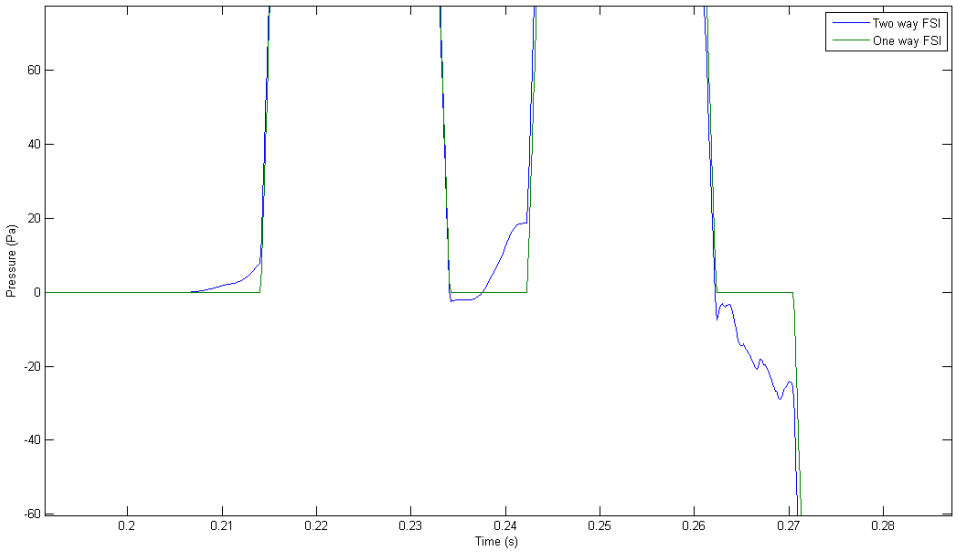
CHALMERS



Development of a two-way fluid-structure interaction model for pipe system analysis

A comparison for validating the one-way fluid-structure interaction method used at Ringhals AB

JANNA HEMPEL
OSKAR LINDGREN

Department of Applied Physics
Division of Nuclear Engineering
Chalmers University of Technology
Gothenburg, Sweden 2013
CTH-NT-271

Abstract

Pipe systems in nuclear power plants must be able to withstand forces caused by changes in the system conditions. To investigate how the pipe system reacts due to these changes mechanics and structural mechanics calculations are performed . At Ringhals AB, a two step simulation is performed where the output from the fluid mechanics calculations is used as an input for the structural mechanics calculations. This one-way fluid-structure interaction does not take into account that the structural movement can influence the flow and pressure of the fluid, which may yield higher forces that the system has to withstand.

This master thesis develops a method for modelling two-way fluid-structure interaction, by using the method of characteristics to transform the partial differential equations of mass and momentum into ordinary differential equations yielding the pressure and flow of the fluid. Matlab is used for calculating these properties, and the toolbox CalFEM is used when calculating the structural mechanics, the two solvers are coupled to obtain the two-way interaction.

The developed Matlab program show satisfactory results both when comparing the two solvers separately to software used at Ringhals AB, and when comparing with results obtained from Adina, a software for calculating two-way fluid-structure interaction. The Matlab program provides the user a familiar and easy way of building pipe systems and evaluating how the fluid properties changes and how the pipe system reacts on these changesss.

Acknowledgements

The authors would like to thank Emil Carlsson at ÅF-Industry AB and Anna Olsson at Ringhals AB for making this thesis possible. We would also like to thank our supervisors, Daniel Edebro and Nina Mong, for their valuable support and guidance throughout this project.

Further, we would like to thank the people working at the technology departments RTAT and RTPK at Ringhals AB for their assistance in validating our software and for their contribution to a pleasant and enjoyable working atmosphere.

Last, but not least, we would like to extend our gratitude to our examiner at Chalmers University of Technology, Prof. Anders Nordlund, for his help during the thesis.

Janna Hempel and Oskar Lindgren, Gothenburg June 2013

Contents

Notations

Abbreviations

FSI	Fluid-structure interaction
MOC	Method of Characteristics
RAB	Ringhals AB
RELAP	Reactor Excursion and Leak Analysis Program
NPP	Nuclear Power Plant
MWC	Meter water column

Letters

Q	Flow $[m^3/s]$
H	Piezometric head $[m]$
V	Velocity $[m/s]$
A	Area (cross sectional area) $[m^2]$
f	Darcy-Weisbach friction factor $[-]$
c	Wave propagation speed $[m/s]$
ν	Poisson ratio $[-]$
E	Young's modulus $[Pa]$
t	Time $[s]$
D	Diameter $[m]$
σ_z	Axial stress $[N/A]$
B	Characteristics impedance $[s/m^2]$
g	Gravitational acceleration $[m/s^2]$
x	Discretization length $[m]$
E	Young's modulus $[Pa]$
τ	Valve opening degree $[-]$
C_v	Flow coefficient $[-]$
R	Resistance coefficient $[s^2/m^3]$

1

Introduction

1.1 Background

In the nuclear power industry it is important to know how different transients influence the pipe system in the plant. The system will be exposed to changes in flow conditions due to pressure reduction, pipe ruptures, managing of valves and start and stop of pumps, among other things. These changes in flow yield forces that the pipe system has to withstand since the system cannot take too high stresses, which can arise due to the forces.

At Ringhals AB, RAB the fluid mechanics and structural mechanics calculations are made in two separate steps, using the software Relap for the fluid mechanics and Pipestress for the structure mechanics. For the fluid part, the flow and pressure in the pipe system is calculated, but also the forces acting on the system. In the structural mechanics calculations the stresses, caused by the forces, are calculated and used to predict how well the structure withstand the forces. This simplification, where only the flow influences the structural movements in the system and not vice versa, could yield computational errors due to the fact that structural movement can affect the behaviour of the flow. Today there is no way to investigate the errors, and it is uncertain to which extent it effects the results.

1.2 Purpose

The purpose of this master thesis is to develop a two-way fluid-structure interaction, two-way FSI, method for analysis of pipe systems. The method should be able to use to investigate the inaccuracies of the one-way interaction approach used at RAB today. It will be investigated how this affect the result, i.e. if the pressure and the flow will differ when using two-way FSI instead.

With software available at RAB, a program will be built to give an understanding

of how different transients affects the flow and pressure in pipe systems and how the systems response to these changes. The program will provide the user with an easy way to know if the results using one-way FSI is a valid approximation or not, and how reliable the results of the calculations are for different scenarios.

1.3 Limitations

Matlab will be used for coding purposes, and Excel for most of the input writing. The focus will be on how the system reacts to pressure changes when using valves and resevoirs; no start/stop of pumps or pipe rupture will be taken to account. Concerning the dimension of the pipes, all pipes in the system are assumed to have the same diameter and thickness. RELAP5 och Pipestress will be used in a small extent for fluid and structural mechanics calculations, respectively, for comparing the present solution method.

1.4 Research questions

Q1. Can a program be built to implement the feedback from structural movement?

Q2. Can the present method used at RAB be used, or should two-way FSI be used instead?

Q3. To what extent does the structural movement of the pipe system affect the pressure and the flow?

2

Theory

In a nuclear power plant, NPP, fluids are transported through many different piping systems. It can be circuits where drain from preheaters is transported to the condense system and . These fluid transporting systems contain pipes, pumps, reservoirs and different types of valves controlling the flow. It is important that the system can withstand different types of transients that can occur, both considering daily changes in the system like opening/closing of valves and pressure gradients, but also transients occurring more infrequently like earth quakes. Due to this, investigations are done to predict how the system reacts while being exposed to this kind of transients. The fluid transported through the system give rise to forces leading to physical effects like displacement of the pipes, yielding stresses in the system. Different parts of the system can handle different stresses; there is a maximum allowable stress ratio that must not be exceeded to ensure that the construction will hold. This stress ratio is established for each segment, and is a relation between the largest allowable stress and the actual stress.

2.1 The present method

At RAB a two step method is used for the analysis of the pipe system, where the FSI is seen as a one-way interaction. In the first step RELAP5, is used for calculating the forces acting on the piping system caused by the movement of the fluid. In the second step these forces are used as an input in the structural analysis software Pipestress, which verifies if the pipe system manages the requirements.

2.1.1 RELAP5

RELAP5 is used for the fluid mechanics calculation, this is an one-dimensional simulation software made for transients analysis in NPPs and for analysis of the reactor design. RELAP5 is used in the first step in the present method at RAB, where different

properties of the fluid are calculated [1]. After the simulation data such as the mass flow through components, opening/closing of valves and the pressure throughout the system can be obtained. By the use of existing variables, new once can be introduced: such as forces acting on the pipe system. The software only provides raw data, but the result can be plotted in the software AptPlot for analysis of the result.

2.1.2 Pipestress

In Pipestress, the system is analysed with respect to the structural quality in situations where the system is exposed to forces. Evaluation of the stresses in the system yields information of how well the piping system manages the different transients. The loads are divided into categories, depeding on the magnitude of the allowable stress.

There are three different load categories: sustained loads, occasional loads and expansion loads. The first two load types are loads arising due to forces and the third is loads caused by structural movement. Sustained loads occur on a regular basis, e.g. loads due to the dead weight of the system. The occasional loads occur during shorter periods like seismic loads, vibrations and pipe rupture. Expansion loads are loads due to displacements of the piping system and can be an effect of thermal expansion of the piping [2].

2.2 Fluid-structure interaction

When pressure waves propagate through the fluid, it also propagates through the pipe wall due to different coupling mechanisms. The coupling mechanisms are divided into Poisson coupling, friction coupling and junction coupling. When the pressure waves transmits energy through the walls, the diameter of the pipe will increase which causes strains. The high speed of the generated precursor waves lead to interaction between the flow and structure, resulting in pressure changes in the fluid and movements of the pipe system. The mentioned interaction is called *Poisson coupling* and is a distributed force, since it interacts along the entire pipe. Another distributed coupling mechanism is represented by the mutual interaction between fluid and pipe and referred to as the *friction coupling* [3]. In most cases, the dominant coupling mechanism between fluid and pipe is due to local forces, this is called *junction coupling* and refers to the axial loads leading to an increase of the dynamic pressure due to structural movement. This interaction often occurs at bends and cross-sections where the pipe area changes [4].

2.3 Finite Element Method

Many physical phenomena can be described by using differential equations, but most of the equations are too complex to be solved in an analytical manner. To understand the physical behaviour, a numerical approach can be applied giving approximative solutions to general differential equations that are valid over a certain region of the area of interest. [10]. Dividing the geometry into these small parts, called *finite elements*, simplifies the

solution process since a generic solution for the entire geometry does not need to be considered. The numerical approximation is done over each of the finite elements and assembled to yield a solution valid for the whole geometry. The number of elements the geometry is divided into depends on the complexity of the area and the level of accuracy of the simulation; the more elements the better solution. However, more elements leads to longer computational time.

2.4 Two-way FSI

There are some software solving two-way fluid-structure interaction problems. The solving process is an iterative procedure, with solvers for the fluid mechanics equations and for the structural mechanics equations. The different solvers gather data from each other, iterating until the transferred data have converged [6]. The system can e.g. involve an iterative process where the mesh is updated with loads from the other solver taking system deformation into account [7].

The main idea for the solution procedure, for this project, can be seen in the flowchart in Figure 2.1.

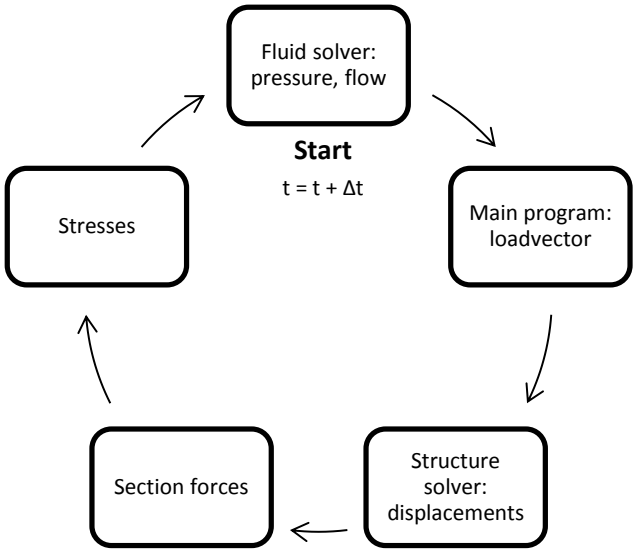


Figure 2.1: The solution procedure

The program will contain several subprograms, solving different parts of the problem. From the fluid solver the pressure and flow is computed, and used for generating a load vector in the main program. The loads causing displacements in the structure are computed using the load vector in the structure solver. The normal forces, caused by the displacements, are used calculating the stresses. The stresses are then included as an extra term in the modified fluid mechanics equation, to calculate the pressure and the flow.

3

Method

A Matlab program is developed for calculating the pressure head and the flow in the pipe system. The main program contains several sub programs; specifying the pipe geometry, calculating pressure and flow for the fluid, setting the different boundary conditions and calculating the structural response. More about how to use the program in section 5 *Users guide*.

