5

The Simulation of Passive Water Hammer in Pipes

by

Murat Erentürk

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Author .... [illegible]
Department of Mechanical Engineering
May 9, 1996

Certified by .... [illegible]
Peter Griffith
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .... [illegible]
Ain Sonin
Chairman, Departmental Committee on Graduate Students

[illegible]

Acknowledgments

First, and foremost, I would like to thank Professor Peter Griffith for his support and guidance provided during my experience at MIT. Without his patience and encouragment, this work would not have completed.

I would also like to thank my mother İlgül Erentürk and my father Oktay Erentürk for their love, never ending support and invaluable encouragement. Your support all throughout my education gave me high motivation and these years would not be the same without your love.

Last but not least, I would like to thank my friends here and overseas who made the high pace of MIT bareable with their advice, humor and facinating discussions. Without them my challange would have been very hard and boring.

I would also like to acknowlege the Fulbright Commison for the Fulbright scolarship that supported my studies for two years.

The Simulation of Passive Water Hammer in Pipes

by

Murat Erentürk

Submitted to the Department of Mechanical Engineering
on May 9, 1996, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

A variety of feedwater heater drain lines suffer from passive water hammer which is caused by the hot/cold water interface in the piping system. Experiments trying to identify the reasons for pressure spikes and the deviation between the theory and experiments failed because there were different time constants associated with the system and setup was not built to account for all of them. Bubble formation and condensation occurred at different rates and transient phase fluid dynamics changed the pressure profile considerably. For a plant trying to avoid passive water hammer, flow control valve should be cracked opened so little that any pressure spikes occurring before the cold water has been flushed out would be tolerable by the piping supports. After cold water leaves the system, the valve can be fully opened to allow the system to come to steady state. This solution is conservative and does not eliminate the problem completely but is enough for most of the practical purposes. For a detailed transient analysis a computer simulation should be done for each system under consideration.

Thesis Supervisor: Peter Griffith
Title: Professor of Mechanical Engineering

Contents

List of Figures

Table of symbols

Latin Characters

A	Cross sectional area
C	Constant depending on pipe support conditions
C_1	First Coefficient in time constant approximation
C_2	Second Coefficient in time constant approximation
C_{calib}	Calibration constant depending on pressure difference
D_1	Diameter of pipe A and B
D_2	Diameter of pipe C
D_i	Inner diameter
D_o	Outer diameter
E	Young's modulus of elasticity
E_k	Kinetic energy
E_p	Young's modulus of elasticity of pipe material
E_s	Young's modulus of elasticity of steel
E_w	Young's modulus of elasticity of water
K	friction factor
K_e	Bulk modulus of elasticity
K_{pipe}	Friction factor for pipe
L	Length of pipe section
L_1	Length of pipe A+B
L_2	Length of pipe C
P	Pressure
Q	Volumetric flow rate
R	Universal gas constant
Re	Reynolds number
T	Temperature
V	Velocity
V_{ss}	Steady state velocity in the pipe
c	Speed of pressure waves

c_f	Speed of sound in liquid phase
c_g	Speed of sound in gas phase
d_i	Directional parameter
e	Pipe wall thickness
f	Frictional loss coefficient
g	Gravitaitional constant
g_o	Gravational constant correction factor
h	Height of water column inside tank
k	Isentropic gas constant
$\dot{m}$	mass flow rate
p_∞	Atmosperic pressure
p_{tank}	Static pressure inside tank
p_{pipe}	Static pressure just after pipe entrance
s	Entropy of the mixture
s_f	Entropy of liquid phase
s_g	Entropy of gas phase
s_{fg}	Vaporisation Entropy
t	time
$t_{valveopening}$	Valve opening time
v	Specific volume of mixture
v_f	Specific volume of fluid phase
v_g	Specific volume of gas phase
v_{fg}	Vaporisation specific volume change
w_1	Velocity in pipe C before flashing
w_2	Velocity in pipe C after flashing
w_{before}	Velocity in pipe A before flashing
w_{after}	Velocity in pipe A after flashing
x	Quality

Greek Characters:

ΔP	Magnitude of pressure difference
ΔP_f	Frictional Pressure drop
ΔP_I	Inertial pressure drop
ΔP_S	Steady-state pressure drop
ΔV	Magnitude of velocity difference
Δx	Space slice for numerical discritization
ϵ	Roughness of pipe
ζ	Friction factor for elbows, valves etc.
θ	Valve angle
μ_f	Dynamic viscosity of liquid phase
μ_g	Dynamic viscosity of gas phase
λ	Friction coefficient for pipe material
ρ_w	Mean fluid density
ϕ^2	Two-phase multiplier

Chapter 1

Introduction

Water hammer has been a concern to engineers for almost a century. It has caused serious damage in some engineering applications. One of the important areas where water hammer causes problems is power plants especially nuclear power plants. There are instances in which pipes as large as 1 meter in diameter shake, damage the supports and even rupture or leak at pipe welds. After years of research some causes were identified and design precautions were taken to avoid the problem. These were generally water hammer events that occurred in one phase flow. They have been classified and well documented. Unfortunately one phase water hammer events were only a part of a larger problem. Water hammer can occur in various other situations. One of them is the two-phase flow induced water hammer which is known as passive water hammer.

This thesis is a continuation of research to identify the causes of passive water hammer in pipes. Previous experiments were unsuccessful in capturing some of the physics of passive water hammer. The main objective of this research is to identify the causes of failure of the models used to describe the experiments of Sweeney[20] and Wolf[22]. The problem is analyzed from an analytical perspective. A detailed mathematical model of the experimental setup is made in order to simulate it on a computer. Basic principles of fluid mechanics and thermodynamics were applied to build the model. This model is also used to correct the experimental setup so that new experiments can be made in which passive water hammer can be observed.

The second chapter gives a brief summary of water hammer, causes and the analytical framework needed to calculate physical quantities. It also identifies and explains several two-phase flow water hammer mechanisms seen in practical use. The next chapter covers two recent researches done by other MIT students on this subject. The older one is a general water hammer research aimed to analyze both one and two phase water hammer. The newer one focuses on two-phase water hammer and includes experiments and analytical calculations. Brief discussion of the results is also provided. The fourth chapter explains the mathematical models used to simulate the experimental apparatus. It gives detailed descriptions of the procedures and analytical framework that is applied to solve the problem. It also gives the results obtained from the mathematical model. The final chapter explains the physics covered by the simulation which was missed in the experiments. It also gives some suggestions to improve the experimental setup in order to capture more physics of the two-phase water hammer.

Chapter 2

Background

2.1 What is water hammer?

Water hammer was a concern to engineers since the end of the nineteenth century. Pioneers like Joukowsky [7] and Allievi [1] investigated the phenomena. Initial research was aimed to aid the design of large scale water pumping installations. Water hammer is produced by a sudden change in velocity of flow in a conduit. This may result from

- The stopping and starting of pumps
- The turning off of a valve at a wash basin (especially assisted by a spring loaded return mechanism)
- The sudden flow demand by an automatic fire protection system
- The fail safe systems which require rapid operation such as closure of a liquid fuel supply
- A resonance build-up due to a dynamically unstable component in a liquid line
- Mechanical failure of an item such as a valve (a casualty of age or wear from repeated use)

One basic scenario in which water hammer results can be described as follows. Assume there is a long pipe section which is connected to a reservoir from one end and to a valve on the other. Initially the valve is open and the water inside is flowing in steady state conditions. Then suddenly valve is closed. At that moment only the valve side of the water knows the pressure differential has changed but water tries to move. As the liquid is almost incompressible, a pressure wave develops moving towards the reservoir. From practical point of view this wave moves at the speed of sound in the media. When it hits the reservoir it bounces back and comes to the valve again. This shock causes the pipe to move depending on the pressure differential. The wave continues to move back and forth but attenuates and dies out after some time. The nature and extent of the damage depends on many factors. The most important one being the mass of liquid put into motion.

Water hammer is the propagation of energy as in transmission of sound and is known from basic physics as a wave motion which is associated with the elastic deformation of the medium. The celerity of sound waves, is expressed as

$$c = \sqrt{\frac{K_e}{\rho_w}} \tag{2.1}$$

For the problem under consideration clerity can easily be taken as the speed of sound in the medium.[16]

If the pipe walls are thin, the elastic deformation during water hammer is high enough to affect the speed of sound. This, in return, affects the amplitude of pressure. The wave speed in such a situation is given by

$$c = \sqrt{\frac{K_e/\rho}{1 + [(K_e/E)(D/e)]C}} \tag{2.2}$$

The constant C ranges between 0.85-1.00 but can be taken as 0.91 as an approximation. The wave speed can be greatly reduced if the liquid in the pipeline entrains even a small amount of air.

The basic water hammer equations are based on the principle of conservation of mass and momentum of the fluid in the pipe. They show that the pressure pulse created by water hammer is given as

$$\Delta P = \rho_w c \Delta V \tag{2.3}$$

which is known as the Joukowsky Equation. The pressure pulse with a constant magnitude determined by equation 2.3 propagates in the pipe at the wave speed until it encounters any changes in piping configuration resulting in wave reflections.[18]

Generally one phase water hammer is caused by pump transients, abrupt changes in valve settings like check valve closure or relief valve operation. It can also arise from main steam turbine trip and the filling of normally empty systems. Conventional water hammer analysis is enough to determine the extent of forces and change the design to accommodate the situation which are verified by experimental and field measurements. Parmakian's book [13] is an excellent source on this subject.

There are several other water hammer events, which can result in severe water hammer effects. These events are more complex to analyze because they usually occur in a transient two phase flow environment. They can effect the plant operation severely and often cause significant damage to pipe supports or piping components.[18]

2.2 Types of two phase water hammer

There are several mechanisms identified which can cause severe damage on piping systems. The most important ones will be briefly explained here.

