance(a = a, A = A_vector[i], g =
            system.g), Qb = Q_array[j, i]);
    end for;
// End space loop
  end for;
// End time loop
// Port equations for mass flow rate
  port_a.m_flow = system.m_flow_start;
  0 = port_a.m_flow + port_b.m_flow;
//port_b.m_flow = Q_array[T,N]*rho;
//port_b.p = H_array[T,N]*rho*system.g;
/*
  // Port equations for trace substance mass flow.
     Not used, but implemented to be compatible
     with the Fluid library.
  port_a.C_outflow = inStream(port_b.C_outflow);
  port_b.C_outflow = inStream(port_a.C_outflow);
  */
// Port equations for enthalpy flow. Not used, but
   implemented to be compatible with the Fluid
   library.
  port_b.h_outflow = inStream(port_a.h_outflow);
//- system.g*height_ab;
  port_a.h_outflow = inStream(port_b.h_outflow);
//+ system.g*height_ab;
```

```
    annotation(
      defaultComponentName = "pipe",
      Documentation(info = "<html>
  <p>Model of a straight pipe modelled using the
     method of characteristics. This is the newest
     version of the component (same as v13).
  </p>
  </html>"),
      __OpenModelica_commandLineOptions = "--
         matchingAlgorithm=PFPlusExt --
         indexReductionMethod=dynamicStateSelection -
         d=initialization,NLSanalyticJacobian,newInst
         ");
end Pipe;
```

Jonas Tveit Hinna

# Modelica model of transient pipe flow in hydraulic laboratory systems using the method of characteristics

NTNU
Norwegian University of
Science and Technology

WATERPOWER LABORATORY
NTNU