3.1 Methods of Characteristics

The pressure head and the flow in the pipe system are calculated using the conservation of linear momentum, Equation 3.1, and conservation of mass (also known as the continuity equation) Equation 3.2 [3].

$$\frac{\partial Q}{\partial t} + gA\frac{\partial H}{\partial z} + SQ|Q| = 0 \tag{3.1}$$

$$S = f/(2DA)$$

$$\frac{\partial H}{\partial t} + \frac{c^2}{gA}\frac{\partial Q}{\partial z} - \frac{2c^2\nu}{gE}\frac{\partial \sigma_z}{\partial t} = 0 \tag{3.2}$$

The term $\frac{2\nu}{gE}\frac{\partial \sigma_z}{\partial t}$, where σ_z represents the axial stress in the pipe, is neglected if not taking axial stress or strain into account, an assumption that usually is made. The following derivations are done under this assumption, but will later be modified [8].
To solve the system numerically, the partial differential equations above are converted into the following ordinary differential equations:

$$C^+ : \quad \begin{cases} \frac{g}{c}\frac{dH}{dt} + \frac{1}{A}\frac{dQ}{dt} + \frac{1}{A}\frac{fQ|Q|}{2D} = 0 \\ \frac{dx}{dt} = +c \end{cases} \tag{3.3}$$

$$C^- : \quad \begin{cases} -\frac{g}{c}\frac{dH}{dt} + \frac{1}{A}\frac{dQ}{dt} + \frac{1}{A}\frac{fQ|Q|}{2D} = 0 \\ \frac{dx}{dt} = -c \end{cases} \tag{3.4}$$

C^+ and C^- are known as the two compatibility equations.

The system of equations can be solved using the finite difference method. A pipe of length L is divided into N number of elements, giving $N + 1$ number of nodes. For every time step Δt, the pressure and flow is computed in each node. The time step is determined by the pipe length and the wave speed according to: $\Delta t = \Delta x/c$. The C^+ equation is valid upstream i.e. when using information from the previous node in the previous time step, represented by the characteristic line with positive slope in Figure 3.1. The C^- equation is valid downstream i.e. when using information from the next node in the previous time step represented by the line with negative slope.

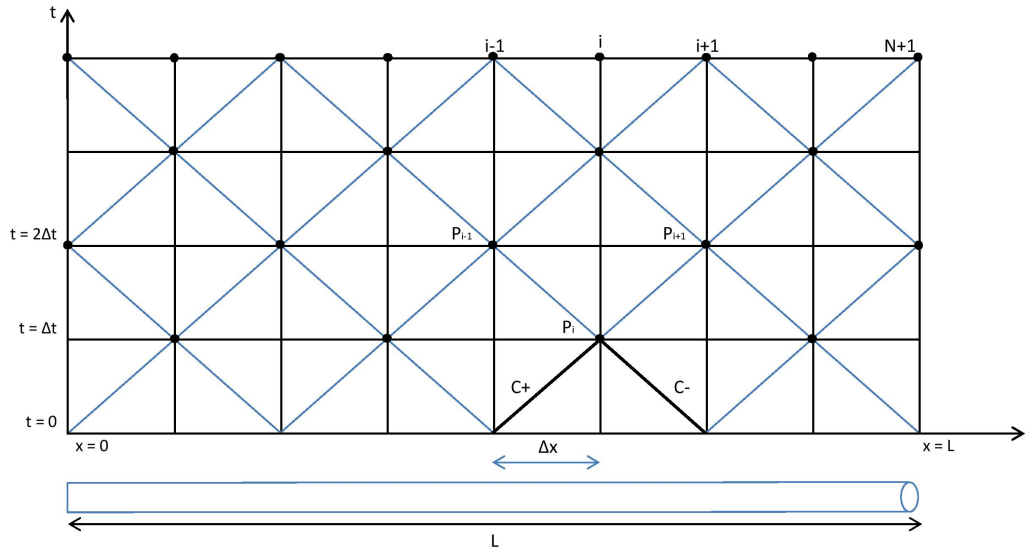


Figure 3.1: The x-t diagram for a pipe

The solution to the system of equations, for some point i, will then be:

$$C^+: \quad H_i = C_P - B_P \cdot Q_i \tag{3.5}$$

$$C^-: \quad H_i = C_M + B_M \cdot Q_i \tag{3.6}$$

$$C_P = H_{i-1} + BQ_{i-1} \tag{3.7}$$

$$B_P = B + R|Q_{i-1}| \tag{3.8}$$

$$C_M = H_{i+1} - BQ_{i+1} \tag{3.9}$$

$$B_M = B + R|Q_{i+1}| \tag{3.10}$$

Where:

$$B = \frac{c}{gA}$$

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

The first compatibility equation is valid along the characteristic line $\Delta x = c \cdot \Delta t$, obtaining C_P and B_P at the distance $i-1$ from the point of interest, i.e. at the previous time step $t - \Delta t$. The other one is valid along $\Delta x = -c \cdot \Delta t$, with C_M and B_M at the distance $i + 1$. The mentioned quantities can be calculated through Equation 3.7-3.10.

Using the expressions above in combination with Equation 3.5 and 3.6, the pressure head and flow can be calculated for each internal node according to Equation 3.11 and Equation 3.12, respectively [3].

$$H_i = \frac{C_P B_M + C_M B_P}{B_P + B_M} \tag{3.11}$$

$$Q_i = \frac{C_P - C_M}{B_P + B_M} \tag{3.12}$$

3.1.1 Boundary conditions

To obtain the pressure head and flow at the boundary nodes, there are different boundary condition that can be applied depending on which element the pipe is attached to. Which equation to use depends on how the pipe is connected to the element i.e. if the boundary condition is upstream or downstream end of the pipe. Having a downstream boundary condition the C^+ equation is used, and for upstream boundary condition the C^- equation is used [5].

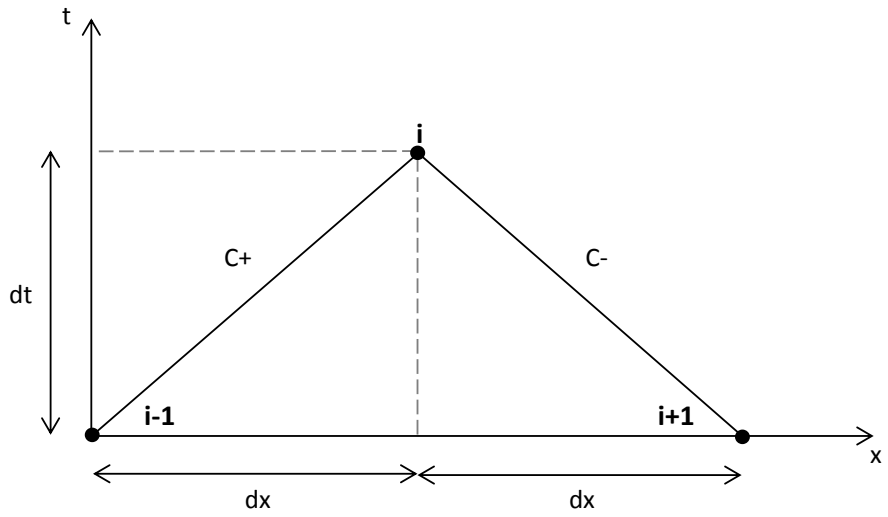


Figure 3.2: x-t diagram

Two pipes connected in a series

If two pipes are connected in a series as illustrated in Figure 3.3, the conservation of mass gives that the flow in the end of the first pipe, i.e. in the last node, and the flow in the beginning of the second pipe, i.e. the first node, will be the same.

$$Q_{pipe1,N+1} = Q_{pipe2,1}$$

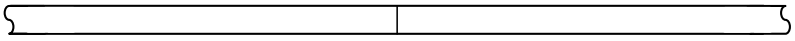


Figure 3.3: Pipes connected in a series

Similar, the conservation of energy gives

$$H_{pipe1,N+1} = H_{pipe2,1}$$

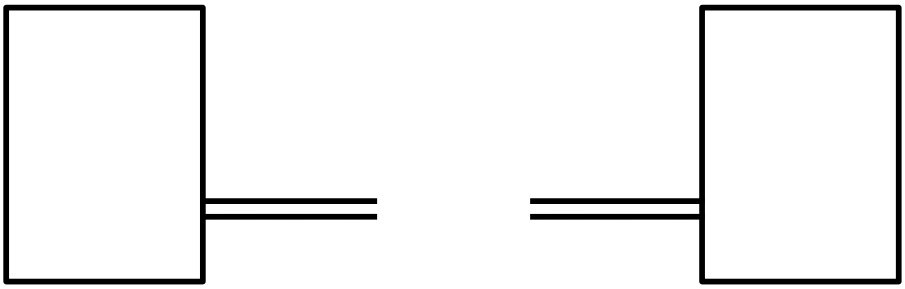
The unknown pressure head is given by Equation 3.13

$$H_{pipe1,N+1} = \frac{C_P/B_{pipe1} + C_M/B_{pipe2}}{1/B_{pipe1} + 1/B_{pipe2}} \tag{3.13}$$

and the flow is calculated by using Equation 3.5.

Reservoir

A reservoir can be modelled in two ways, either the reservoir is placed in the beginning of the system, Figure 3.4(a), or at the end of the system, Figure 3.4(b).



(a) Reservoir at the start of the system (b) Reservoir at the end of the system

Figure 3.4: Boundary condition: reservoir

The pressure head in the node in contact with the reservoir is assumed to be equal to the pressure head of the reservoir. Having a reservoir at the upstream end of the pipe this yields:

$$H_{pipe,1} = H_{res,up} \tag{3.14}$$

and with a reservoir at the downstream end, the pressure at the last node in the adjacent pipe is set to

$$H_{pipe,N+1} = H_{res,dwn} \tag{3.15}$$

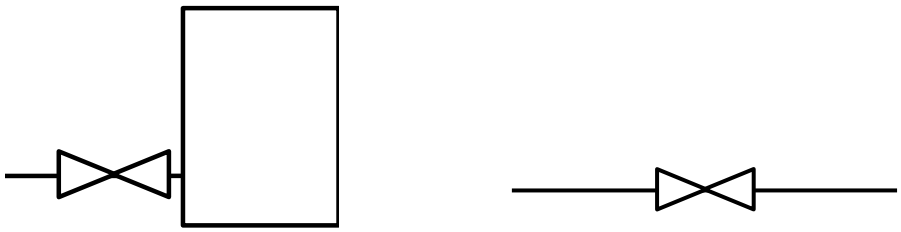
The flow can then be calculated by using Equation 3.6 for the first case, and Equation 3.5 for the latter. This yields the following equations:

$$Q_1 = \frac{H_1 - C_M}{B_M} \tag{3.16}$$

$$Q_{N+1} = \frac{C_P - H_{N+1}}{B_P} \tag{3.17}$$

Valve

Depending on which component the valve is attached to, different equations are used for computing the flow and the pressure head. In this case the valve is attached either to a reservoir, as illustrated in Figure 3.5(a), or as a coupling between two pipes as in Figure 3.5(b).



(a) Valve attached to a reservoir (b) Valve as a coupling between two pipes

Figure 3.5: Boundary condition: valve

Valve with resevoir at the downstream end

A pressure drop will occur over the valve, denoted as ΔH, which can be calculated by using information from the last node of the previous pipe $H_{pipe,N+1}$.

$$\Delta H = H_{pipe,N+1} - H_{reservoir} = \frac{KQ^2_{pipe,N+1}}{A^2 2g} \tag{3.18}$$

where K is the loss coefficient of the valve. The flow at the end of the previous pipe can be calculated using the degree of opening of the valve τ, and the steady state condition (subscripted with 0). The opening of the valve is adjusted depending on the required levels in the reservoirs.

$$Q_{pipe,N+1} = \frac{Q_0}{\sqrt{H_0}} \tau \sqrt{H_{pipe,N+1}} \tag{3.19}$$

This can be used in combination with the C^+ compatibility equation, for computing the flow at the downstream pipe end

$$Q_{pipe,N+1} = -B_P C_v + \sqrt{(B_P C_v)^2 + 2C_v C_P} \tag{3.20}$$

C_v is the flow coefficient for the valve and is a function of the valve opening grade

$$C_v = \frac{(Q_0 \tau)^2}{2H_0}$$

The pressure head in $H_{pipe,N+1}$ can then be determined with the C^+ equation.

Two pipes connected by a valve

The modelling of two pipes connected through a valve can be simplified as an abrupt change of area of the pipe. The pressure drop over the valve is calculated through the following expression:

$$\Delta H = H_{up} - H_{down} = \frac{1}{2gA_{valve}(t)} \cdot Q_i|Q_i| \tag{3.21}$$

Replacing H_{up} with Equation 3.5, and H_{down} with Equation 3.6, Q_i for the point of interest can be solved

$$(C_M + B_M \cdot Q_i) - (C_P - B_P \cdot Q_i) = \frac{1}{2gA_{valve}(t)} Q_i|Q_i|$$

$$\Leftrightarrow$$

$$Q_i = -\frac{(B_M + B_P)(2gA_{valve}(t))}{2} \pm \sqrt{\left(\frac{(B_M + B_P)(2gA_{valve}(t)}{2}\right)^2 - (C_P - C_M)} \tag{3.22}$$

The flow will be equal in and out through valve, giving

$$Q_i = Q_{up} = Q_{down}$$

The pressure head can then be calculated with Equation 3.5 and 3.6 for the previous pipe respective the next pipe.