2.2.1 Water cannon

This mechanism was reported to occur where steamlines discharge into the suppression pool. When steam exhausts into a pool of subcooled water at a sufficiently low velocity a pocket of steam is often trapped above the subcooled liquid interface. This usually arises when exhaust valve is fully or partially closed. valve. The trapped steam rapidly condenses and draws water almost instantly into the exhausted line. The water impacts onto the fully or partially closed valve. This causes a substantial pressure pulse, which travels through the water filled pipe creating significant forces on the pipe segments.[18]

2.2.2 Steam/water counterflow

This mechanism occurs in badly designed feedwater lines leading to the spargers in PWR steam generators. There is a large interfacial area between the steam and subcooled water during steam and water counterflow in a horizontal pipe. This is generally the case when small flow of subcooled water is injected into a large horizontal pipe leading to a reservoir of high pressure steam. A high velocity steam flow counter to the direction of liquid water flow results from rapid condensation on the liquid surface. With this countercurrent flow, transition to slug flow will occur and a steam pocket will be formed. Steam will condense rapidly causing a large differential pressure across the slug of water formed by the trapping wave. This slug is accelerated into the collapsing void resulting in a moderate water hammer. Another possibility is, as filling continues, the water bridges the elbow and forms an isolated steam pocket. Steam inside condenses dropping the pressure. This causes the slug to accelerate. After the steam pocket collapses, the slug is rapidly stopped and large pressure waves can be generated. These waves also propagate through the piping system and can result in severe damage. [18]

2.2.3 Steam pocket collapse

Steam pocket collapse transient can result from pressurized water entering the bottom of a vertical pipe or one that is inclined upward more than 3° from horizontal. The filling rate inertia of the liquid is the key factor for the filling rate for this mechanism. The pressure of the pump or other filling device should also be taken into consideration. the leaking of steam or hot water causes steam filled pipe at any plant elevation which in return flashes from a higher pressure region. A waterhammer will not occur with a slow rate of top filling. However, if the top filling rate is faster than the bubble rise velocity, the slug flow pattern will occur. The slug flow fills the pipe and bubbles collapse resulting in a water hammer. A pressure wave will then propagate through the water filled piping.[18]

2.2.4 Low pressure discharge

This type of water hammer is the major topic of this thesis so only a summary will be given here.

Hot water entering a low pressure line causes significant transients in many power plants, especially in the heater drain dump discharge condenser. There is pressurized water at saturation temperature which is connected to a discharge pipe containing stagnant subcooled water adjacent to the valve inlet. When the valve opens, the subcooled water quickly passes the valve when hot water enters it a water hammer may occur due to the flashing. The velocity of the saturated water and vapor will be lower than the velocity of the subcooled water. Equation 2.3 can be used to predict the pressure surge from this change.[18]

2.2.5 Water slug

Systems which collect condensate upstream of a closed valve or that could form water slugs in normally empty steam discharge lines are prone to steam propelled water slugs.

When the valve is opened, steam flows to accelerate the water slug. Even a relatively modest steam pressure across the slug can result in high slug velocity and large forces. This is especially significant when this rapidly moving slug passes an elbow or hits a restriction in the piping system. The initial slug might not completely fill the pipe cross section, when condensate collects in a normally empty line, The flowing steam could "sweep up" the water into a slug resulting in a significant water hammer.[18]

2.2.6 Valve slam

Rapid valve actuation transients are defined as rapid closure of a stuck-open check valve or abnormal valve opening or closing events such as those due to actuator failure. These sudden changes in flow velocity cause abrupt pressure pulses which are more severe than normal valve opening or closing transients. Local column separation and rejoining could occur on the low pressure side of these rapidly closing valves.[18]

2.2.7 Column rejoining

This type of water hammer occurs when there is a vertical pipe section longer than 10 meters. When system is not operational, check valve may leak or pump may trip causing a vacuum in the long vertical section. When pump restarts, water will rise causing the vacuum to collapse. This will result in a substantial pressure pulse which can damage the piping system.[18]

2.3 Origin of the problem

A variety of heater drain pipes found in power plant steam systems sometimes suffer from one of the water hammer mechanisms described in the previous section. The one mentioned as low pressure discharge occurs as a result of flashing in a downstream flow restriction. This flashing decreases the flow rate which in return can cause a water hammer. A simple schematic is shown in figure 2-1.

The tank at the left represents the bottom of a feed water heater. During normal operation condensate collects in the tank and flows to the condenser from the drain pipe. The piping system contains valves so that operator can control the flow. During a shut-down this condensate cools down. If the time is long enough it may well be around room temperature. When the plant restarts operation, fresh hot condensate from feed water heater enters the drain line, pushing the cold water. There is a hot/cold water interface in between and heat transfer is slow compared to the flow. When this interface reaches the partially open flow control valve, pressure drops

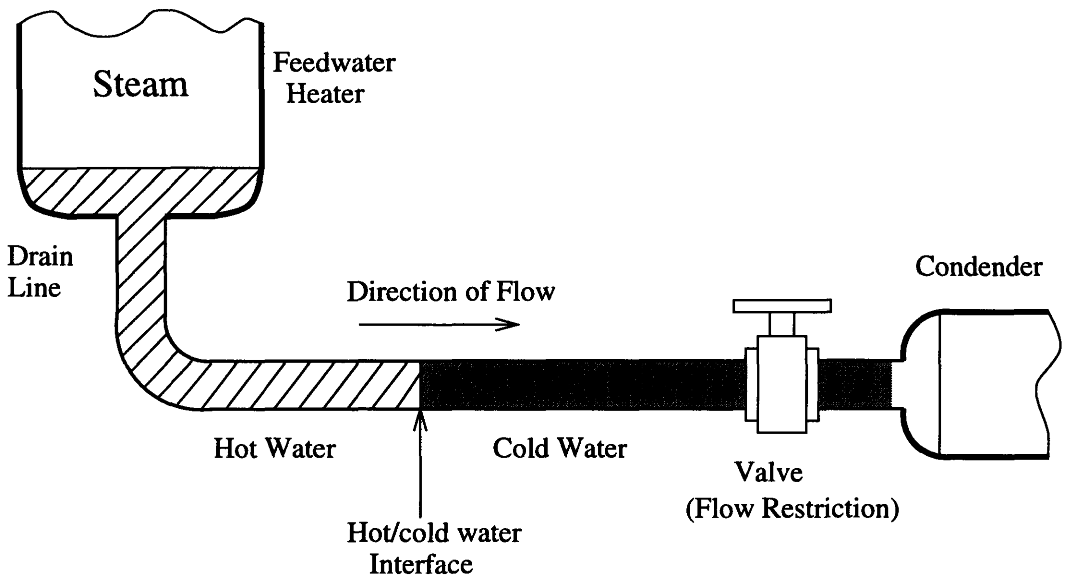

Figure 2-1: Schematic of a feedwater heater drain line

sharply due to the friction of valve against flow. As the condensate passes through the valve, it flashes and the pressure drop across the valve increases further causing the condensate upstream of the valve to quickly decelerate. This deceleration causes the event known as passive water hammer.[22]

One of the actual systems had a piping of 100ft (30.5m) with an internal diameter of 22 inches (559mm). The system did not have too many fittings. Hot operating temperature was $260°F(127°C)$ with a pressure of 81 psig (660kPa absolute). The valve was much smaller than the pipe with an internal diameter of 6 inches(152mm). Pipe damage was reported. [18]

Another system had 98ft (30m) of piping with an internal diameter of 20 inches (508mm). The system had considerable number of fittings (7 bends, 2 tees). Hot operating temperature was $200°F(93°C)$ with a pressure of 30 psia (207kPa absolute). Valve diameter was 16 inches(406mm). Support and pipe damage were reported due to water hammer.

Chapter 3

Experiments

There were two researches done under the same supervisor on passive water hammer in pipes before this thesis. Both were carried by students attending MIT. The first one is a MS thesis done by E. Sweeney. It is available in MIT libraries so will be summarized shortly. The second one is a Senior project presented at Techniche Universitaet Muenchen by Arne Wolf and will be explained in greater detail.

3.1 Sweeney's Experiments

Sweeney's experiments mainly focused on steam bubble collapse induced water hammers and rapid valve closure. Late in his experiments he dealed with passive water hammer transients given in Appendix D of his thesis. [20]

The experimental apparatus he used consisted of a tank and horizontal and vertical pipe sections. The tank was a 30 gallon (113.4 liters) cylindrical vessel that had several lines entering it to admit air, steam and water controlled by valves. Leading out of the tank was a one inch I.D. (25.4mm) horizontal insulated copper pipe which was 38 feet long including bends. After a tee branch there was a 3 foot vertical section with the 1 inch diameter. This was connected to a 1 foot pipe having an internal diameter of three eighths of an inch by a reduction. At the end of this reduced section was a ball valve which was open to atmospheric pressure. There were 2 other valves along this part of the piping system. The first one was at tank exit, the second was after