Dead end

A dead end, illustrated in Figure 3.6 at the downstream end of the pipe gives no flow at the last node, i.e.

$$Q_{N+1} = 0$$

.

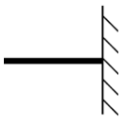


Figure 3.6: Boundary condition: dead end

The pressure head is calculated using Equation 3.5, or Equation 3.6 if the dead end is on the upstream side [5].

3.2 Modified MOC

By introducing an additional term in the compatibility equations, the influence of structural movement can be taken into account. Using Equation 3.2, having axial stress, the compatibility equations can be rewritten according to Equation 3.23 and 3.24.

$$C^+: \quad \frac{1}{A}\frac{dQ}{dt} + \frac{g}{c}\frac{dH}{dt} + f\frac{1}{A^2}\frac{Q|Q|}{2D} - gc\frac{2\nu}{gE}\frac{\partial \sigma_z}{\partial t} = 0 \tag{3.23}$$

$$C^- : \quad \frac{1}{A}\frac{dQ}{dt} - \frac{g}{c}\frac{dH}{dt} + f\frac{1}{A^2}\frac{Q|Q|}{2D} - gc\frac{2\nu}{gE}\frac{\partial \sigma_z}{\partial t} = 0 \tag{3.24}$$

The derivative of the stress can be approximated as shown below, for the C^+ equation [4].

$$\frac{\partial \sigma_z}{\partial t} = \sigma_{i-\frac{1}{2}}(t + \Delta t) - \sigma_{i-\frac{1}{2}}(t)$$

Using the approximation together with the finite difference method the compatibility equations above are rewritten as:

$$H_i = H_{i-1} - B(Q_i - Q_{i-1}) - RQ_i|Q_{i-1}| + c\Delta x\frac{2\nu}{gE}(\sigma_{i-\frac{1}{2}}(t) - \sigma_{i-\frac{1}{2}}(t - \Delta t)) \tag{3.25}$$

$$H_i = H_{i+1} - B(Q_i - Q_{i+1}) + RQ_i|Q_{i+1}| + c\Delta x\frac{2\nu}{gE}(\sigma_{i+\frac{1}{2}}(t) - \sigma_{i+\frac{1}{2}}(t - \Delta t)) \tag{3.26}$$

Instead of interpolating between the current step and the previous one when using $\sigma_{i-\frac{1}{2}}(t)$, further simplifications are done for the Matlab program, using only the previous node $\sigma_{i-1}(t)$. This approximation is also used in the C^- equation, but using the next node.

3.3 Structural modelling

For the structural modelling of the pipes, the finite element toolbox CalFEM in Matlab is used. The pipes are modelled as three-dimensional beams and the dynamic behaviour is computed using predefined functions. The CalFEM functions used are: beam3d, beam3s and step2.

3.3.1 Beam3d

To generate the mass and stiffness matrices, Ke and Me, of the pipe geometry the function beam3d is used. The function returns the element matrices for a dynamic analysis of three-dimensional beam elements according to the expression below.

$$[K,M] = beam3d(ex,ey,ez,ec,ep,eq)$$

where ex, ey and ez are element coordinate matrices in x, y and z direction. The variable ec is a vector giving the direction of the local z axis. The data vector ep contains the modulus of elasticity, the shear modulus, cross sectional area, moment of inertia with respect to y and z axes, and the St Venant torsional stiffness factor [11].

From the stiffness and mass matrix, the damping matrix can be computed according to $C = \mu M + \lambda K$ where μ is the mass proportional Rayleigh damping coefficient and λ is the stiffness proportional Rayleigh damping coefficient.

3.3.2 Beam3s

The beam3s function shown below is used to compute the section forces, es, and the element displacement, edi, at the evaluation points. The x-coordinates for these points are specified in eci. The input variables ex, ey, ez, ec, ep, and eq are described under the beam3d section.

$$[es,edi,eci] = beam3s(ex,ey,ez,ec,ep,ed,eq,n)$$

The variable ed contains the element displacement vector, and n specifies the number of evaluation points in which the section forces and displacements are computed [11].

3.3.3 Step2

To compute the dynamic solutions to a set of second order differential equations, the CalFEM function step2 is used. The function computes the solution to a second order differential equation of the form:

$$M\ddot{d} + C\dot{d} + Kd = f(x,y)$$

The function returns new displacements, velocities and acceleration according to the expression below. In $Dsnap$ snapshots of the displacements are stored for certain timesteps, that can be plotted to see how the structure move due to the forces.

$$[Dsnap,D,V,A] = step2(K,C,M,d0,v0,ip,f,bc)$$

The input arguments for the function are, besides the stiffness, damping and mass matrix, the initial conditions for the displacement and velocity, $d0$ and $v0$. The parameters governing the time integration are given in the variable ip, the matrix f containing the time dependent load vector and bc containing the boundary conditions [11].

4

Users guide

The input is divided into two sections; one for the fluid mechanics and one for the structural mechanics. In the fluid input sheet, boundary conditions for the system as well as fluid properties and pipe lengths are entered. In the structure input sheet, structure properties, lengths and angles are entered. To provide the user with a familiar and simple input module, the sheets are built in the same was as the existing Excel script used at RAB for the present RELAP5 calculations.

4.1 Building the system

4.1.1 Fluid input sheet

The fluid input sheet, Figure 4.1, recognizes a number of boundary conditions: *Pipe*, *Tank*, *Valve* and *Deadend*.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Input														
2	#Pipe	Unique ID	Name	Length[m]	dx[m]	Di[mm]	t[mm]	Vert.ang.[°]	Azi.ang.[°]	e[mm]	K[Pa]	ForceID	E[Pa]	v[-]	f[-]
3	#BC	input-filename	fromID	toID	valve:tau, tank:head										
4					tau[-], 0<tau<1										
5					head[m]										
6															
7	**Geometry**														
8															
9	Tank	-	-	PIPE_11	12										
10															
11	Pipe	PIPE_11	pipe	10.0000	0.2000	250.0000	15.0000	0	0	4.50E-05	0	0	2.1E+11	0.3	0.02
12	Pipe	PIPE_12	pipe	10.0000	0.2000	250.0000	15.0000	0	0	4.50E-05	0	0	2.1E+11	0.3	0.02
13															
14	Valve	vlv.dat	PIPE_12	PIPE_16	-										
15															
16	Pipe	PIPE_16	pipe	20.0000	0.2000	250.0000	15.0000	0	0	4.50E-05	0	0	2.1E+11	0.3	0.02
17	Pipe	PIPE_17	pipe	5.0000	0.2000	250.0000	15.0000	0	0	4.50E-05	0	0	2.1E+11	0.3	0.02
18	Pipe	PIPE_18	pipe	20.0000	0.2000	250.0000	15.0000	0	0	4.50E-05	0	0	2.1E+11	0.3	0.02
19															
20	Deadend	-	PIPE_18	-	-										
21															

Figure 4.1: Fluid input sheet

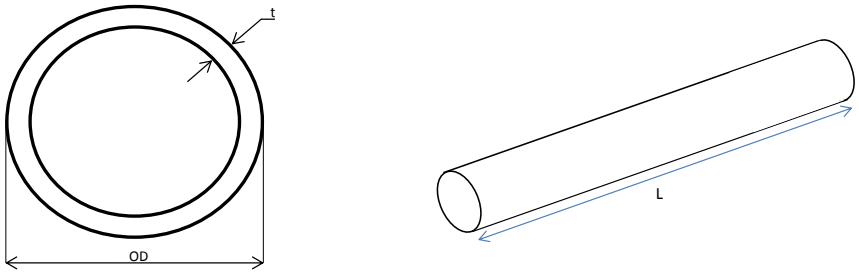
Pipe properties

To add a pipe to the system, the word *Pipe* is entered on the first column in the Excel sheet, see Figure 4.2. Note that pipes that are linked to each other has to be entered in the right order, without inserting any space between the rows (compare with when adding a valve after a pipe in Figure 4.1).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Input														
2	#Pipe	Unique ID	Name	Length[m]	dx[m]	Di[mm]	t[mm]	Vert.ang.[°]	Azi.ang.[°]	e[mm]	K[Pa]	ForceID	E[Pa]	v[-]	f[-]
11	Pipe	PIPE_11	pipe	10.0000	0.2000	250.0000	15.0000	0	0	4.50E-05	0	0	2.1E+11	0.3	0.02
12	Pipe	PIPE_12	pipe	10.0000	0.2000	250.0000	15.0000	0	0	4.50E-05	0	0	2.1E+11	0.3	0.02
16	Pipe	PIPE_16	pipe	20.0000	0.2000	250.0000	15.0000	0	0	4.50E-05	0	0	2.1E+11	0.3	0.02
17	Pipe	PIPE_17	pipe	5.0000	0.2000	250.0000	15.0000	0	0	4.50E-05	0	0	2.1E+11	0.3	0.02
18	Pipe	PIPE_18	pipe	20.0000	0.2000	250.0000	15.0000	0	0	4.50E-05	0	0	2.1E+11	0.3	0.02

Figure 4.2: Fluid input: pipe

Each pipe must have a unique pipe-ID, specified in column B. The ID is used for specifying to which pipe a specific boundary condition is connected to. A pipe name can be entered in column C for simplification purposes; for the user to keep track of where in the system the pipe is connected, but this is optional. The list below explains the rest of the properties that has to be set, where some of them are presented in Figure 4.3.



(a) Pipe: crossection (b) Pipe

Figure 4.3: Pipe: the geometry

- Column D: *Length* $[m]$ – Pipe length
- Column E: dx $[m]$ – Length of each pipe element that the pipe is divided into for the calculations
- Column F: Di $[m]$ – Inner diameter of the pipe
- Column G: t $[m]$ – Pipe thickness
- Column H: *Vert. ang.* $[°]$ – Vertical angle

- Column I: *Azi. ang.* [°] – Azimuthal angle
- Column J: e [mm] – Surface roughness
- Column K: K [Pa] – Bulk modulus
- Column L: *ForceID* – To facilitate the identification of a force acting on a specific pipe segment
- Column M: E [Pa] – Young's modulus
- Column N: v [-] – Poisson ratio
- Column O: f [-] – Friction factor

Boundary conditions

The boundary conditions are entered in a similar manner to the pipes; specifying which boundary condition that is being used in the first column, see Figure 4.4.

	A	B	C	D	E
1	Input				
3	#BC	input-filename	fromID	toID	valve:tau, tank:head
4					tau[-], 0<tau<1
5					head[m]
9	Tank	-	-	PIPE_11	12
14	Valve	vlv.dat	PIPE_12	PIPE_16	-
20	Deadend	-	PIPE_18	-	-

Figure 4.4: Fluid input: boundary conditions

The boundary conditions do not necessarily need to be modelled in a specific order, but the connection between the boundary conditions and the pipes has to be specified. This is done by writing the ID of the connecting pipes in the *fromID* column (if the pipe is connecting from the boundary condition) and in the *toID* column (if the pipe is connecting to the boundary condition).

Tank

If there is a reservoir in the beginning of the system, the word *Tank* has to be entered in the first column and the *toID* in column D to specify the position of the reservoir. If the reservoir is placed at the end of the system the *fromID* in column C is entered instead. There are two options for entering the pressure: in column B a file name of a dat-file can be entered if the pressure change is given over a certain time period, or a constant pressure can be entered in column E. If e.g. having a pulse where the pressure alternates from 0 Meter Water Column (MWC) to 600 MWC and back to 0 MWC again, in a time range of 10 milliseconds this is written as follows in the dat-file:

0	0
0.005	600
0.010	0

The first column represents at which time the different pressures occurs, and the other column represents the pressure head. These values are then interpolated in the program to get the distribution of the pressure head over the time interval.

Valve
The degree of valve opening can be entered in the same way as the pressure in the boundary condition *Tank*: either specified in a dat-file or as a constant degree of opening. For the first case the file name is entered in column B, and for the constant degree of opening, the value is entered in column E.

Dead end
To add a dead end, the word *Deadend* is entered in the first column, and the *fromID* in column C are entered.