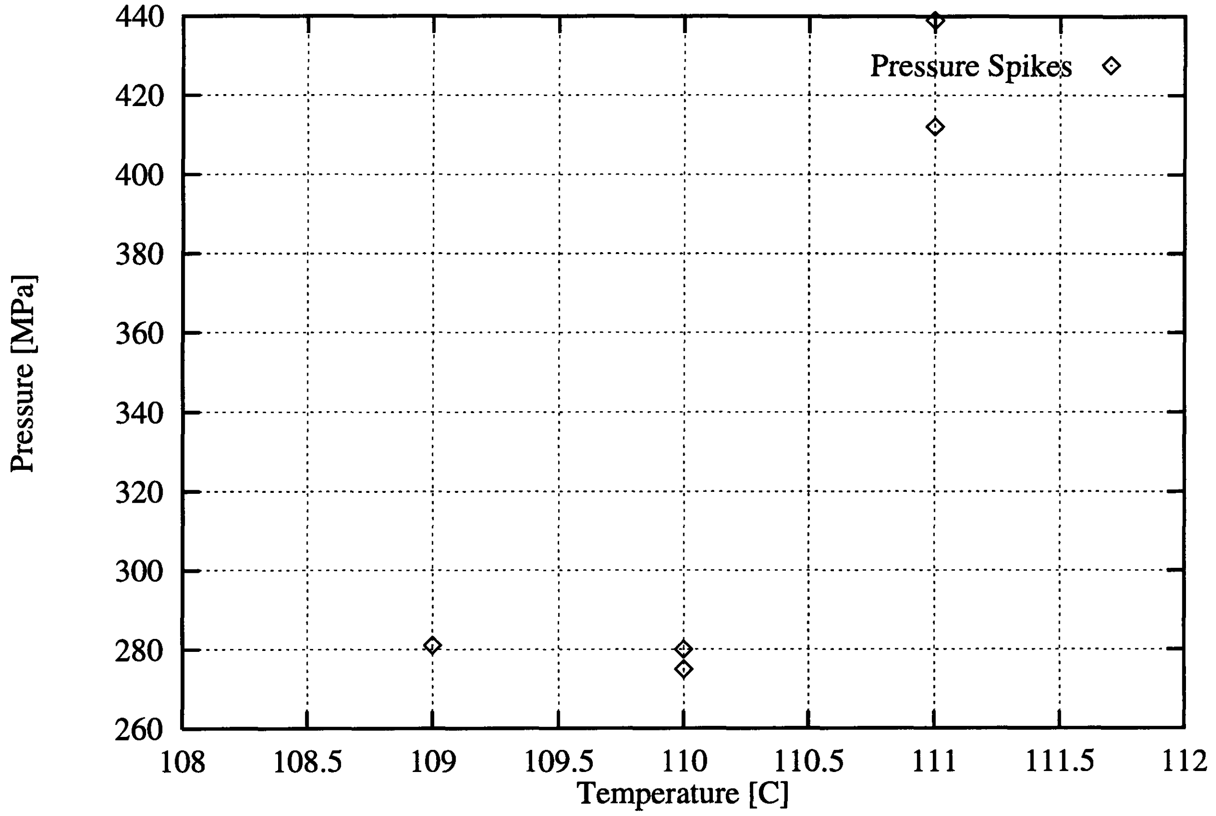

Figure 3-1: Peak pressures obtained by Sweeney with 30 psig Tank pressure

the tee. Data was collected by pressure transducers and a personal computer was used to obtain a voltage-time trace.

A variety of pressure and temperature combinations were tried to get a high pressure surge. Highest pressure surge occurred at a driving pressure of 30 psig (308 kPa) and $231°F(111°C)$. The maximum pressure recorded was 49 psig (439 kPa) and the pressure surge occurred at the expected time of the cold/hot water interface passing through the flow restriction. The valve opening time was approximately 0.1 seconds. Between the time of valve opening and pressure spike occurrence, there was considerable fluctuation in the reading from the pressure transducer which was attributed to the dynamics involved in valve opening.

Figure 3-1 is a plot of runs made with a tank driving pressure of 30 psig and a variety of tank temperatures. Some of the runs did not exhibit pressure spikes after the valves were opened. Instead they exhibited a pressure rise that did not overshoot the steady state pressure. Tests were also run for a flow reduction from one inch I.D.

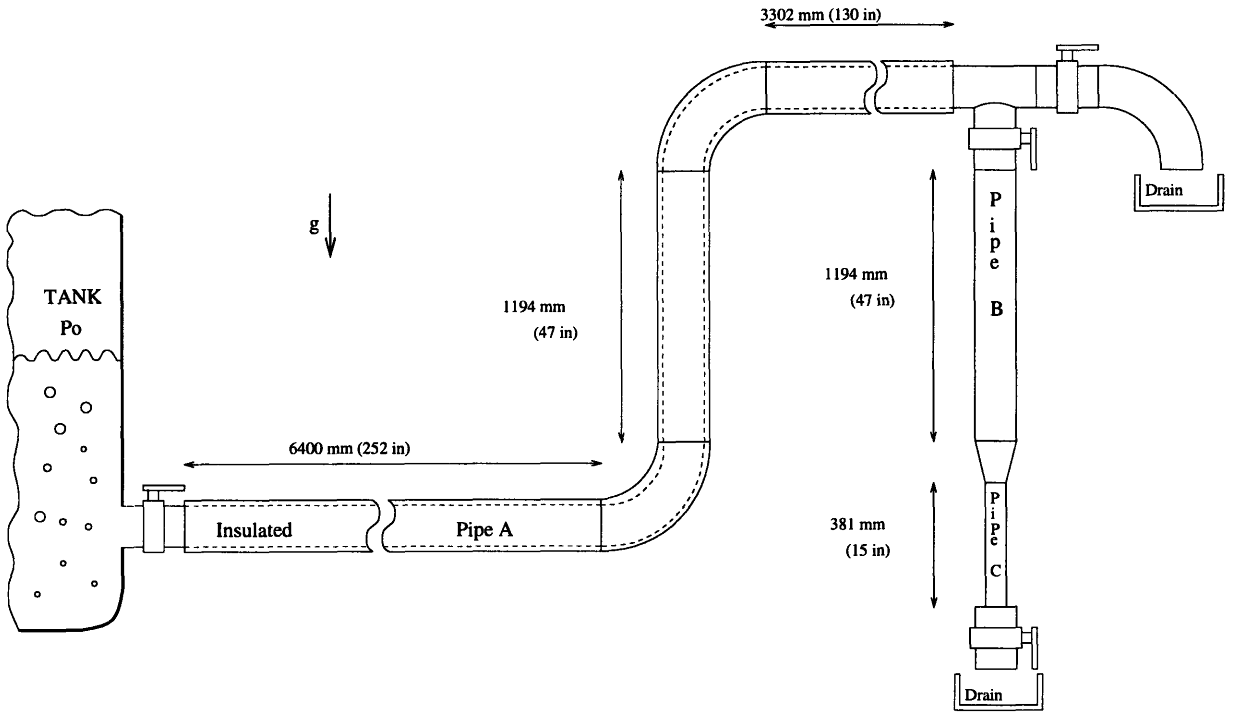

Figure 3-2: Experimental Setup used by Arne Wolf which gave the largest pressure spikes.

to one-half inch I.D. There was considerable scatter in the results.

3.2 Wolf's Research

3.2.1 Experiments

The main elements and goals of Wolf's experiments were the same with Sweeney's but apparatus was more complicated. Although he tried several different setups the one giving the largest pressure spikes which can be seen in figure 3-2 can be described as follows:

Water used in the experiment came from a steel vessel which was kept under 30 psig (308 kPa absolute) constant pressure during the experiment. A cold water supply, two steam supplies to heat the water, a drain, a vent and the outlet pipe were connected to the vessel. Pipe A left the vessel horizontally entering a ball valve (Valve 1). Then there was a straight pipe of 21 ft(6.4m) followed by a vertical elbow

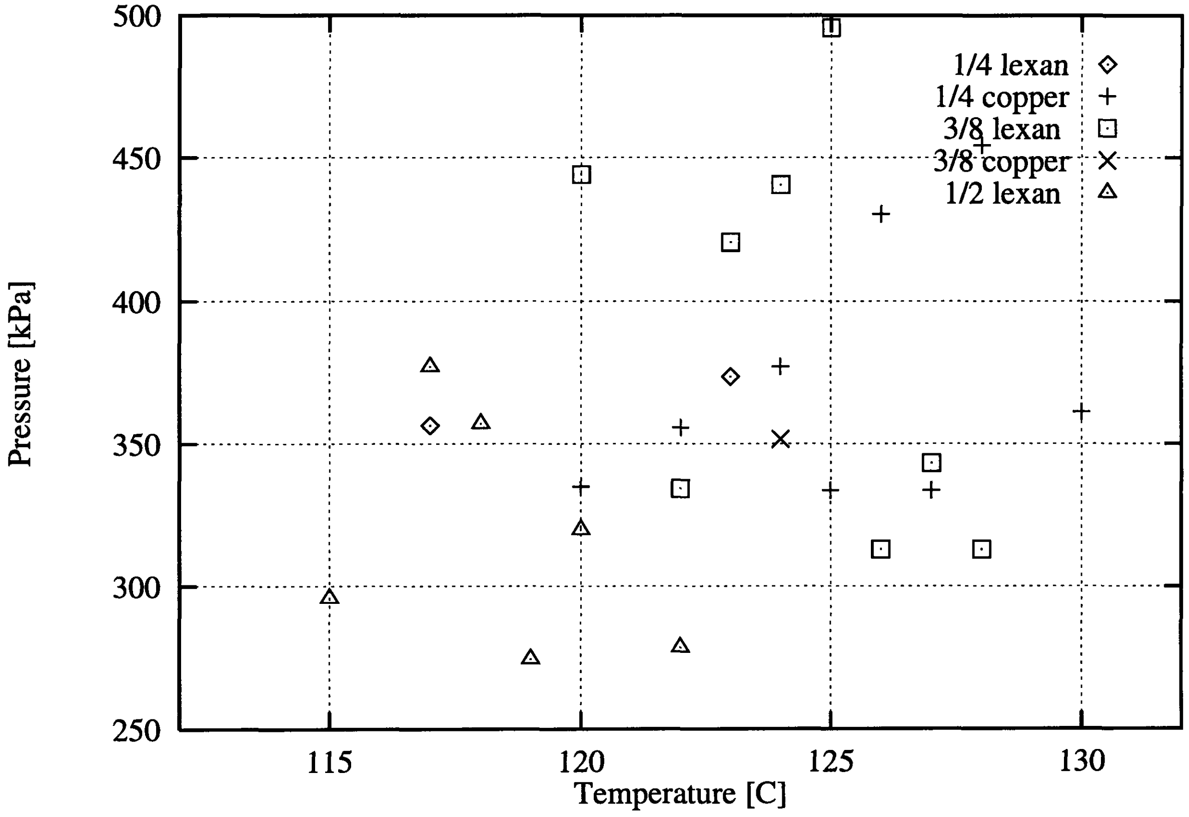

Figure 3-3: Peak pressures obtained by Arne Wolf in the third rig

and a vertical pipe section which was 47 inches(1.2m). After a second elbow there was a horizontal pipe section of 130 inches(3.3m). This pipe and the elbows were made of galvanized steel and were insulated to reduce the heat loss.

At this point the pipe branched in a tee. The through part was connected to an exit valve (Valve 2). The branch part was connected to a ball valve (valve 3) and the vertical pipe named B. This pipe section was 47 inches(1.2m) long and had 2 inches(50.8mm) I.D. Pipe B was connected to pipe C by a smooth contraction. Being the smallest pipe in the apparatus pipe C was chosen to be 1, 0.75 or 0.5 inch in I.D. Different pipe materials were tried to see the effect of pipe elasticity on the magnitude of pressure spikes. Among the three materials Plexiglas, lexan and copper, the first was the softest. The length was chosen to be 20 times the diameter which is 380 millimeters in average. Finally pipe C was connected to an enlargement to change the diameter to 2 inches and an exit valve(valve 4).