4.1.2 Structure input sheet

The structure input sheet, Figure 4.5, is very similar to the fluid input sheet. The module consists of three input sections: one defining general material properties, one specifying the cross sections and one defining the geometry.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Input										Control		
2	#Pipe	Unique ID	Material	Cross-section	Length[m]	Vert.ang.[°]	Azi.ang.[°]				x	y	z
3	#Junction	fromID	toID	fromNode	toNode								
4	#Support	toID	toNode	k_axial	k_horiz	k_vert							
5	#AxiLoad	toID	toNode	filename	forceID	koeff	const						
6	#Anchor	toID	toNode										
7													
8	**Material**												
9		ID	E	G									
10	Matd	MAT_10	2.10E+11	8.00E+10									
11	Matd	MAT_11	2.10E+11	8.00E+10									
12	Matd	MAT_12	2.10E+11	8.00E+10									
13													
14													
15	**Cross-section**												
16		ID	OD (mm)	t (mm)	A	Iy	Iz	Kv	mass/length	j	Name		
17	Cros	CROS_17	280.000	15.00	1.249E-02	1.736E-04	1.736E-04	3.471E-04	1.436E+02	1.655E-03	4"sch10S		
18	Cros	CROS_18	114.300	6.02	2.048E-03	5.332E-06	5.332E-06	1.066E-05	2.369E+01	1.109E-04	4"sch40S		
19	Cros	CROS_19	114.300	8.56	2.844E-03	4.347E-06	4.347E-06	8.695E-06	2.935E+01	1.503E-04	4"sch80S		
20	Cros	CROS_20	114.300	5.00	1.717E-03	5.771E-06	5.771E-06	1.154E-05	2.134E+01	9.383E-05			
21													
22													
23	**Geometry**												
24											0.0000	0.0000	0.0000
25	Pipe	PIPE_25	MAT_10	CROS_17	10.0000	0.0000	0.0000				10.0000	0.0000	0.0000
26	Pipe	PIPE_26	MAT_10	CROS_17	10.0000	0.0000	0.0000				20.0000	0.0000	0.0000
27	Pipe	PIPE_27	MAT_10	CROS_17	20.0000	0.0000	0.0000				30.0000	0.0000	0.0000
28	Pipe	PIPE_28	MAT_10	CROS_17	5.0000	0.0000	0.0000				25.0000	0.0000	0.0000
29	Pipe	PIPE_29	MAT_10	CROS_17	20.0000	0.0000	0.0000				50.0000	0.0000	0.0000
30													
31													
32	Anchor	PIPE_25	1								0.0000	0.0000	0.0000
33	Anchor	PIPE_29	2								0.0000	0.0000	0.0000

Figure 4.5: Structure input sheet

Material

In the material section, Figure 4.6, an ID is given to each of the materials in column B. The ID is used when referring to the material utilized in the geometry section. In

column C and D respectively, the Young's module and the shear module are given.

	A	B	C	D	E
8	**Material**				
9		ID	E	G	
10	Matd	MAT_10	2.10E+11	8.00E+10	
11	Matd	MAT_11	2.10E+11	8.00E+10	
12	Matd	MAT_12	2.10E+11	8.00E+10	
13					

Figure 4.6: Structure input: material

Cross section

In the same manner to the material input, each cross section for the pipes has to be given an ID name in column B that is unique for the specific cross section, see Figure 4.7.

	A	B	C	D	E	F	G	H	I	J	K
15	**Cross-section**										
16		ID	OD (mm)	t (mm)	A	Iy	Iz	Kv	mass/length	j	Name
17	Cros	CROS_17	280.000	15.00	1.249E-02	1.736E-04	1.736E-04	3.471E-04	1.436E+02	1.655E-03	4"sch10S
18	Cros	CROS_18	114.300	6.02	2.048E-03	5.332E-06	5.332E-06	1.066E-05	2.369E+01	1.109E-04	4"sch40S
19	Cros	CROS_19	114.300	8.56	2.844E-03	4.347E-06	4.347E-06	8.695E-06	2.935E+01	1.503E-04	4"sch80S
20	Cros	CROS_20	114.300	5.00	1.717E-03	5.771E-06	5.771E-06	1.154E-05	2.134E+01	9.383E-05	
21											

Figure 4.7: Structure input: cross section

In addition to this there are some other information that has to be set in the input sheet, but the user only has to enter the first two parameters, and the last one:

- Column C: OD $[mm]$ – Outer diameter
- Column D: t $[mm]$ – Pipe wall thickness
- Column E: A $[m^2]$ – Inner pipe area
- Column F: Iy $[kgm^2]$ – Moment of inertia, y direction
- Column G: Iz $[kgm^2]$ – Moment of inertia, z direction
- Column H: Kv $[m^4]$ – Section factor of torsional stiffness
- Column I: *mass/length* $[kg/m]$ – Mass per unit length
- Column J: j $[m^3]$ – Polar moment of inertia per unit length
- Column K: *Name* – Name of the cross section

The information i column E-J is calculated automatically in excel by the following equations:

$$A = \frac{\pi(OD^2 - (OD - 2t)^2)}{4} \tag{4.1}$$

$$I_y = \pi \cdot \frac{OD^4 - (OD - 2t)^4}{64} \tag{4.2}$$

$$I_z = \pi \cdot \frac{OD^4 - (OD - 2t)^4}{64} \tag{4.3}$$

$$K_v = \pi \cdot \frac{OD^4 - (OD - 2t)^4}{32} \tag{4.4}$$

$$mass/length = A \cdot \rho_{pipe} + \frac{\rho_{water}\pi(OD - 2t)^2}{2^2} \tag{4.5}$$

$$j = 2\pi t \left(\frac{OD - t}{2}\right)^2 \tag{4.6}$$

Geometry

The geometry input, Figure 4.8, is similar to the pipe input in the fluid sheet. Concerning the pipes, the following information must be given:

- Column B: *Unique ID* – A unique pipe ID used when referring to the pipe
- Column C: *Material* – Specify which pre-defined material that is used
- Column D: *Cross section* – Specify which pre-defined cross section that is used
- Column E: *Length* [m] – Pipe length
- Column F: *Vert. ang.* [°] – Vertical angle
- Column G: *Azi. ang.* [°] – Azimuthal angle

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Input										Control		
2	#Pipe	Unique ID	Material	Cross-section	Length[m]	Vert.ang.[°]	Azi.ang.[°]				x	y	z
3	#Junction	fromID	toID	fromNode	toNode								
4	#Support	toID	toNode	k_axial	k_horiz	k_vert							
5	#AxiLoad	toID	toNode	filename	forceID	koeff	const						
6	#Anchor	toID	toNode										
23	**Geometry**												
24											0.0000	0.0000	0.0000
25	Pipe	PIPE_25	MAT_10	CROS_17	10.0000	0.0000	0.0000				10.0000	0.0000	0.0000
26	Pipe	PIPE_26	MAT_10	CROS_17	10.0000	0.0000	0.0000				20.0000	0.0000	0.0000
27	Pipe	PIPE_27	MAT_10	CROS_17	20.0000	0.0000	0.0000				30.0000	0.0000	0.0000
28	Pipe	PIPE_28	MAT_10	CROS_17	5.0000	0.0000	0.0000				25.0000	0.0000	0.0000
29	Pipe	PIPE_29	MAT_10	CROS_17	20.0000	0.0000	0.0000				50.0000	0.0000	0.0000
30													
31													
32	Anchor	PIPE_25	1								0.0000	0.0000	0.0000
33	Anchor	PIPE_29	2								0.0000	0.0000	0.0000

Figure 4.8: Structure input: geometry

When coupling pipes with each other no junction has to be added in the sheet, but to set junctions for other purposes, add the word *junction* in the first column and then set the following information:

- Column B: *fromID* – Unique ID of the element that is located before the junction
- Column C: *toID* – Unique ID of the element that is located after the junction
- Column D: *fromNode* – Is set to 1 if the coupling is set at the beginning of the coupled element, or 2 if the coupling is set at the end of the coupled element
- Column E: *toNode* – Same as described above

4.1.3 Anchor points

To introduce anchor points, enter the unique ID of the pipe connected to the anchor point and specify how it is connected to it: 1 for the beginning of the pipe and 2 for the end of the pipe.

4.1.4 Load case input

The load case specifies where the load is applied, at which pipe and in which nodes. The input is written as a matrix in Matlab and saved as a mat-file. The matrix defines where a pipe starts and where it ends, which is important since a pipe can be divided into several sections in the input sheet. The node column can be assigned the values 1 or 2, where 1 is the first node of the pipe and 2 is the last node of the pipe. This data is written as an matrix in the following format:

$$\begin{bmatrix} \text{Fluid pipe} & \text{Fluid node} & \text{Structure pipe} & \text{Structure node} \\ \text{Fluid pipe} & \text{Fluid node} & \text{Structure pipe} & \text{Structure node} \end{bmatrix}$$

The first row represents the beginning of the pipe, and the second row the end of the pipe.
For example, if having two pipes that is connected, where the first pipe has been divided into three pipes (in both the fluid sheet and the structure sheet) the following matrix is entered in the mat-file:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 2 & 3 & 2 \\ 4 & 1 & 4 & 1 \\ 4 & 2 & 4 & 2 \end{bmatrix}$$

4.2 Saving the result

In Matlab a result matrix is generated where information can be stored for each time step, giving how properties like flow, pressure and displacements alternates over time. The information can then be plotted to illustrate how the properties varies. The pipe

geometry is visualized through a plot, where the pipes are numbered and the degrees of freedom are shown. The content of the result matrix can be changed to get the time dependence of the wanted variable. This must be done in the Matlab code under section "Plot of the result", that can be seen in appendix B.1 Main program.

4.3 Running the program

To run the two-way FSI solver, the main program *Master* is used. *Master* will ask for the fluid input sheet and for the structure input sheet to get the required information to run the program. The computational process is an iterative process, as the one described in Figure 2.1.

5

Results and discussion

5.1 Model accuracy

To determine the validity of the Matlab program, comparisons were made with existing software: the fluid mechanics solver was compared to RELAP5, and the structure mechanics solver was compared to Pipestress. The simulation made using the two-way FSI model was compared to a simulation made by Onsala Ingenjörsbyrå, in the software Adina.

5.1.1 Validation of the fluid solver

To validate the fluid solver, a system was simulated in both the Matlab program, modified to only use the fluid mechanics solver, and in RELAP5. The system, represented in Figure 5.1, consists of two reservoirs connected by 15 mm thick steel pipes of 255 mm inner diameter, and a valve placed in the middle. The ends of the pipes, that are connected to each reservoir, are set as anchor points and are completely fixed.

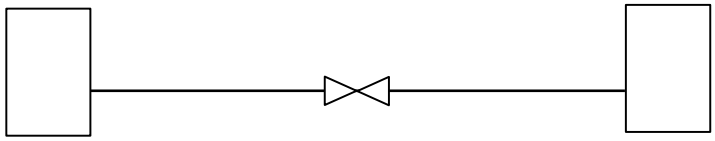


Figure 5.1: RELAP simulation: the system

In the simulation, the pressure in the reservoir to the left is 70 bar and the pressure in the right one is 65.5 bar. The valve closes during 33 millisecond after one second of simulation, yielding pressure changes along the pipes.

A pressure wave develops, traveling back and forth through the system between the reservoirs. How the pressure alternates in the node right before the valve can be seen in Figure 5.2 for the RELAP5 simulation, and in Figure 5.3 for the Matlab simulation.

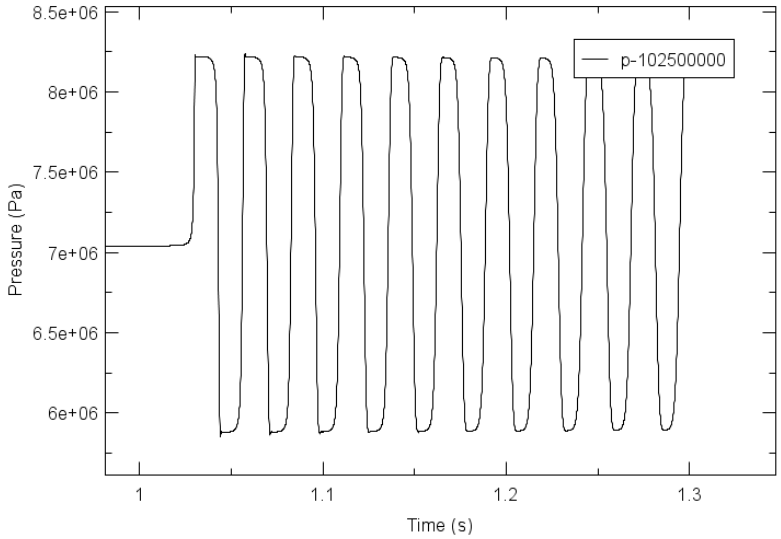


Figure 5.2: RELAP simulation: Pressure as a function of time in the first pipe

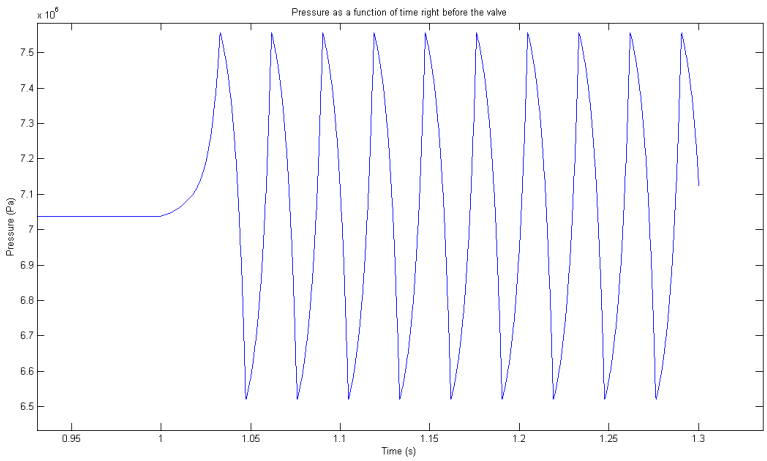


Figure 5.3: Matlab simulation: Pressure as a function of time in the first pipe

Comparing the two graphs it shows that the two simulations gives similar result. The maximum pressure is a bit higher in the first graph, and do not alter as fast as the other one, rather stay quite constant for a short period of time at the highest and lowest points in the graph. The slightly different shape and the difference in maximum and minimum values is is probably a consequence of the approximations made when deriving the equations that are used.