To prepare the setup, first Pipes A and B were filled with cold water from the unheated vessel to avoid air trapping inside. All valves were closed. Then the vessel was pressurized and heated up to the desired temperature. Valve 1 and valve 4 were opened to heat pipe B until its wall temperature exceeded 230°F(110°C). Then valve 4 was closed and operator waited so that system came to equilibrium. After the temperature in pipe B fell down to 203°F(95°C) valve 3 was closed and valve 2 was opened. This ensured that pipe A had fresh hot water. Then valve 2 was closed, valve 3 was opened which created a hot/cold water interface. The experiment was ready to begin. To start the experiment valve 4 was slam opened. After 2 to 3 seconds it was closed again and the run was saved and plotted.

Figure 3-3 shows the peak pressures caused by the spikes obtained by using this rig. The magnitude of the spike depended on several factors. First of all there seemed to be a certain temperature that provided the largest spike for a given flow restriction diameter. Below and above this temperature the magnitude of the spikes decreased. The optimum temperature if we can call it increased as diameter of pipe C decreased and almost reached saturation temperature for a value of 0.5 inch. Second the diameter ratio (D_C/D_B) was an important parameter. Largest spikes occurred around a value of 3/8. Finally pipe material was effecting the spikes. Steel pipe caused higher spikes than Plexiglas whereas use of copper was unsatisfactory. It was predicted that nucleation properties of plastic were less than optimum. Initial bubbling was slight and this inhibited the development of further bubbles.

3.2.2 Calculations

The calculations for estimating the magnitude of the spike were of iterative nature and are very complicated to do by hand. For a given set of inlet conditions and discharge pressure, it is necessary to use trial and error procedure in order to find resulting steady state void fraction and velocity profiles. For a given set of inlet conditions the pressure spike was calculated. A check was made to see if the flow at the discharge was choked. If it was, the assumed discharge pressure was increased, a new mass velocity chosen and the pressure, void and velocity profile were calculated again. When the choked mass flow rate and assumed exit pressure were compatible, the calculation was complete. This flow rate was used to calculate the velocity of the fluid in the pipe upstream of the flow restriction by use of pipe to flow restriction area ratio. A brief description of the formulae used is provided here to give the underlying ideas.

Wolf used the Joukowsky relation [7] which can be rewritten as

$$\Delta P = \rho_w c(w_{after} - w_{before}) \tag{3.1}$$

Speed of pressure waves was given by

$$c = \sqrt{\frac{E_w}{\rho_w \left[1 + \frac{E_w(D_o+D_i)}{E_s(D_o+D_i)}\right]}} \tag{3.2}$$

Velocities were calculated from Bernoulli's equation. Velocity before the water hammer was found from:

$$p_{tank} + \rho_w g h = p_{pipe} + \rho_w \frac{{w_1}^2}{2} + 2\Delta p_v + \rho_w \frac{{w_1}^2}{2}\left(\lambda_1 \frac{L_1}{D_1} + \sum \zeta_i\right) \tag{3.3}$$

and velocity after was found from

$$p_{tank} + \rho_w g h = p_\infty + \rho_w \frac{{w_2}^2}{2} + \Delta p \tag{3.4}$$

where

$$\Delta p = 2\Delta p_v + \rho_w \frac{{w_1}^2}{2}\left(\lambda_1 \frac{L_1}{D_1} + \sum \zeta_i\right) + \rho_w \frac{{w_2}^2}{2}\left(\lambda_2 \frac{L_2}{D_2} + \sum \zeta_i\right) \tag{3.5}$$

The two-phase flow was regarded to be in steady homogeneous equilibrium. This model treats the mixture as if it consisted of one component only having mean fluid properties. The pressure drop has to be calculated starting with the liquid phase. This pressure drop is then multiplied by so called "two-phase multiplier" which is given as

$$\phi^2 = \left[1 + x\left(\frac{v_g - v_f}{v_f}\right)\right]\left[1 + x\left(\frac{\mu_g - \mu_f}{\mu_f}\right)\right]^{-0.25} \tag{3.6}$$

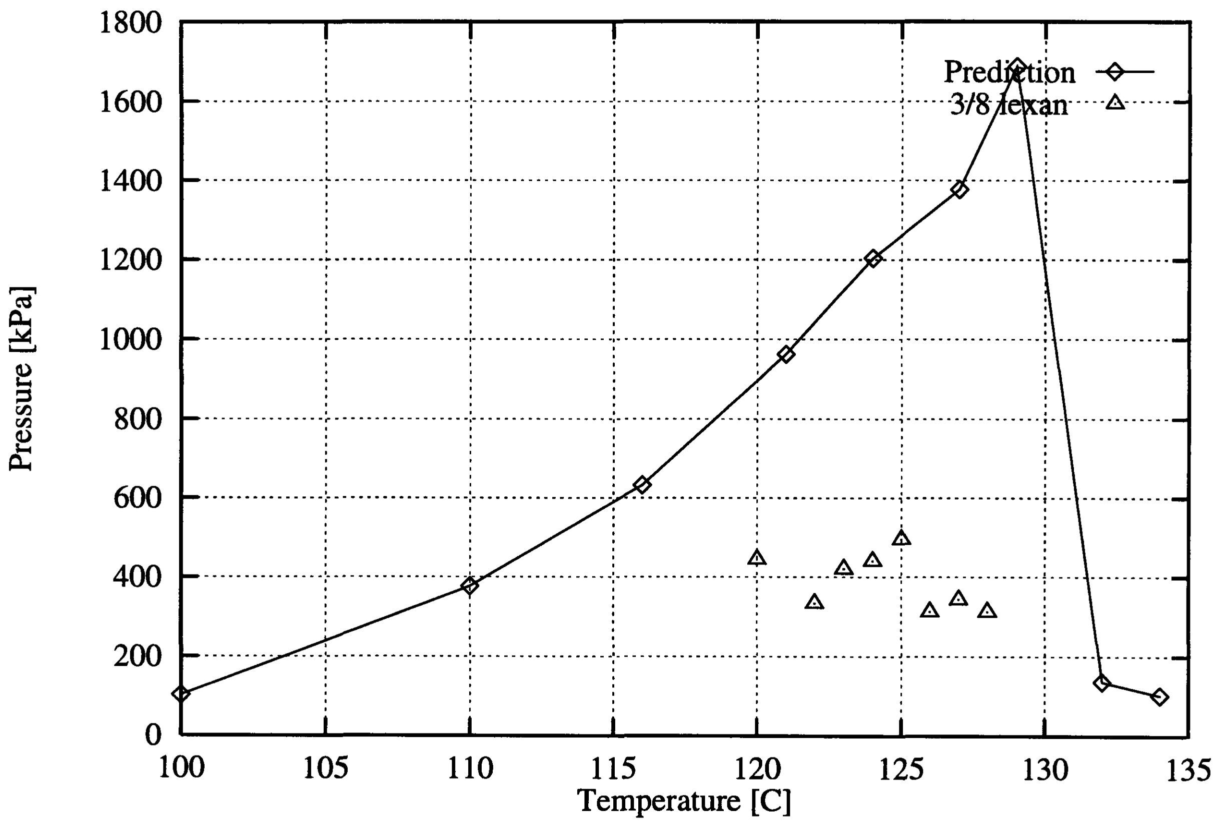

Figure 3-4: Comparison between experimental results of third rig and theory

to obtain the correct value of pressure drop which can be expressed as

$$\Delta P_{two-phase} = \phi^2 \Delta p_{single-phase} \tag{3.7}$$

3.2.3 Results

The method described above was used to predict the peak pressures in the experiments. A set of results from a specified set of conditions is plotted in Figure 3-4.

The curve is of bell shape but due to the number of sampling points it is crudely understandable. Clearly there is no spikes below saturation temperature of the environment and above the saturation temperature of the tank. (In the case of the figure 135°C) The values show absolute pressures so they start at 100kPa instead of 0. It is evident that the model used is overpredicting the spikes as experimental data is well below the model as shown by triangles.

A couple of different sources to evaluate the two-phase steady state flow were used to calculate the velocity after flashing. Using Moody [10] led to an overprediction of the pressure spike by a factor of 6. The sonic speed was also lower than predicted by Moody [10]. Two-phase homogeneous model did not describe the problem well enough because of the departure from thermal equilibrium. The bubble nucleation was also not good due to deviations from thermal equilibrium and the short length of pipe C. Compared to Sweeney's data the scale up in pipe diameters had little influence on the peak values. A shift to higher temperatures can be noticed at least regarding the maximum spikes. Flow regime observations were possible in pipe C when Plexiglas was used. When Plexiglas was used as a pipe material in pipe C (figure 3-2) the exit was always in bubbly flow. The void fraction was less than 50%, while it was expected to be 80%. if the flow were in equilibrium. Slugging, which might be expected for the larger exit void fraction runs, was never either visible nor audible.

The purpose of this work is to understand the reasons for the deviation shown in figure 3-4 and to show how experiments should be run in order to show or eliminate the experimental spikes reported on figure 3-4.

Chapter 4

Mathematical Models

This chapter gives a description of the mathematical models used to simulate the physical setup used in Wolf's experiments. A computer program was written in C language to make it portable which enables it to be compiled in both $UNIX^{TM}$ and PC environments. A general pressure element approach was taken to generalize the problem. This property enables the user to try different pipe setups and transient scenarios to see the dynamic behavior of the system. Valve dynamics was also modeled to see the effects of valve opening and closing times. This variable can change the system behavior significantly. The simulation solved a set of first and second order differential equations with variable properties. Water and steam properties were mathematically modeled to reflect temperature variations. Original program code can be seen at the appendix and a detailed analysis of the code will not be given here.

4.1 General control volume

The mathematical model is assumed to be assembled from control volumes which may have different frictional properties. The elements with their related parameters are given in a data file and the program sets up a table of these elements. The control volumes that are in the experimental setup are predefined in the program. These include pipe sections, ball valves, elbows, expansions and contractions. However the list can be extended to cover other elements like mitre bends and other fittings.