Conclusions

Due to the similar behaviour and magnitude of the pressure in the two simulations, the conclusion is made that the fluid solver is accurate enough to be used in the two-way FSI model.

5.1.2 Validation of the structural solver

The validity of the structural mechanics calculations in the Matlab solver was evaluated by comparing the results from the structure mechanics solver, i.e. when using CalFEM, with the results from Pipestress. The geometry of the case studied can be seen in Figure 5.4 below. The nodes where the pipes are connected to the wall were set as anchor points.

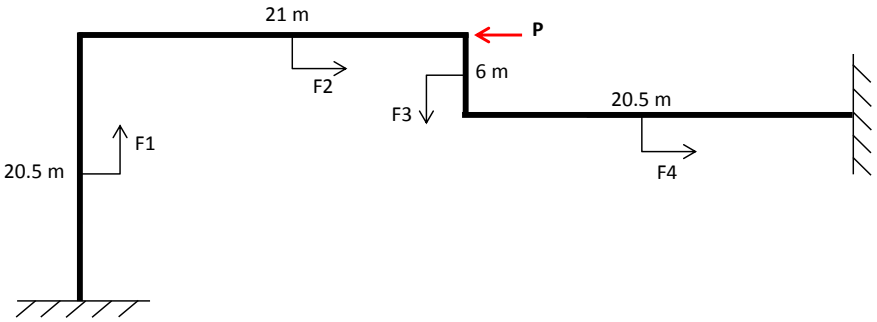
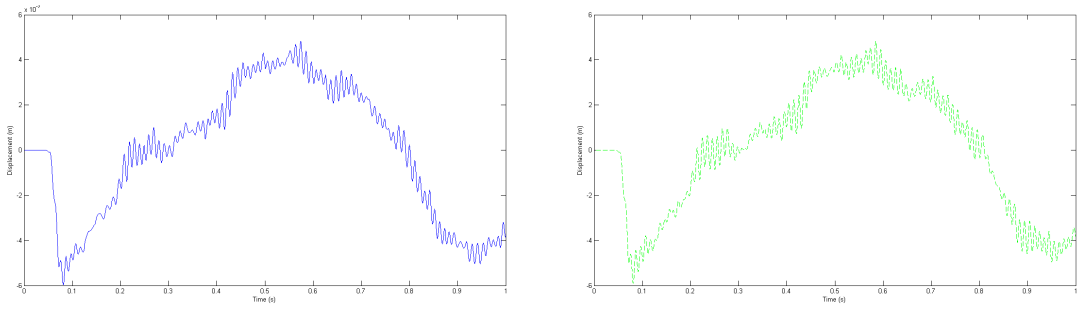


Figure 5.4: Geometry for validating the structure solver

In the end of each pipe segment forces, were applied, marked with *F1-F4* in the figure above. The forces, that varies over time, had the same magnitude in both simulations and were applied at the same locations. The displacements as a function of time at point P in the figure above are visualized in Figure 5.5 for the Matlab and Pipestress simulations.



(a) Displacements when using Matlab

(b) Displacements when using Pipestress

Figure 5.5: Displacements as a function of time, using only the structural mechanics solvers

The displacements varies in a very similar manner in the two cases, in general the look of the two graphs are almost the same where the fluctuations only differs a bit in some places.

Conclusions

The previous figures shows that the behaviour of the displacements are quite much the same for the two simulations, which are verified in the combined plot in Figure 5.6. It is hard to distinguish the two results from each other, indicating that the structural mechanics solver in the Matlab program yields satisfactory results.

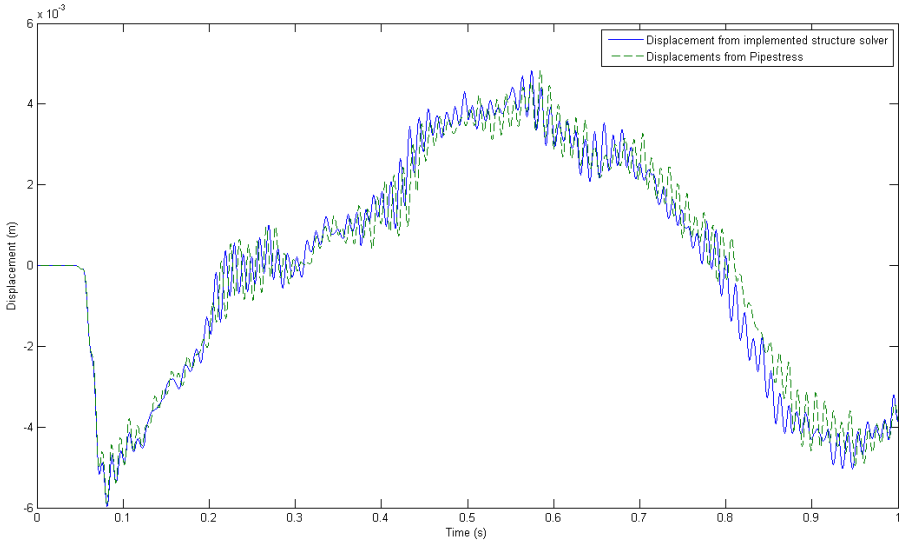


Figure 5.6: Comparison between the displacement obtained from Matlab and Pipestress

5.1.3 Validation of the two-way FSI model

To validate the accuracy of the two-way FSI model, combining the fluid and the structural mechanics solvers, a simulation done by Onsala Ingenjörsbyrå was studied. In the simulation two-way FSI has been used to analyse the flow and pressure in the pipe system shown in Figure 5.7, by using the software Adina. The construction is a simple pipe system without any reservoirs or valves, that are divided into four sections with lengths according to the figure below. The beginning of the first vertical pipe and the end of the last pipe are anchor points. The end of last pipe is modelled as a dead end, i.e. the flow is zero.

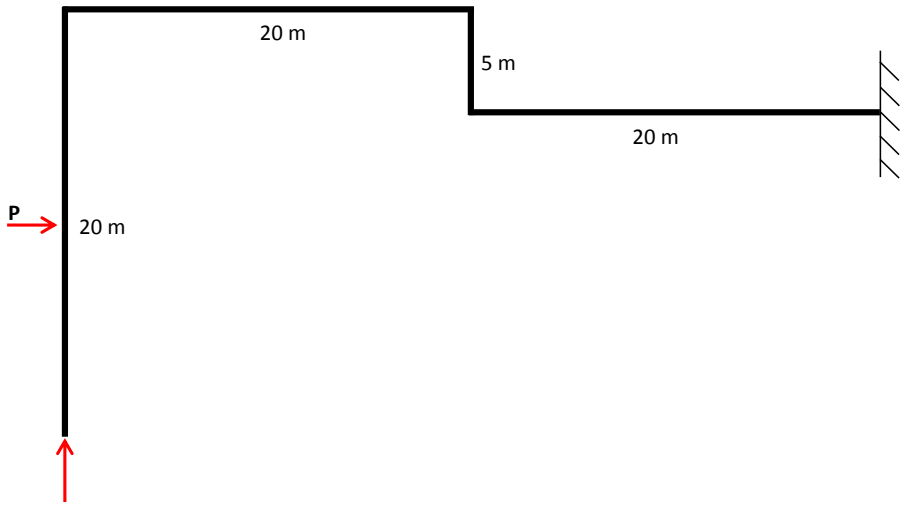


Figure 5.7: Onsala simulation: the system

A triangular pressure pulse, starting in the pipe where the vertical arrow points, propagates through the system exposing it to 60 bar over a time of 10 milliseconds. Figure 5.8 shows how the pressure alternates during a time period of 0.12 seconds at point P, in the middle of the first pipe section.

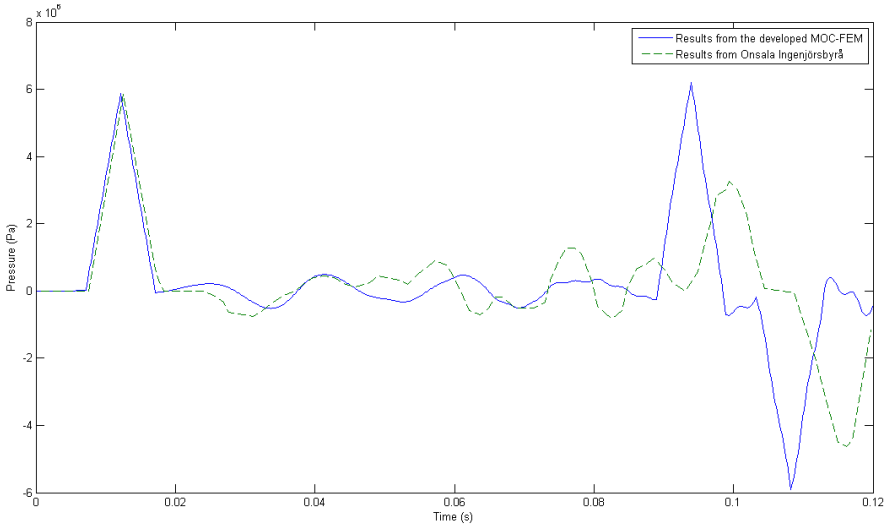
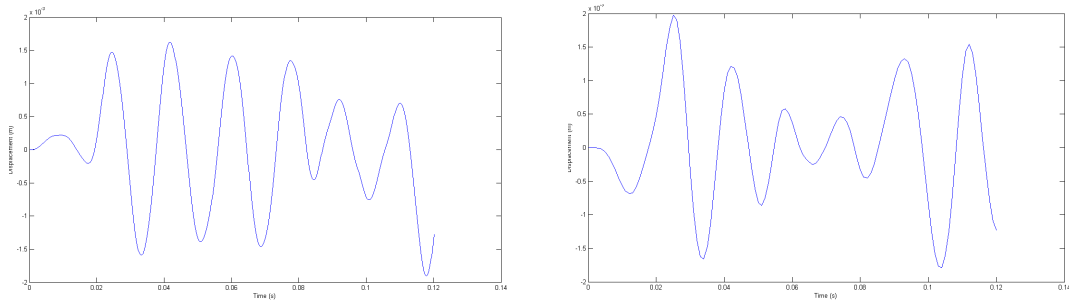


Figure 5.8: Pressure in the first pipe: comparison between Adina and Matlab

The continuous line shows how the pressure in the Matlab simulation alternates, and the other one how it alternates in the Adina simulation.

The pressure between the two big peaks varies in a similare manner, but concerning the magnitude of the second peak the pressure differs very much. The Matlab simulations shows a reduction of about 2 bar after one millisecond, while the other shows a reduction of more than 20 bar.

Figure 5.9 shows the displacements in millimetres over a time period of 0.12 seconds in the first pipe obtained through the Matlab program in Figure 5.9(a), and for the Adina simulation in Figure 5.9(b).



(a) Displacements when using Matlab

(b) Displacements when using Adina

Figure 5.9: Displacements as a function of time: comparison between Matlab and Adina

There are some variations in the displacement, having almost the same magnitude for the maximum value and behaves quite much in the same way, but the amplitude varies more over time using Adina.

Conclusion

The Matlab program gives decent result when comparing to the Adina simulation, where the pressure follows the same pattern and the displacements alters in a similar way but with different amplitudes. The difference in displacements may depend on the difference in pressure, which yields different loads and thereby the displacements will vary.

5.2 One-way versus two-way FSI

In this section, the pressure exerted on the pipe walls when excluding the influence of structural movement are referred to as one-way FSI. The forces generated when including the influence of structural movement are referred to as two-way FSI.

This system, represented in Figure 5.10, has a reservoir in one end and a dead end in the other. The system is coupled using five pipes of 240 mm inner diameter, where the beginning of the first pipe and the end of the last pipe are anchor points. The transient is initiated by altering the pressure in the tank, creating a pressure pulse of 800 MWC over 20 milliseconds, that propagates through the system.

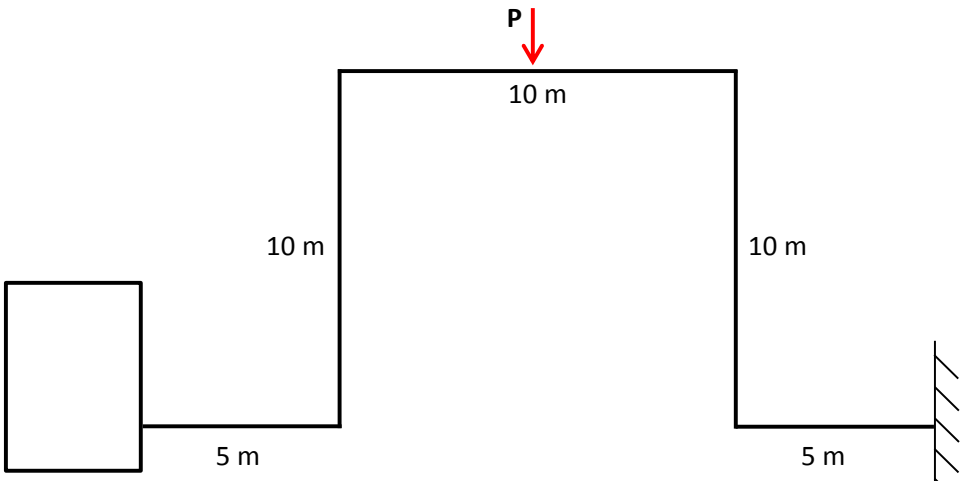


Figure 5.10: Two tank system

Figure 5.11 shows the pressure variation as a function of time at point P, both for the one-way FSI in and for the two-way FSI.