The frictional resistance is a function of velocity in the element. It is important to calculate the mass flow rate passing through so all the elements have one property in common which is the diameter. Using this data, the program calculates the velocity inside the element which is then used to calculate the frictional loss coefficient. Although the velocity changes throughout the simulation, dynamics of change of frictional effects are not included in the model. Thus friction loss is calculated as if the system is in steady-state operation in each time slice. This quasi-steady state approach simplifies the analysis.

From liquid properties point of view it is assumed that the flow is incompressible and fluid in each control volume has constant viscosity and velocity. This seems appropriate because the fluid velocity is too low to consider compressibility effects and fluid temperature does not change too much. Thus viscosity is taken to be the viscosity of saturated water at the tank temperature.

The amount of liquid inside the elements is an important parameter in dynamics of the fluid flow. The program keeps track of the amount of liquid or gas inside element. Fittings are assumed to have one percent(%1) of the liquid relative to the liquid inside the pipe sections. This not only simplifies the mass calculation but also eliminates the need to give calculation methods for volume of each new element introduced to the system.

Every element has a directional parameter which can be denoted as d_i. This is used to add the gravity effects to the system. During static and steady-state pressure distribution calculation a value of 1 adds the gravitational contribution whereas a -1 subtracts it.

To approximate frictional loss coefficient in turbulent flow in a smooth pipe

$$f = \frac{0.316}{Re^{1/4}} \qquad 5000 < Re < 100,000 \tag{4.1}$$

is used whereas in rough pipe we need Moody diagram. This diagram has an approximation given in Streeter's book[19] which is

$$f = \frac{1.325}{\left[\ln\left(\frac{\epsilon}{3.7D_i} + \frac{5.74}{Re^{0.9}}\right)\right]^2} \quad \text{where} \quad \begin{array}{c} 10^{-6} \leq \frac{\epsilon}{D_i} \leq 10^{-2} \\ 5000 \leq Re \leq 10^8 \end{array} \tag{4.2}$$

Although flow starts from rest, 95 % of time the fluid is in turbulent flow so these relations can be safely used. Other loss coefficients were taken from Crane's Technical paper [4].

4.2 Valve dynamics

For frictional calculations we have to evaluate friction factor K of the ball valve which is a function of valve angle θ. We choose the angle such that $\theta = \pi/2$ shows open valve and $\theta = 0$ shows closed valve. For a ball valve in fully open position, we have a certain value $K_{ballvalve}$ which can be taken from Crane's technical paper [4]. For closed position there is no flow so we can model this as K being infinity as long as there is flow inside the pipe. We have to fit a curve for getting values in between.

Sweeney [20] made experiments to determine effective closing time of the manually operated ball valves. They were made using a constant head tank to measure the flow rate as a function of valve position. Taking this data as the starting point we can fit a curve and assuming that there were only a valve and a tank we can write the Bernoulli equation to get

$$\frac{P_1 - P_2}{\rho} = \frac{KV^2}{2} \tag{4.3}$$

which leads to

$$K \propto \frac{1}{V^2} \tag{4.4}$$

The relation between θ and V can be given as a fourth order polynomial so we can propose

$$K = \frac{C_{calib}}{(-1.666 \times 10^{-3} + 0.1577\theta - 0.535\theta^2 + 1.689\theta^3 - 0.694\theta^4)^2} \tag{4.5}$$

The calibration constant C_{calib} depends on the pressure difference in the experiments and this data is not mentioned. Using the fact that loss coefficient should match the steady-state value at $\theta = \pi/2$ we can use a value of 1.475 which will give us a loss coefficient of 0.95 at steady-state. The shifting of a reciprocal type function brings several disadvantages. The most notable one is that pressure difference should only be imposed if there is flow. To ensure this, checking is necessary at first time step, where there is acceleration but no velocity. For relatively large time intervals, this may reveal results which predict no flow at small valve angles. A graphical representation of the model is given in figure 4-1.

4.3 Start dynamics and steady state

During the starting phase inertia forces dominate the system. At this point we can safely ignore the frictional effects as the velocity is very low. Acceleration pressure drop comes from the increasing kinetic energy of the system which can be given as

$$E_k = \frac{1}{2}\rho V^2 \tag{4.6}$$

As volumetric flow rate is the same throughout the system, smaller pipe sections have higher velocity and acceleration. So these pipes consume most of the available pressure head.

For a single section of pipe we can write the momentum equation as

$$\Delta P_I = \frac{\rho L}{A}\frac{dQ}{dt} \tag{4.7}$$

For different pipe sections we have to take a summation over these pressure drops which can be given by

$$\sum \Delta P_I = \sum \frac{L}{A}\frac{d\dot{m}}{dt} \tag{4.8}$$

and is a constant which is defined by the difference between the ambient temperature and the tank pressure. These facts are combined and inertance L/A is summed up

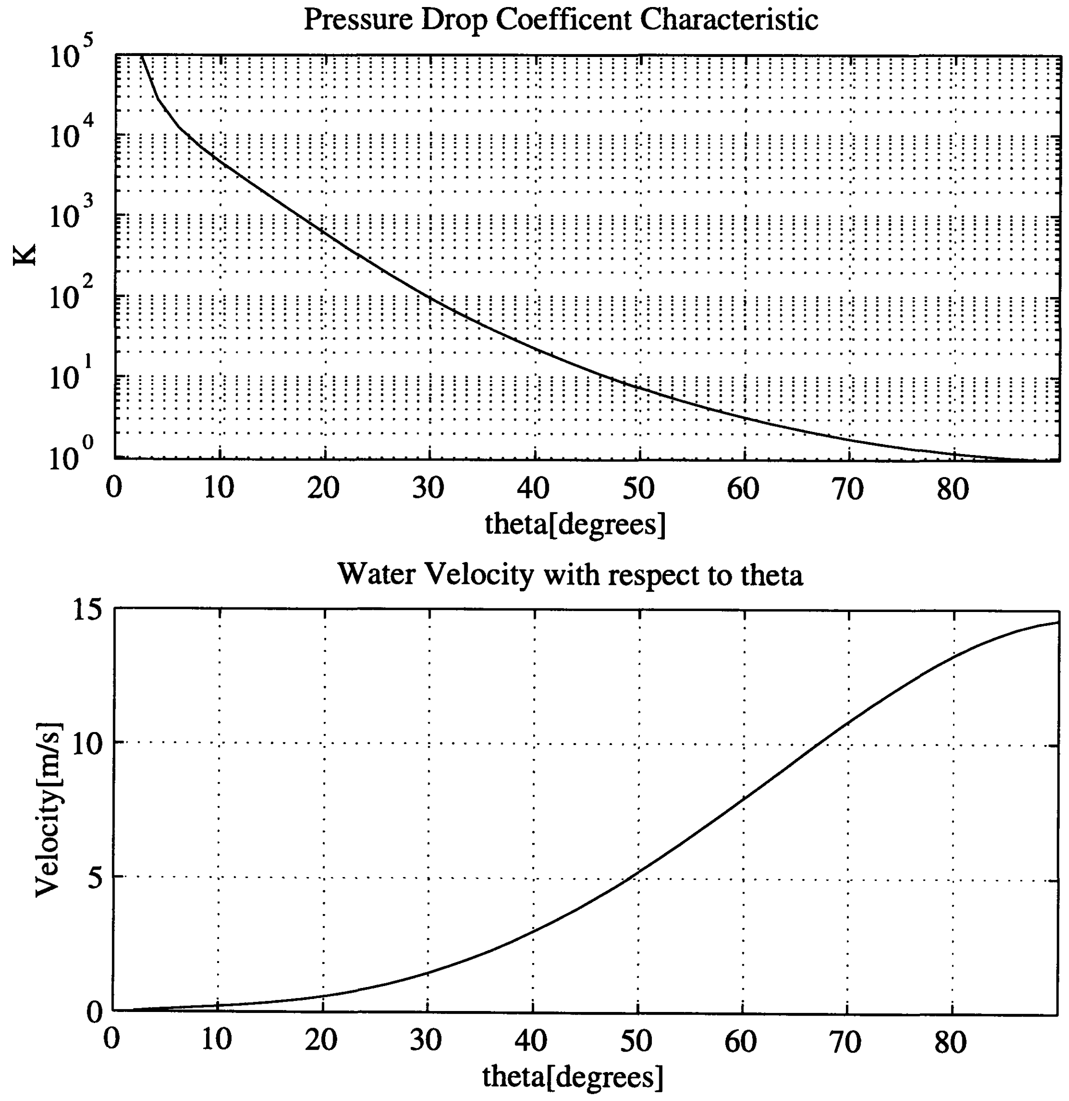

Figure 4-1: Valve Pressure Drop Coefficient model changing with θ

to find the initial change in mass flow rate. Then knowing change in mass flow rate, elemental pressure drops are calculated from

$$\Delta P_{I_i} = P_{I_{i-1}} - \frac{L_i}{A_i}\frac{d\dot{m}}{dt} - d_i \rho_i g \tag{4.9}$$

For steady state pressure distribution Bernoulli equation is used for each pressure element

$$\Delta P_{S_i} = P_{S_{i-1}} - \Delta P_{f_i} - g d_i \rho_i \tag{4.10}$$

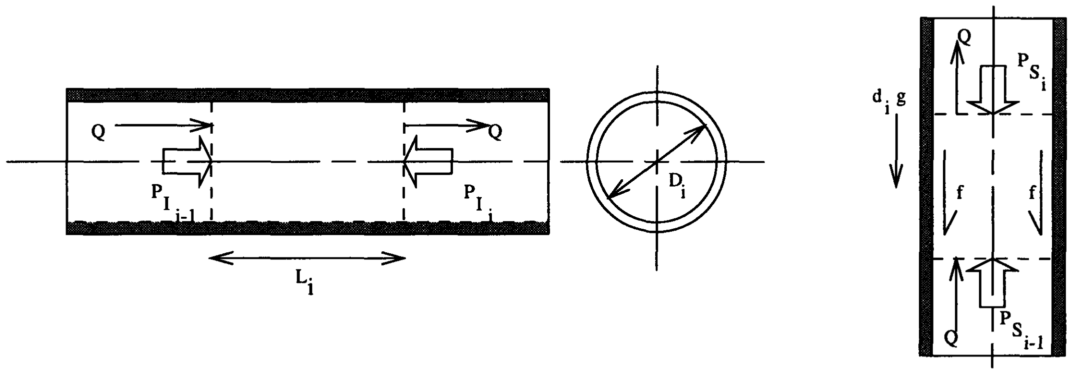

Figure 4-2: Schematic of for one phase flow model

where ΔP_{f_i} is the frictional pressure loss due to each element. Viscous friction is proportional to the square of velocity and can be written as

$$\Delta P_{f_i} = \frac{\rho g K V^2}{2} \tag{4.11}$$

The K parameter depends on the element geometry and for pipes has the form

$$K_{pipe} = f \frac{L}{D_i} \tag{4.12}$$

The friction factor f is given by the equations in the previous sections. For standard elements K factor is defined and can be found in Crane's Technical Paper [4].