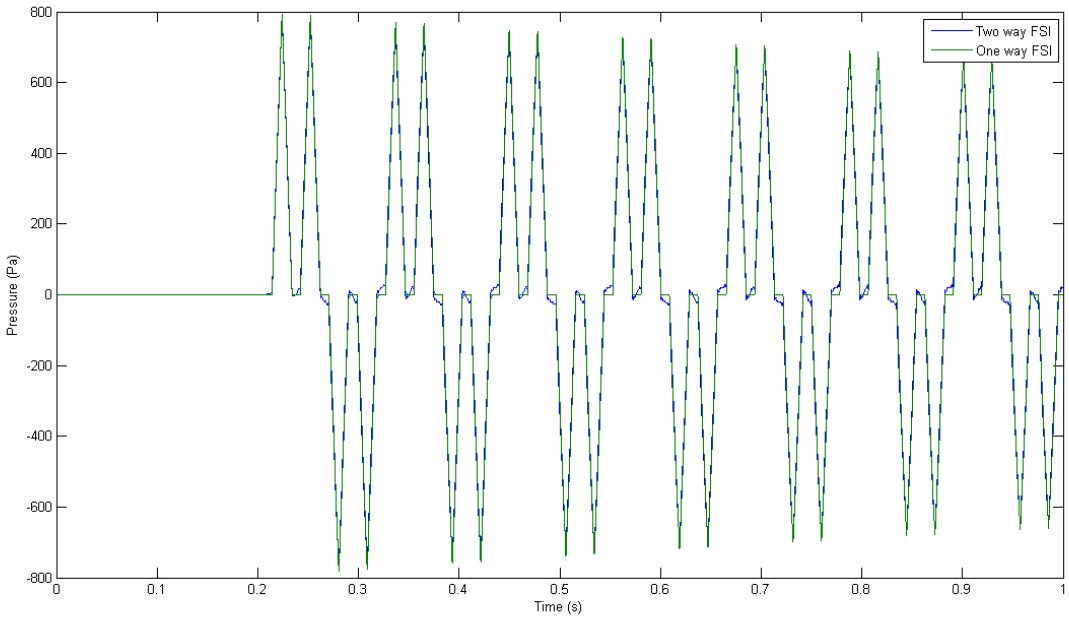
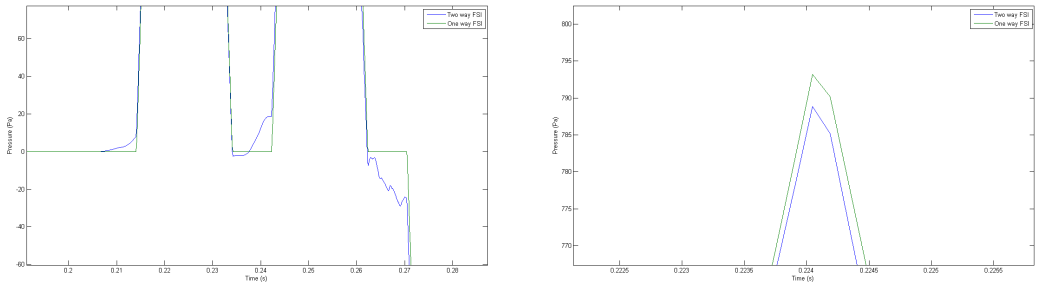


Figure 5.11: Two tank system

In Figure 5.11 the pressure is plotted for both the one-way FSI simulation and the two-way FSI simulation. The difference between the results are very hard to see, but with a closer look at the peaks and inbetween the peaks some difference can be seen, shown in Figure 5.12.



(a) Pressure at P, zoom between the peaks

(b) Pressure at P, zoom at the first peak

Figure 5.12: Pressure: comparing one-way and two-way FSI

The pressure variation is similar in both cases, where almost no difference in magnitude is seen when comparing the peaks in Figure 5.12(b). Though, in between the peaks there are some variation in the behaviour where fluctuations can be seen while using two-way FSI, see Figure 5.12(a). When using one-way FSI, the pressure immediately after the pressure wave has passed returns to the initial value.

Conclusion

The pressure, and thus the time-resolved forces acting on the structure, is overall very similar in both one-way and two-way FSI. Since the only noticeable difference is when evaluating the behaviour in between the pressure peaks, and due to the fact that the maximum pressure is almost the same for the two methods, the conclusion is drawn that two-way interaction does not need to be considered for this case.

6

Conclusion

The main focus during the master thesis were the following research questions:

Q1. Can a program be built to implement the feedback from structural movement?

Q2. Can the present method used at RAB be used, or should two-way FSI be used instead?

Q3. To what extent does the structural movement of the pipe system affect the pressure and the flow?

An overall conclusion is made from this, concerning the accuracy of the developed program, the accuracy of the mathematical model and how the results differs then using two-way FSI instead of one-way FSI.

6.1 The developed program

From the three validation cases, when the developed Matlab program is compared with other software, the conclusion is made that the program predicts the fluid behaviour and the structural response in a decent manner.

The pressure and displacement comparisons made with the Adina simulation shows satisfactory results. The higher pressure obtained with the developed program can be seen as conservative since it would generate higher forces acting on the structure. Concerning the displacements, the order of magnitude is correct compared to the Adina result, which is of high importance when determine the structural response to the load.

The results from the comparison case in chapter 5.2 indicates that there is a minor difference using one-way FSI instead of two-way FSI. However, using two-way FSI did not give a higher maximum pressure, and the forces acting on the pipe system remained almost the same. Due to this, the conclusion is drawn that the present method used at RAB gives satisfying results and a two-way interaction does not need to be considered.

Though, it should be noted that more complex systems have not been tested in the developed program, where two-way FSI might be of more importance e.g. when involving valves in the system.

6.2 Approximations

Some approximations are done when deriving the compatibility equations when using the method of characteristics, escpecially concerning the modified version. How large errors that occur when using the first order approximation, for deriving the stress term, is unknown and should be investigated further. For simplifying the programming, the stress term was further simplified using the stress from the previous (or the next) node instead of using half-steps as in Equation 3.25-3.26. An interpolation between the time steps and nodes, to yield the stresses at half time step and and half distance, should be done to ensure better results.

6.3 Further development

The program should be further developed, to be able to properly implement the valve function in the two-way FSI simulations, enabling the possibility of building more complex systems. In order to make this possible, the load vector that specifies where the loads are applied, should take boundary conditions like valves into account. In addition, the applied load should be based on the valve degree of opening. Also the approximations, mentioned in section 6.2, should be further developed providing a better coupling between the structural mechanics solver and the fluid mechanics solver.

Bibliography

[1] Paul Sherrer Institut. (2001). *Laboratory for thermal-hydraulics, Thermal-hydraulic Research for Reactor Safety, RELAP5.* http://lth.web.psi.ch/codes/RELAP5.htm

[2] Mårtensson, U. *Pipestress.* [Powerpoint-presentation] FS Dynamics 2006 (Acc 2013-01-31).

[3] A.R.D Thorley. *Fluid Transients in Pipeline Systems.* Förlag 2004.

[4] Lavooij, C.S.Q, Tijsseling A.S. Fluid-structure interaction in liquid filled piping systems. *Journal of Fluids and Structures Volume 5, Issue 5.*, p 573-595, 1991.

[5] Streeter, V.L., Lisheng, S., Wylie, E.B. *Fluid Transients in Systems.* Förelag 1993.

[6] Bi-directional (two-way) FSI. Sharcnet, 2010, https://www.sharcnet.ca/Software/Fluent13/help/cfx_ref/i1319161.html

[7] Fluid strucure interaction. CFD Online, 2012, http://www.cfd-online.com/Wiki/Fluid-structure_interaction#2-way_FSI (Acc 2013-03-21)

[8] Bergant A., Tijsseling A. S., Vítkovský, J.P., Covas, D. I. C., Simpson, A. R., Lambert, M. F. Parameters affecting water-hammer wave attenuation, shape and timing - Part 1: Mathematical tools. *Journal of Hydraulic Research Vol. 46. No 3*, p. 373-381, 2008. doi: 10.3826/jhr.2008.2848

[9] Melchers, R.E., Hough, R. *Modeling complex engineering structures.* American Society of Civil Engineers 2007.

[10] Ottosen, N., Petersson, H. *Introducton to the finite element method.* Prentice Hall Europe, 1992.

[11] Department of Mechanics and Materials, Lund University. *CalFEM. A finite element toolbox to MATLAB.* ISSN 0281-6679

A

Derving the compability equations

Momentum equation

$$\frac{1}{A}\frac{\partial Q}{\partial t} + g\frac{\partial H}{\partial x} + \frac{f}{2D}Q|Q| = 0 \tag{A.1}$$

Continuity equation

$$\frac{\partial H}{\partial t} + \frac{c^2}{gA}\frac{\partial Q}{\partial x} - \frac{2c^2\nu}{gE}\frac{\partial \sigma_x}{\partial t} = 0 \tag{A.2}$$

Combining the equations, using Lagrange multiplier λ at Equation A.3, and gathering related terms with each other:

$$\lambda\left(\frac{1}{A}\frac{\partial Q}{\partial t} + gA\frac{\partial H}{\partial x} + \frac{f}{2DA^2}Q|Q|\right) + \left(\frac{1}{c^2}\frac{\partial H}{\partial t} + \frac{1}{gA}\frac{\partial Q}{\partial x} - \frac{2c^2\nu}{gE}\frac{\partial \sigma_x}{\partial t}\right) = 0$$

$$\left(\lambda\frac{\partial Q}{\partial t} + \frac{c^2}{g}\frac{\partial Q}{\partial x}\right) + \left(\lambda g\frac{\partial H}{\partial x} + \frac{\partial H}{\partial t}\right) + \frac{f}{2D}Q|Q| - \frac{2c^2\nu}{gE}\frac{\partial \sigma_x}{\partial t} = 0 \tag{A.3}$$

Identification of λ using derivative rules:

$$\lambda\frac{dV(x,t)}{dt} = \lambda\frac{\partial V}{\partial t} + \lambda\frac{\partial V}{\partial x}\frac{dx}{dt} \tag{A.4}$$

$$\lambda\frac{dx}{dt} = \frac{c^2}{g} \Leftrightarrow \frac{dx}{dt} = \frac{1}{\lambda}\frac{c^2}{g}$$

$$\frac{dp(x,t)}{dt} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x}\frac{dx}{dt} \tag{A.5}$$

$$\frac{dx}{dt} = \lambda g$$

$$\Rightarrow \lambda g = \frac{1}{\lambda}\frac{c^2}{g} \Rightarrow \lambda = \pm\frac{c}{g} \tag{A.6}$$

The stress term kan be integrated over respective characteristic line, and approximated as follows:

$$\int_{i-1}^{i} \frac{\partial \sigma_x}{\partial t} dt \approx \left\langle \frac{\partial \sigma_x}{\partial t} \right\rangle_{i-\frac{1}{2}}^{n+\frac{1}{2}} \Delta t \approx \frac{\sigma_{i-1}(t) - \sigma_{i-1}(t-\Delta t)}{\Delta t}\Delta t = \sigma_{i-\frac{1}{2}}(t) - \sigma_{i-\frac{1}{2}}(t-\Delta t) \tag{A.7}$$

Combining Equation A.4 and A.5 with Equation A.3 gives

$$\lambda\frac{1}{A}\frac{dQ}{dt} + \frac{dH}{dt} + \lambda f\frac{1}{A^2}\frac{Q|Q|}{2D} - \frac{2\nu}{gE}\frac{\partial \sigma_x}{\partial t}c^2 = 0$$

$$\frac{1}{A}\frac{dQ}{dt} + \frac{1}{\lambda}\frac{dH}{dt} + f\frac{1}{A^2}\frac{Q|Q|}{2D} - \frac{1}{\lambda}\frac{2\nu}{gE}\frac{\partial \sigma_x}{\partial t}c^2 = 0$$

The two values of λ, computed in Equation A.6, gives two different values to the expression above. The positive root, $\lambda = \frac{c}{g}$, gives the expression for the C^+ compability equation.