4.4 Transient System calculations

Transient system analysis requires solving of simultaneous first order equations for single phase calculations and second order equations for two-phase flow.

4.4.1 One phase flow

The flow should be written in terms of one of the velocities in the system. For convenience the first element is taken as a pivot point. Thus all the velocities are converted to this scale. Cross sectional area ratio is defined as

$$A_{ratio_i} = \frac{A_i}{A_1} \tag{4.13}$$

The governing differential equation can now be written as

$$\frac{dV_1}{dt} A_{ratio_i} = -\frac{1}{\rho_i} \frac{P_i - P_{i-1}}{L_i} - \frac{K_i {A^2}_{ratio_i} {V_1}^2}{2L_i} - d_i g \tag{4.14}$$

in elemental form. This reduces the system so that it can be seen from first element point of view. Then summation over the elements gives

$$\left(\sum_{i=1}^{n} \frac{\rho_i L_i}{\rho_n L_n} A_{ratio_i}\right) \frac{dV_1}{dt} = -\frac{(P_n - P_o)}{\rho_n L_n} - \left(\sum_{i=1}^{n} \frac{\rho_i K_i}{\rho_n L_n} {A^2}_{ratio_i}\right) \frac{V_1^2}{2} - \left(\sum_{i=1}^{n} \frac{\rho_i L_i}{\rho_n L_n} d_i\right) g \tag{4.15}$$

This equation is solved at each time step by trapezoidal numerical method. Runge-Kutta was also tried but the increase in accuracy did not justify the complexity of a fourth order method. Non-erratic behavior of the system enabled us to use lower order methods.

Then again for each time step pressure distribution is calculated for each element i

$$P_i = P_{i-1} - \rho_i L_i \left(\frac{dV_1}{dt} A_{ratio_i} + \frac{K_i {A^2}_{ratio_i} V_1^2}{2L_i} d_i g \right) \tag{4.16}$$

4.4.2 Two-phase flow

In two-phase flow we are faced with a second order partial differential equation for each element which can be discritized first in space domain to reduce to ordinary differential equation. This can be written as

$$\frac{dV_i}{dt} = -\frac{P_i - P_{i-1}}{\rho \Delta x} - \phi_i^2 \frac{K_i V_i^2}{2\Delta x} - d_i g \tag{4.17}$$

This equation assumes that variation of velocity with respect to axial position is negligible and velocity is small. Validity of this assumption is questionable. Two-phase coefficient ϕ^2 is taken as in Wolf's thesis [22] and should be calculated for each element in time and space. It is important to note that his calculations were assuming that there was thermal equilibrium which is not correct. Similar equations as Equation 4.14 and 4.15 result from this analysis but space slicing is not only made to identify geometric boundaries like changing pipe sizes but also to identify propagation of two-phase flow. The calculations are very complex and need a lot of computing time.

4.4.3 Time constant approximation

The simulation uses lumped parameter model for a continuous system. From theoretical point of view, it reaches steady state at $t = \infty$. From practical point of view we need a settlement time which shows nearly steady state conditions. After the simulation is finished a first order system is fitted to the available data using time span and derivatives of the curve. The coefficients C_1 and C_2 are calculated from

$$\ln\left(\frac{\frac{dV_n}{dt}}{\frac{dV_n}{dt} + v_n C_2}\right) + C_2 t = 0 \tag{4.18}$$

and

$$C_1 = \frac{\frac{dV_n}{dt}}{C_2 e^{(C_2 t)}} \tag{4.19}$$

The results seem very reasonable in terms of expected time constants of the system. For example the simulation of Wolf's experimental setup gave $C_1 = 2.39$ and $C_2 = 9.47$. These correspond to a steady state velocity of 2.39 m/s and a settlement time of 0.53 seconds. Validity can be checked from figure 4-5.

4.5 Speed of sound calculations

Calculation of speed in a media is a strong function of density. Density of both liquid and vapor phase of water is used at saturation temperature. This seems a reasonable assumption because although non equilibrium conditions exist both phase are near saturation temperature. These variables are combined by

$$v = v_g x + (1 - x)v_f \tag{4.20}$$

to give the specific volume of the gas-liquid mixture. The speed of sound in liquid is calculated by

$$c_f = \sqrt{\frac{E_w}{\rho_w \left[\frac{E_w(D_o+D_i)}{E_p(D_o-D_i)}\right]}} \tag{4.21}$$

whereas speed of sound in gas is simply taken from ideal gas law

$$c_g = \sqrt{kRT} \tag{4.22}$$

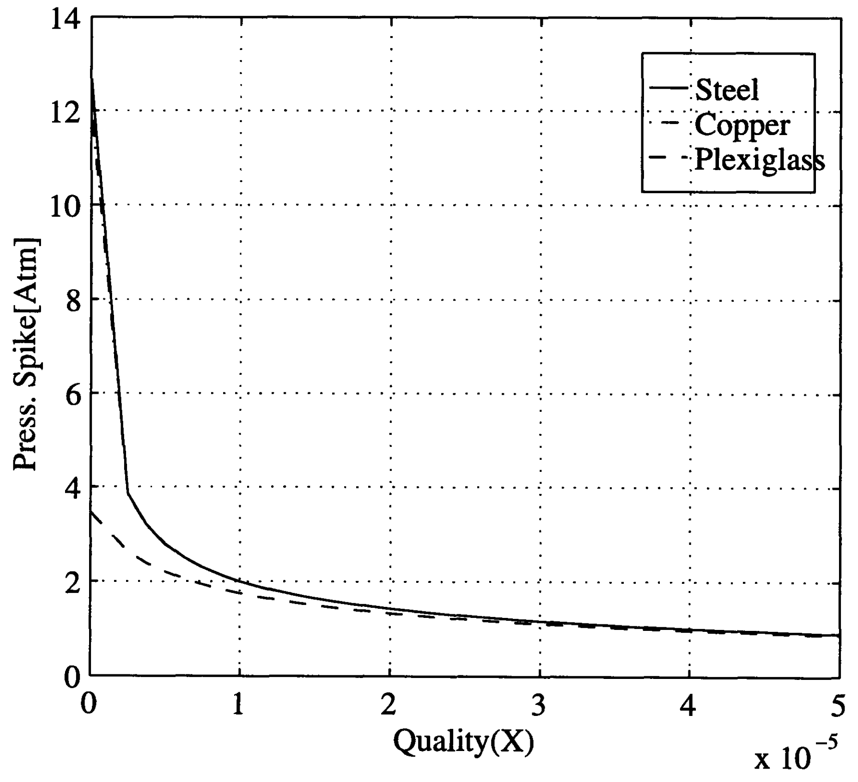

Figure 4-3: Pressure Spike originating from a unit velocity difference under different quality conditions

Finally to calculate the Speed of sound the following relationship given in Moody's book [10]:

$$\frac{c}{c_g} = \frac{\frac{v}{v_g}\sqrt{1 - \frac{v_f}{v_g}}}{\sqrt{\frac{v}{v_g}\left[1 - \left(\frac{v_f c_g}{v_g c_f}\right)^2\right] - \left[\frac{v_f}{v_g} - \left(\frac{v_f c_g}{v_g c_f}\right)^2\right]}} \tag{4.23}$$

This equation is given for compressive pressure waves. The actual speed of sound calculation for experimental setup under consideration should be based on decompressive speed of sound in the mixture. But this involves solving an equation with derivatives

$$c = v\sqrt{-g_o\left[\frac{dv_f}{dP} - \frac{d(s_f v_f/s_{fg})}{dP}\frac{d(v_{fg}/s_{fg})}{dP}s\right]^{-1}} \tag{4.24}$$

which is much more complicated than the one above. Also it needs mathematical model of entropy of water.

To see the effect of changing void fraction on the speed of sound, we can take an example. Take water at 125°C in a pipe of 50mm which has a wall thickness of 3 mm. Assuming that velocity in the pipe drops by 1 m/s, The resulting change in pressure spike is plotted in figure 4-3. Even one percent (%1) of vapor in the system will reduce the spike by a factor of 5 to 6.

Pipe elasticity seems to be an important parameter in evaluating the speed of sound. But from figure 4-3 it can be seen that for two phase flow this is not the case. The speed of sound for all pipe materials is almost the same when quality is larger than 2×10^{-5}.

4.6 Remarks on water properties

Water properties were evaluated by using a software library which was constructed for another project by the author. It was compiled from different sources including books, charts and tables. Some of the data were curve fitted by the author. The accuracy of the approximation changes from point to point in each used formulation but special care is taken so that maximum error is not greater than one percent. Formulations used were chosen to reflect an optimum point between accuracy and complexity. As they are used repeatedly, complex algorithms were avoided, to sacrifice accuracy. The library has its own range checking so that it does not reveal wrong answers.

4.7 Discussion of simulation

4.7.1 Simulations on existing setup

All the methods explained above were used to predict the system behavior of the experimental setup which Wolf used. The initial conditions which revealed the highest pressure spike was water at 125°C and hot water front at 11.5 meters from the tank, along the pipe. Water at this temperature has a saturation pressure of 231 kPa. From previous experience it was known that slug flow was not observed. This shows that void fraction was very low. Thus we assumed that two-phase flow did not effect the flow dynamics very much but changed the wave propagation speed.

Simulation results shown in figure 4-4 represent three different parts of the solution. Before the experiment started, there was pressure change along the pipe which was due to elevation changes in differential sections of the pipe.

When a valve is suddenly opened, pressure drops sharply to a curve much below its steady value which is shown by the curve labeled start. It is important to note that some part of the pipe is under saturation pressure. After system reaches steady-state, we can see the different pressure elements easily. For example the almost horizontal part in the graph shows large pipe section whereas the sharp drop shows the contraction.

Results of the transient system simulation show that system reaches steady state in about 0.68 seconds and the velocity in the smallest pipe section is about 2.4 m/s. The acceleration starts at 12 m/s^2 and drops to zero at steady state conditions. The valve opening was chosen to simulate human behavior. It was opened linearly in 0.1 seconds but for slower operations this might not be a realistic assumption. This time was relatively short with respect to the settlement time and it could be accepted as a slam opening from system dynamics point of view. Graphical representation of this results is given in figure 4-5.

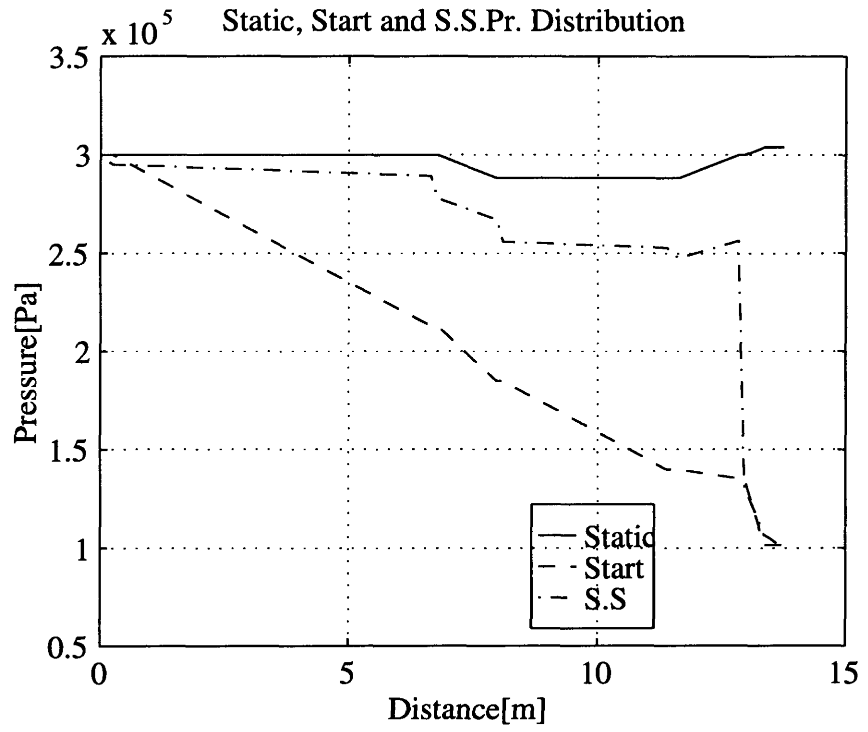

Figure 4-4: Simulation results from static, start dynamics and steady state analysis of Wolf's experimental setup

It should be noted that settlement time is a strong function of the frictional resistance in the system. There is an important relation between the pressure head available and the momentum of the fluid to be moved. If there is less frictional elements along the way, the steady-state velocity will be higher but to attain this velocity transient flow will be longer. Also the amount of fluid to be put into motion effects the pressure distribution inside the pipe.

The pressure distribution inside the pipe changes with time due to the transient flow and accelerating fluid. The acceleration of long pipe sections consume more of the pressure head at startup. Pressure in large pipe sections increases with time and reaches their steady value.

During transient flow, the contribution of the contraction increases and accounts for most of the pressure drop in steady state operation. The predicted pressure surface generated by the program can be seen in figure 4-6. Transient simulation starts after $t = 0$ so the horizontal line corresponding to $t = 0$ in this graph does not show the static pressure distribution. It is important to note that pressure drops drastically

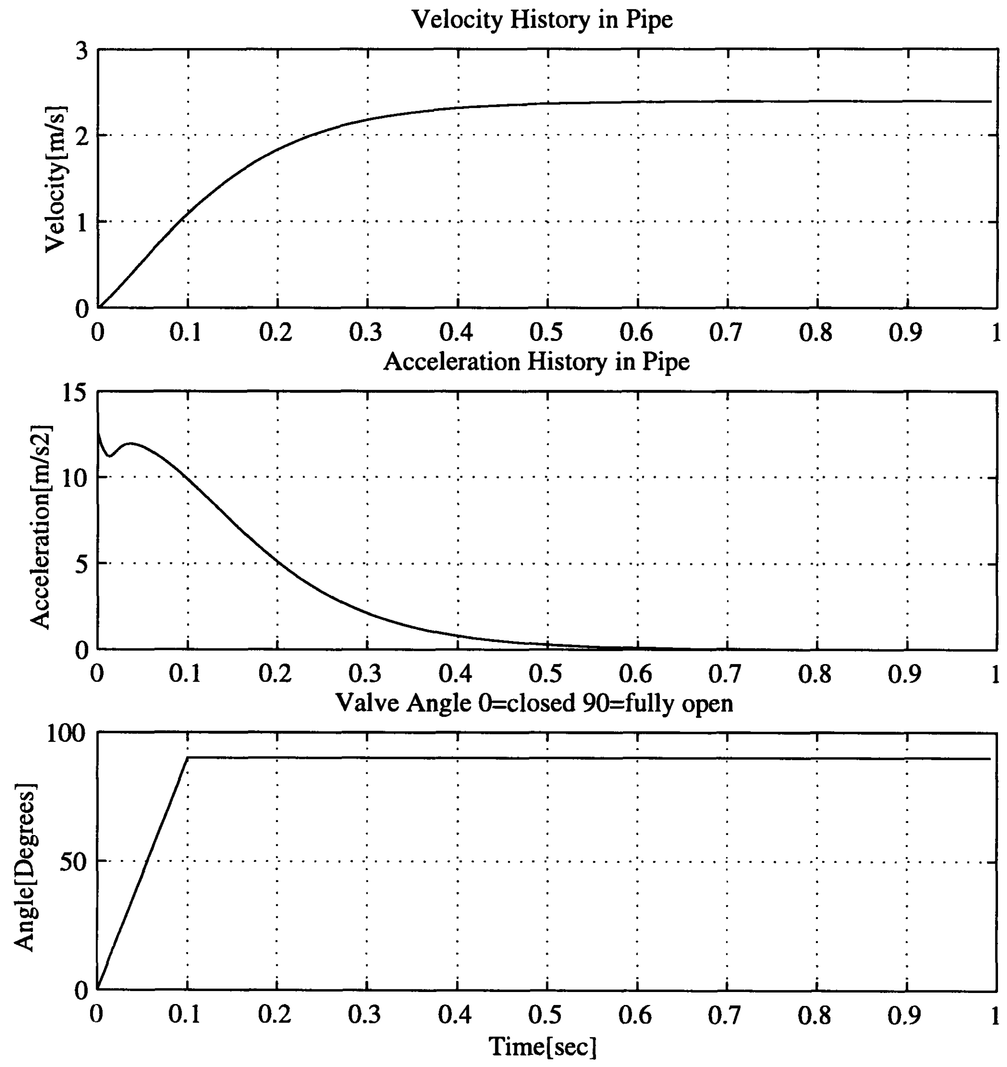

Figure 4-5: Simulation results from transient analysis of Wolf's experimental setup

during the initial valve opening. The wave speed with ten percent (10 %) quality is around 160 m/s. This shows that it takes 0.2 seconds at most to reach the tank and come back to the valve. We can safely say that flow can easily follow the change in pressure and there is no information gap in the pipe.

Although figure 4-6 contains lots of information it is not easily understandable. We can slice the figure at different times during the simulation. Figure 4-7 shows such slices taken out of this figure. The horizontal line in each slice represents the saturation pressure of hot water inside pipe. When system is at rest, pressure inside the pipe is above this line so system is in one phase. After 0.1 seconds of start we have a drastic pressure drop along the pipe. The pressure profile intersects the saturation