$$C^+: \quad \frac{1}{A}\frac{dQ}{dt} + \frac{g}{c}\frac{dH}{dt} + f\frac{1}{A^2}\frac{Q|Q|}{2D} - \frac{g}{c}\frac{2\nu}{gE}\frac{\partial \sigma_x}{\partial t}c^2 = 0$$

$$\frac{1}{A}\frac{dQ}{dt} + \frac{g}{c}\frac{dH}{dt} + f\frac{1}{A^2}\frac{Q|Q|}{2D} - gc\frac{2\nu}{gE}\frac{\partial \sigma_x}{\partial t} = 0 \tag{A.8}$$

Rewriting Equation A.8 using finite difference method and Equation A.7:

$$\frac{1}{A}\frac{Q_i - Q_{i-1}}{\Delta t} + \frac{g}{c}\frac{H_i - H_{i-1}}{\Delta t} + \frac{1}{A^2}\frac{f}{2D}Q|Q| - gc\frac{2\nu}{gE}(\sigma_{i-\frac{1}{2}}(t) - \sigma_{i-\frac{1}{2}}(t-\Delta t)) = 0$$

$$\frac{1}{A}(Q_i - Q_{i-1}) + \frac{g}{c}(H_i - H_{i-1}) + \frac{f\Delta t}{2DA^2}Q|Q| - gc\Delta t\frac{2\nu}{gE}(\sigma_{i-\frac{1}{2}}(t) - \sigma_{i-\frac{1}{2}}(t-\Delta t)) = 0$$

$$H_i = H_{i-1} - \frac{c}{gA}(Q_i - Q_{i-1}) - \frac{c}{g}\frac{f\Delta t}{2DA^2}Q|Q| + c^2\Delta t\frac{2\nu}{E}(\sigma_{i-\frac{1}{2}}(t) - \sigma_{i-\frac{1}{2}}(t-\Delta t) = 0$$

$$H_i = H_{i-1} - \frac{c}{gA}(Q_i - Q_{i-1}) - \frac{f\Delta x}{2gDA^2}Q|Q| + c\Delta x\frac{2\nu}{E}(\sigma_{i-\frac{1}{2}}(t) - \sigma_{i-\frac{1}{2}}(t-\Delta t) = 0 \quad \text{(A.9)}$$

The negative root, $\lambda = -\frac{c}{g}$, gives the C^- compability equation.

$$C^-: \quad \frac{1}{A}\frac{dQ}{dt} - \frac{g}{c}\frac{dH}{dt} + f\frac{1}{A^2}\frac{Q|Q|}{2D} + \frac{g}{c}\frac{2\nu}{gE}\frac{\partial\sigma_x^2}{\partial t}c^2 = 0$$

$$\frac{1}{A}\frac{dQ}{dt} - \frac{g}{c}\frac{dH}{dt} + f\frac{1}{A^2}\frac{Q|Q|}{2D} + gc\frac{2\nu}{gE}\frac{\partial\sigma_x^2}{\partial t} = 0 \quad \text{(A.10)}$$

Using the same method, rewriting Equation A.10 using Equation A.7:

$$\frac{1}{A}\frac{Q_i - Q_{i+1}}{\Delta t} - \frac{g}{c}\frac{H_i - H_{i+1}}{\Delta t} + \frac{1}{A^2}\frac{f}{2D}Q|Q| + gc\frac{2\nu}{gE}(\sigma_{i+\frac{1}{2}}(t) - \sigma_{+-\frac{1}{2}}(t-\Delta t)) = 0$$

$$\frac{1}{A}(Q_i - Q_{i+1}) - \frac{g}{c}(H_i - H_{i-1}) + \frac{1}{A^2}\frac{f\Delta t}{2D}Q|Q| + gc\Delta t\frac{2\nu}{gE}(\sigma_{i+\frac{1}{2}}(t) - \sigma_{i+\frac{1}{2}}(t-\Delta t)) = 0$$

$$H_i = H_{i+1} + \frac{c}{gA}(Q_i - Q_{i+1}) + \frac{c}{g}\frac{f\Delta t}{2DA^2}Q|Q| + c^2\Delta t\frac{2\nu}{E}(\sigma_{i+\frac{1}{2}}(t) - \sigma_{i+\frac{1}{2}}(t-\Delta t)) = 0$$

$$H_i = H_{i+1} + \frac{c}{gA}(Q_i - Q_{i+1}) + \frac{f\Delta x}{2gDA^2}Q|Q| + c\Delta x\frac{2\nu}{E}(\sigma_{i+\frac{1}{2}}(t) - \sigma_{i+\frac{1}{2}}(t-\Delta t)) = 0 \quad \text{(A.11)}$$

The term $\frac{c}{gA}$ is the characteristic impedance of the pipe, denoted as B, and the term $\frac{f\Delta x}{2gDA^2}$ is the resistance coefficient of the pipe, denoted as R. These denotations are used in Equation A.9 and A.11, yielding the final expressions:

$$H_i = H_{i+1} - B(Q_i - Q_{i+1}) - RQ|Q| + c\Delta x\frac{2\nu}{E}(\sigma_{i+\frac{1}{2}}(t) - \sigma_{i+\frac{1}{2}}(t-\Delta t)) = 0 \quad \text{(A.12)}$$

$$H_i = H_{i+1} + B(Q_i - Q_{i+1}) + RQ|Q| + c\Delta x\frac{2\nu}{E}(\sigma_{i+\frac{1}{2}}(t) - \sigma_{i+\frac{1}{2}}(t-\Delta t)) = 0 \quad \text{(A.13)}$$

B

Matlab code

B.1 Main program

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MASTER
% Authors: Daniel Edebro, Oskar Lindgren, Janna Hempel
%
% Main program that runs an iterative process for a specified time, calling
% the fluid mechanics solver, moc.m, and the structure mechanics solver,
% getBeamForces.m. The program calls the sub programs; initilizing the
% pressure and the flow, and defining the loads.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
clc
close all
global pipes BC  %Global matrices for pipe propterties and boundary conditions

t = 0;           %Initilize time t
t_break = 0.3;   %When to stop the iteration
dt = 0.1e-3;     %Timestep

%%%%%%%%%%%%%%%%%%%%%%% Generates matrices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[pipes, BC] = readInputFluid();         %Pipes and boundary conditions
[pipeGeo, supportGeo,anchorList,loadList,DOFcnt,inputpathname] = ...
readInputStructure();                   %Structure input
[K,M,bc] = writeMatrixAndBC(pipeGeo,supportGeo,anchorList,DOFcnt);
                                        %Stiffness, mass, boundary
```

```
[loadDefFile, loadDefPath] = uigetfile({'*.mat','mat-files (*.mat)';'*.txt',
'Text files (*.txt)';'*.*','All files (*.*)'}
,'Read  load definition...');
if loadDefFile == 0, return; end
if ~strcmp(loadDefPath(length(loadDefPath):length(loadDefPath)),'\'),
loadDefPath = [loadDefPath,'\']; end

d0_old = zeros(DOFcnt,1);       %Old displacement, from last timestep
v0_old = zeros(DOFcnt,1);       %Old velocity, from last timestep

N_old = cell(size(pipes,1),1);  %Old forces, from last timestep
for i = 1:size(pipes,1);
    for j = 1:size(pipes{i,1},2)
        N_old{i} = zeros(j,1);
    end
end

moc(5000,0);                    %Fluid solver: initialzing the matrices
                                %steady state

loadDefinition = load([loadDefPath,loadDefFile],'-ascii');
%Defining where the load is applied


C = 0.000*M+0.003*K;            %Damping matrix

%%%%%%%%%%%%%%%%%%%%% Time iteratation parameters %%%%%%%%%%%%%%%%%%%%%%
result = zeros(t_break/dt,7);   %Time history for each time step
cnt = 0;                        %Counting the time steps


%%%%%%%%%%%%%%%%%%%%% Iteration of the solution %%%%%%%%%%%%%%%%%%%%%%%%
while 1
cnt = cnt + 1;                  %Time step
t = t + dt;                     %Time

%%%%%%%%%%%%%%%%%%%%% Starting the fluid solver %%%%%%%%%%%%%%%%%%%%%%%%
%Iterating one timestep at the time
moc(1,t);

%%%%%%%%%%%%%%%%%%%%% Time integration parameters %%%%%%%%%%%%%%%%%%%%%%
%Generating load vector
ntimes = 0;
```

```
nhist = 1:DOFcnt;
ip = [dt dt 0.25 0.5 size(ntimes,2) size(nhist,2) ntimes nhist];
f = loadvector(loadDefinition,pipes,pipeGeo,DOFcnt);

%%%%%%%%%%%%%%%%%%%%% Time integration %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integration in second-order systems
[Dsnap,D,V,A] = step2(K,C,M,d0_old,v0_old,ip,f,bc);
d0_old = D(:,2);            %Updating old displacements
v0_old = V(:,2);            %Updating old velocities

%%%%%%%%%%%%%%%%%%%% Calculating the stresses %%%%%%%%%%%%%%%%%%%%%%%%%
    for k = 1:size(pipes,1)
        %Extracting forces, shear stresses and momentum
        [es,edi,eci] = getBeamForces(pipeGeo,k,D,size(pipes{k,1},2));
        N = es{1,2}(:,1);                       %Force from a pipe element
        A = pipeGeo{k,3}(1,3);                  %Pipe cross-sectional area
        deltaSigma = (N - N_old{k})/(A);        %Change in stress
        N_old{k} = N;                           %Updating old foces
        pipes{k,4} = deltaSigma;                %Saving stress in pipe-cell
    end

%%%%%%%%%%%%%%%%%%%% Updating result matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%
%Saving results from each time step
result(cnt,1) = t;                                  %Time vector
%Pressure head
result(cnt,2) = pipes{1,1}(1,size(pipes{1,1},2));   %First pipe last node
result(cnt,3) = pipes{2,1}(1,1);                    %Second pipe first
%Flow
result(cnt,4) = pipes{1,2}(1,size(pipes{1,1},2));   %First pipe last node
result(cnt,4) = pipes{2,2}(1,1);                    %Second pipe first node
%Displacement
result(cnt,6) = D(size(pipes{1,1},2)/2,2);
%Force
result(cnt,7) = f(size(pipes{1,1},2)/2,2);

%%%%%%%%%%%%%%%%%%%% Stop the iteration %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       if t > t_break
            break;
       end

end
```

```
%%%%%%%%%%%%%%%%%%%% Plot of the result %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1)                           %Pressure head, first pipe last node
plot(result(:,1),result(:,2));

figure(2)
plot(result(:,1),result(:,4));      %Flow, first pipe last node

figure(3)                           %Pipe struture
plotPipeSystem(pipeGeo,supportGeo)
```

B.2 MOC

```
%moc(nr_of_tsteps,time) - calculates the pressure head and the flow at
%                          the inner nodes of the pipes
% Authors: Janna Hempel, Oskar Lindgren
%
% Input arguments
%   nr_of_tsteps - the number of time steps the iteration will run
%   time - used as input parameter in the boundary conditions tank and
%          valve
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global pipes BC

for i = 1:nr_of_tsteps

%%Calculation of flow and pressure head in internal nodes
    for j=1:size(pipes,1)

        nr_of_nodes = size(pipes{j,1},2);      %Number of nodes in
                                               %respective pipe

           for k = 2:nr_of_nodes-1             %Iteration for each inner
                                               %node

                Ha = pipes{j,1}(2,k-1);
                Hb = pipes{j,1}(2,k+1);
                Qa = pipes{j,2}(2,k-1);
                Qb = pipes{j,2}(2,k+1);

                sigmaconst = pipes{j,3}(1,7);
                dsigma = pipes{j,4}(k);

                B = pipes{j,3}(1,6);
                R = pipes{j,3}(1,5);

                Cm = Hb-B*Qb;
                Bm = B+R*abs(Qb);
                Cp = Ha+B*Qa;
                Bp = B+R*abs(Qa);

                Hp = (Cp*Bm+Cm*Bp)/(Bp+Bm)+(sigmaconst*dsigma);
                Qp = (Cp-Cm)/(Bp+Bm);
```

```
                pipes{j,1}(1,k) = Hp;
                pipes{j,2}(1,k) = Qp;

            end
    end


%%Setting the boundary conditions using sub programs
    for j = 1:size(BC,1)

        BCtype = BC{j,1};

        switch BCtype
          case 'Tank'
                tank(j,time);
          case 'Valve'
                valve(j,time);
          case 'Junction'
                pipe(j);
          case 'Deadend'
                deadend(j);
          case 'Valvedwn';
                valvedwn(j);
          case 'Orifice';
                valvedwn(j);
         end

    end

    for row=1:size(pipes,1)
            pipes{row,1}(2,:) = pipes{row,1}(1,:);
            pipes{row,2}(2,:) = pipes{row,2}(1,:);
    end

end

end
```

B.3 Valve

```
%valve(BCIndex) - calculates the pressure head and the flow before
%                  and after a valve.
% Authors: Janna Hempel, Oskar Lindgren
%
%  Input arguments
%    BCIndex - the index of the certain boundary condition in the BC-matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global pipes BC                     %Calls the global variables
g = 9.81;  %Gravitational constant

pipe1 = BC{BCIndex,2}(1,1);         %Pipe index before the valve
pipe2 = BC{BCIndex,2}(1,2);         %Pipe index efter the valve

nodes1 = size(pipes{pipe1,1},2);    %Number of nodes in pipe1

%Interpolation if the degree of opening is not constant
if max(size(BC{BCIndex,4},1)) > 1
    time = BC{BCIndex,4}(:,1);      %Time vector
    tau = BC{BCIndex,4}(:,2);       %Vector: degree of opening of the valve
    A_const = interp1(time,tau,t);  %Interpolation

%If tau has a fixed value
else
    A_const = BC{BCIndex,4};

end


Ha = pipes{pipe1,1}(2,nodes1-1);    %Pressure before the valve
Qa = pipes{pipe1,2}(2,nodes1-1);    %Flow before the valve

Ba = pipes{pipe1,3}(6);             %Charactaristic impedance of pipe1
Ra = pipes{pipe1,3}(5);             %Resistance coefficient of pipe1

Hb = pipes{pipe2,1}(2,2);           %Pressure after the valve
Qb = pipes{pipe2,2}(2,2);           %Flow after the valve

Bb  = pipes{pipe2,3}(6);            %Charactaristic impedance of pipe2
Rb  = pipes{pipe2,3}(5);            %Resistance coefficient of pipe2
```

```
Di = pipes{pipe2,3}(3);              %Inner diameter
A_vlv = (pi()*Di^2/4)*(A_const);     %Area of the valve

a = (1/2)*(1/(A_vlv^2*g));
b = Ba+Ra*abs(Qa)+Bb-Rb*abs(Qb);
c = Ha+Ba*Qa-Hb+Bb*Qb;

if A_const <= 0.00 || a == Inf;      %Zero flow if the valve is closed
    Qp = 0;
else
    Qp = -b/(2*a)+sqrt((b/(2*a))^2 + c/a);  %Flow
end

sigmaconst_a = pipes{pipe1,3}(1,7);    %Stress constant, pipe1
dsigma_a = pipes{pipe1,4}(nodes1-1);   %Change in stresses before the valve

sigmaconst_b = pipes{pipe2,3}(1,7);    %Stress constant, pipe2
dsigma_b = pipes{pipe2,4}(2);          %Change i stresses after the valve

%Head, node before the valve
Hpa = Ha-Ba*(Qp-Qa)-Ra*abs(Qa)+sigmaconst_a*dsigma_a;
%Head  node after the valve
Hpb = Hb+Bb*(Qp-Qb)-Rb*abs(Qb)+sigmaconst_b*dsigma_b;


%%%%%%%%%%%%%%%%%%%%% Updating the pipes-matrix %%%%%%%%%%%%%%%%%%%%%%%%%
pipes{pipe1,1}(1,nodes1) = Hpa;    %Pressure head at the last node in pipe1
pipes{pipe1,2}(1,nodes1) = Qp;     %Flow at the last node in pipe1
pipes{pipe2,1}(1,1)= Hpb;          %Pressure head at the first node in pipe2
pipes{pipe2,2}(1,1) = Qp;          %Flow at the first node in pipe2

end
```

B.4 Tank

```
%tank(BCIndex) - calculates the pressure head and the flow in the pipe
%                 connected to the reservoir.
%  Authors: Janna Hempel, Oskar Lindgren
%
%  Input arguments
%    BCIndex - the index of the certain boundary condition in the BC-matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global pipes BC                     %Call the global variables.

pipe1 = BC{BCIndex,2}(1,1);         %Pipe index before the reservoir
pipe2 = BC{BCIndex,2}(1,2);         %Pipe index efter the reservoir

%Interpolation if the pressure in the tank not is constant
if length(BC{BCIndex,4}) > 1
        tVector = BC{BCIndex,4}(:,1);        %Time vector
        PVector = BC{BCIndex,4}(:,2);        %Pressure vector
        H_tank = interp1(tVector,PVector,time);  %Interpolation

%If the pressure is constant
else
        H_tank = BC{BCIndex,4};

end

%%Tank located to the left of the pipe
if pipe1 == 0

    pipes{pipe2,1}(2,1) = H_tank;   %Pressure in the first node of the connected pipe
    Hb = pipes{pipe2,1}(2,2);       %Pressure in the second node in the pipe
    Qb = pipes{pipe2,2}(2,2);       %Flow in the second node in the pipe
    B  = pipes{pipe2,3}(6);         %Charactaristic impedance of the pipe
    R  = pipes{pipe2,3}(5);         %Resistance coefficient of the pipe

    Cm = Hb-B*Qb;
    Bm = B+R*abs(Qb);

    Hp = H_tank;                    %Pressure head at the first node in the pipe
    Qp = (Hp-Cm)/Bm;                %Flow at the first node in the pipe
```

```
%%%%%%%%%%%%%%%%%%%% Updating the pipes-matrix %%%%%%%%%%%%%%%%%%%%%%%%
pipes{pipe2,1}(1,1) = Hp;       %Pressure head in the first node in the pipe
pipes{pipe2,2}(1,1) = Qp;       %Flow in the first node in pipe


%%Tank located to the right of the pipe
elseif pipe2 == 0
    nodes = size(pipes{pipe1,1},2);     %Number of nodes in the pipe

    Ha = pipes{pipe1,1}(2,nodes-1);
    Qa = pipes{pipe1,2}(2,nodes-1);
    B  = pipes{pipe1,3}(6);
    R  = pipes{pipe1,3}(5);

    Cp = Ha+B*Qa;
    Bp = B+R*abs(Qa);

    Hp = H_tank;
    Qp = (Cp-Hp)/Bp;


%%%%%%%%%%%%%%%%%%%% Updating the pipes-matrix %%%%%%%%%%%%%%%%%%%%%%%%
pipes{pipe1,1}(1,nodes) = Hp;   %Pressure head at the last node in the pipe
pipes{pipe1,2}(1,nodes) = Qp;   %Flow at he last node in the pipe

end
```

B.5 Pipe

```
%pipe(BCIndex) - calculates pressure head and flow for two connected pipes
% Authors: Janna Hempel, Oskar Lindgren
%
%  Input arguments
%    BCIndex - the index of the certain boundary condition in the BC-matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global pipes BC                     %Calingl the global variables

pipe1 = BC{BCIndex,2}(1,1);         %Pipe index before the pipe connection
pipe2 = BC{BCIndex,2}(1,2);         %Pipe index efter the pipe connection

nodes1 = size(pipes{pipe1,1},2);    %Number of nodes in pipe1

sigmaconst = pipes{pipe1,3}(1,7);   %Stress constant, pipe1
dsigma = pipes{pipe1,4}(nodes1-1);  %Change in stresses at the end of pipe1

Ha = pipes{pipe1,1}(2,nodes1-1);    %Pressure head before pipe connection
Qa = pipes{pipe1,2}(2,nodes1-1);    %Flow before pipe connection
Ba  = pipes{pipe1,3}(6);            %Charactaristic impedance of pipe1
Ra  = pipes{pipe1,3}(5);            %Resistance coefficient of pipe1

Hb = pipes{pipe2,1}(2,2);           %Pressure head after pipe connection
Qb = pipes{pipe2,2}(2,2);           %Flow after connection
Bb = pipes{pipe2,3}(6);             %Charactaristic impedance of pipe2
Rb = pipes{pipe2,3}(5);             %Resistance coefficient of pipe2

Cp = Ha + Ba*Qa - Ra*Qa * abs(Qa);
Cm = Hb - Bb*Qb + Rb*Qb * abs(Qb);

%Pressure at the point of  the connection
Hp = ((Cp/Ba)+(Cm/Bb))/((1/Ba)+(1/Bb))+sigmaconst*dsigma;
%Flow at the point of connection
Qp = (Cp - Cm)/(Ba+Bb);

%%%%%%%%%%%%%%%%%%%%%%% Updating the pipes-matrix %%%%%%%%%%%%%%%%%%%%%%%
pipes{pipe1,1}(1,nodes1) = Hp;    %Pressure head at the last node in pipe1
pipes{pipe1,2}(1,nodes1) = Qp;    %Flow at the last node in pipe1
pipes{pipe2,1}(1,1)= Hp;          %Pressure head at the first node in pipe2
pipes{pipe2,2}(1,1) = Qp;         %Flow at the first node in pipe2
end
```

B.6 GetBeamForces

```
%[es,edi,eci] = getBeamForces(pipeGeo,pipeIndex,D,n)
%                 returns section forces, the displacements and
%                 the evaluation points for the beam element
% Authors: Daniel Edebro, Oskar Lindgren, Janna Hempel
%
% Input arguments
%   nr_of_tsteps - the number of time steps the iteration will run
%   time - used as input parameter in the boundary conditions tank and
%          valve
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

N = size(D,2);
es  = cell(1,N);
edi = cell(1,N);
eci = cell(1,N);

xVec = [pipeGeo{pipeIndex,9}(1),pipeGeo{pipeIndex,10}(1)];
yVec = [pipeGeo{pipeIndex,9}(2),pipeGeo{pipeIndex,10}(2)];
zVec = [pipeGeo{pipeIndex,9}(3),pipeGeo{pipeIndex,10}(3)];
DOF1 = pipeGeo{pipeIndex,7};
DOF2 = pipeGeo{pipeIndex,8};


%%%%%%%%%%%%%%%% Pipe propoerties from pipeGeo %%%%%%%%%%%%%%%%
% pipeGeo{i,3} = [E,G,A,Iy,Iz,Kv,m,j,OD,t,YS,TS]
E  = pipeGeo{pipeIndex,3}(1);
G  = pipeGeo{pipeIndex,3}(2);
A  = pipeGeo{pipeIndex,3}(3);
Iy = pipeGeo{pipeIndex,3}(4);
Iz = pipeGeo{pipeIndex,3}(5);
Kv = pipeGeo{pipeIndex,3}(6);

vertAngle = pipeGeo{pipeIndex,4}(2)*pi/180;
aziAngle =  pipeGeo{pipeIndex,4}(3)*pi/180;

%%%%%%%%%%%% Global vector parallel to the beam local z-direction %%%%%%%%%%%%
c = [cos(vertAngle+pi/2)*cos(aziAngle),cos(vertAngle+pi/2)*sin(aziAngle),...
    sin(vertAngle+pi/2)];

ep = [E,G,A,Iy,Iz,Kv];
q = [0,0,0,0];   % Element load vector [qx,qy,qz,qw]
```

```
for i = 1:N
    ed = D([DOF1,DOF2],i)';
    [es_tmp,edi_tmp,eci_tmp] = beam3s(xVec,yVec,zVec,c,ep,ed,q,n);
    es{1,i} = es_tmp;
    edi{1,i} = edi_tmp;
    eci{1,i} = eci_tmp;
end
```

B.7 LoadVector

```
%f = loadvector(loadDefinition,pipes,pipeGeo,DOFcnt)
%                 returns section forces, the displacements and
%                 the evaluation points for the beam element
% Authors: Daniel Edebro, Oskar Lindgren, Janna Hempel
%
% Input arguments
%   loadDefinition - Defining where the loads are applied
%   pipes - Matrix for pipe propterties from the fluid sheet
%   pipeGeo - Matrix for pipe propterties from the structure sheet
%   DOFcnt - Number of degrees of freedom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

loadCnt = size(loadDefinition,1);  %Defines the count load
f = zeros(DOFcnt,1);               %Force vector for all degrees fo freedom

for i = 1:loadCnt                       %Loop through the count load
    FluidPipe = loadDefinition(i,1);    %Defines where in the load defintion
                                        %matrix the fluid pipe is
    FluidNode = loadDefinition(i,2);    %Defines where in the load defintion
                                        %matrix the fluid node is
    StructPipe = loadDefinition(i,3);   %Defines where in the load defintion
                                        %matrix the structure pipe is
    StructNode = loadDefinition(i,4);   %Defines where in the load defintion
                                        %matrix the structure node is

    pipeArea = pipes{FluidPipe,3}(1,3);    %Defines the pipe area
    rho = 998;                             %Density of water
    g = 9.81;                              %Gravitational constant

if FluidNode == 1                           %First node
        Force = pipes{FluidPipe,1}(2,1)*pipeArea*rho*g; %Force acting
                                                        %on the structure
    else
        nodeCnt = size(pipes{FluidPipe,1},2);   %Last node
        Force = pipes{FluidPipe,1}(2,nodeCnt)*pipeArea*rho*g;
    end

    %Specifying the degrees of freedom
    DOFs = pipeGeo{StructPipe,6+StructNode}(1:3);
    %Generates a vector with the same length as the struture pipe
    pipeVector = pipeGeo{StructPipe,10} - pipeGeo{StructPipe,9};
```

```
    %Creates a unit vector for the beam element
    pipeUnitVector = pipeVector/sqrt(pipeVector(1)^2+pipeVector(2)^2+pipeVector(3)^2);

    if FluidNode == 1
        ForceVector = -pipeUnitVector*Force;  %Defines the force vector, with direction,
    else                                      %if the node in the fluid pipe is 1
        ForceVector = pipeUnitVector*Force;   %Defines the force direction, if the
    end                                       %node in the fluid pipe is the last node

    f(DOFs(1)) = f(DOFs(1)) + ForceVector(1);
    f(DOFs(2)) = f(DOFs(2)) + ForceVector(2);
    f(DOFs(3)) = f(DOFs(3)) + ForceVector(3);
end
```