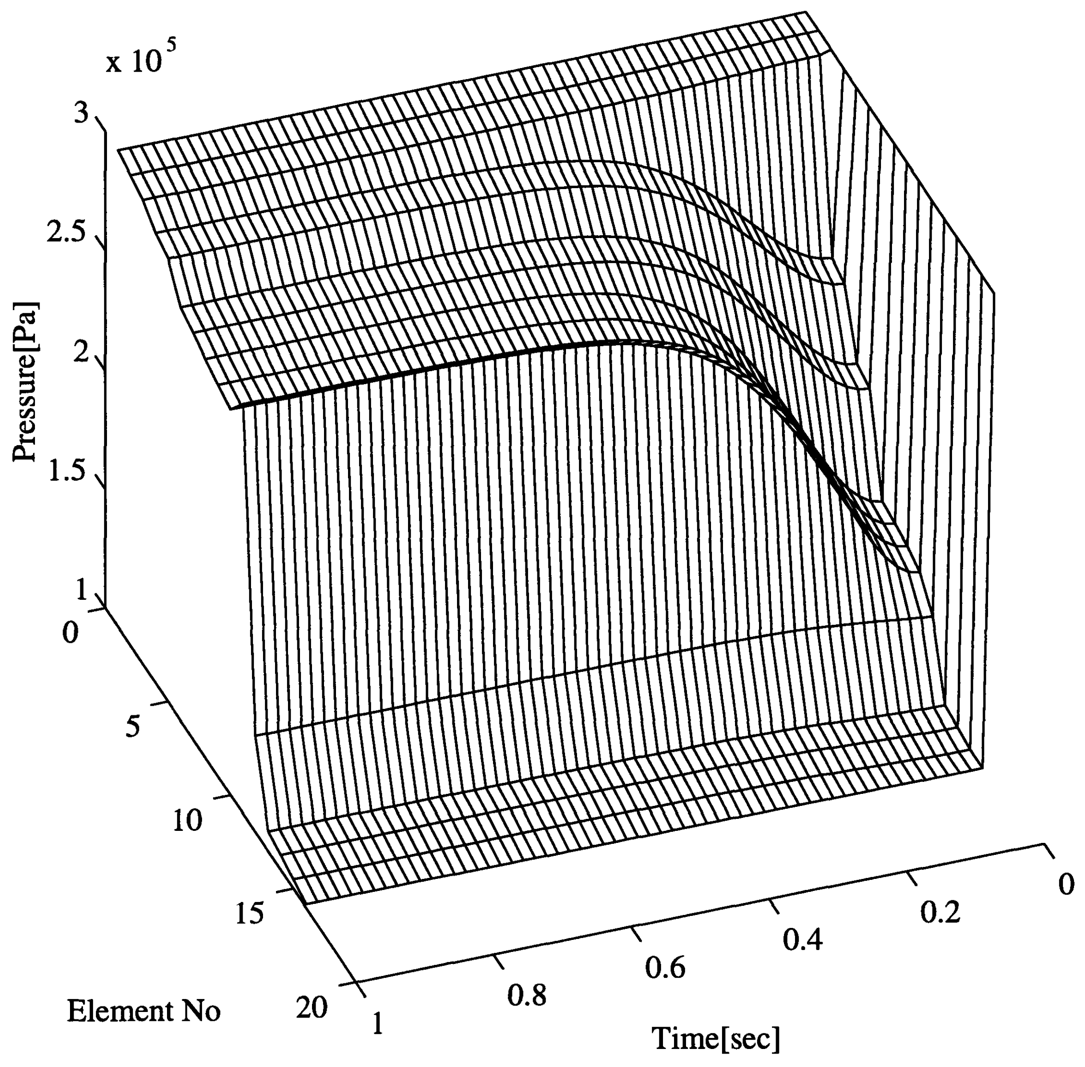

Figure 4-6: Pressure surface formed in transient analysis of Wolf's experimental setup

line. This is the point where two phase flow starts. At 0.2 seconds intersection point moves to the end of the pipe and the profile builds up as pressure rises in the pipe. After 1 second has passed the system is almost in steady state and intersection point moves to the small pipe section which is indicated by the sharp pressure drop. From the first and last figures we can say that two states are easily predictable. Unfortunately moving from the first state to the last has complicated physics and was not understood or predicted in the experiments.

The intersection point of saturated pressure line and the pressure profile in the pipe shown in figure 4-7 is only a part of the picture because this point moves with time. Hot water moves as it accelerates along the pipe and this acceleration changes considerably at the contraction. Plotting these two curves on the same graph we have a very informative plot given in figure 4-8.

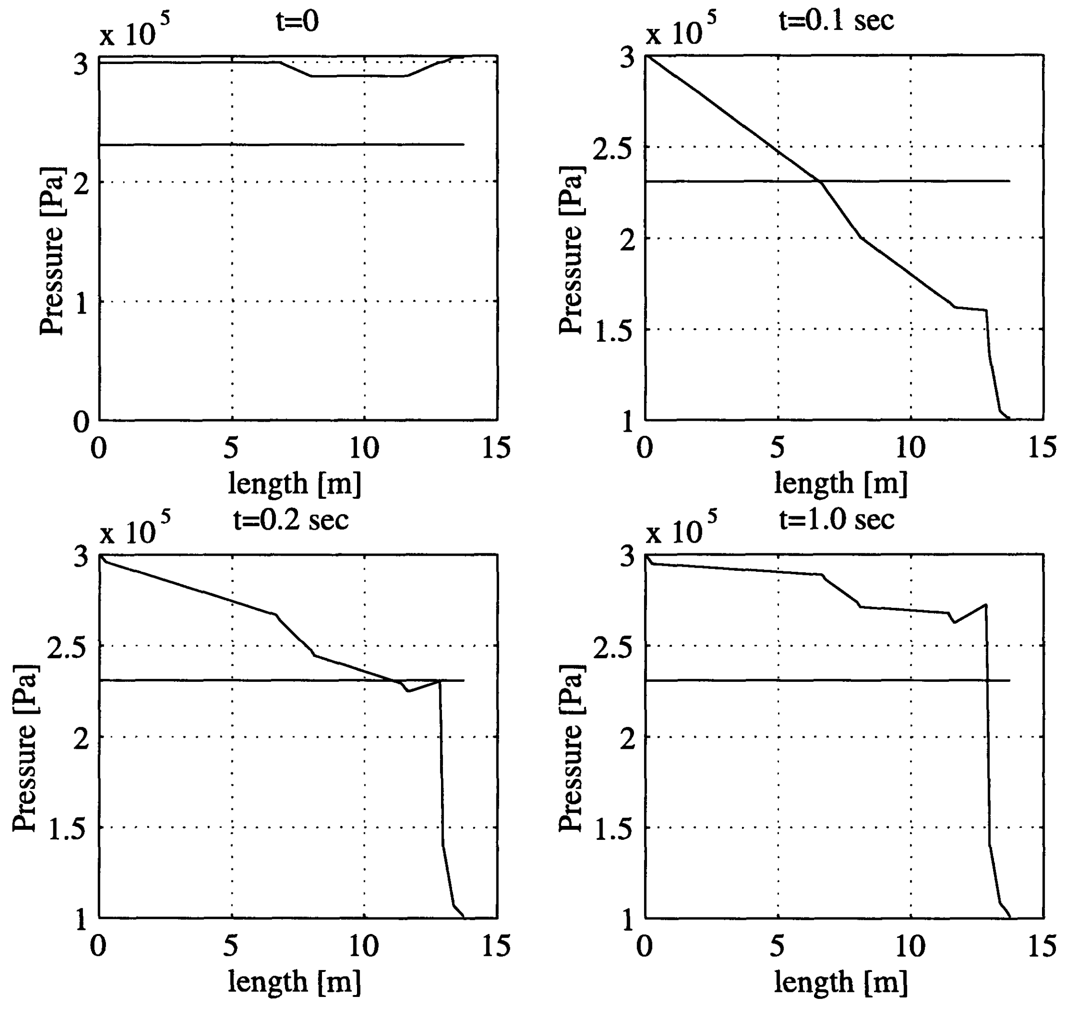

Figure 4-7: Pressure profile inside the pipe at different time slices

This plot shows that at startup the flashing point is initially at 5.6 meters from the tank which is well in the hot water region. Graph does not start from $t = 0$ because of time discritization. The gap shows the time slice for the simulation. As pressure builds up, flashing point quickly moves to its steady state position which is in the small pipe section. This is the design point but it does not take the starting transient into consideration. But during this starting stage hot water in the pipe flashes. Bubbles form in about 1 millisecond. So at just after startup we have a very low quality

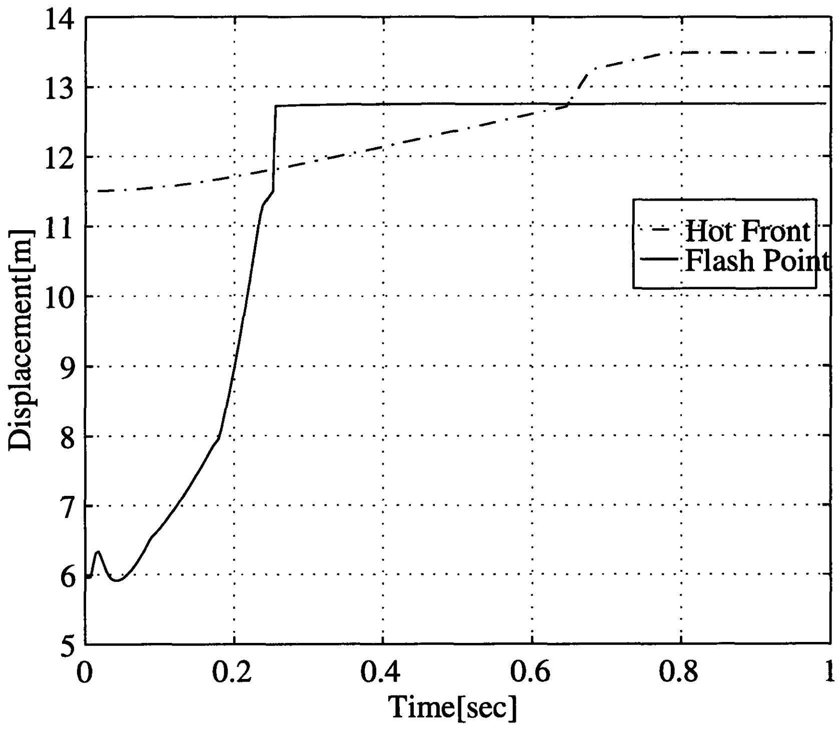

Figure 4-8: Flashing Point vs. Hot Water front in transient analysis of Wolf's experimental setup

two-phase flow in the pipe. The air in the water also collects in these bubbles as their pressure is lower than the surrounding liquid. Then in about 0.25 seconds pressure rises and in steady state conditions bubbles should collapse bringing the fluid back into single phase. Unfortunately bubble collapse is a slow process because the air trapped inside the bubbles has to diffuse back into the liquid but the pressure differential is so small that bubbles persist. After 0.63 seconds the hot water front reaches flashing boundary once more and the expected two-phase flow starts revealing a pressure spike. At this point there are still uncollapsed bubbles inside the liquid which has a very important effect in reducing the magnitude of the pressure spike. It is very important the note that system has different time constants. Although from bulk fluid motion point of view flow reaches steady-state in 0.68 seconds, from thermodynamics point of view this time is not enough to accommodate bubble formation and collapse. Hot water flashes in small pipe before the velocity profile has settled down. Wave propagation is still very fast with respect to both of these phenomena and can carry information in much small time intervals. For example the flashing point moves 7.2 meters in 0.24 seconds along the pipe.

4.7.2 Simulations for hypothetical setups

The mathematical model was not only used to analyze flow in Wolf's experiment but also used as a tool in designing new setups that can demonstrate passive water hammer. Different setups were simulated to understand the importance of the various elements in the setup. Due to the nature of piping systems under consideration it is impossible to eliminate flashing in starting transients by design only. However, if the valve controlling the flow is opened slowly then the starting transients do not show two phase flow. As a result when hot water clears from the system, it flashes at the design point and hopefully gives a spike which can be predicted from the steady-state design conditions. From different simulations the optimum bad setup was chosen which is shown in figure 4-9. This is a direct descendent of Wolf's apparatus.

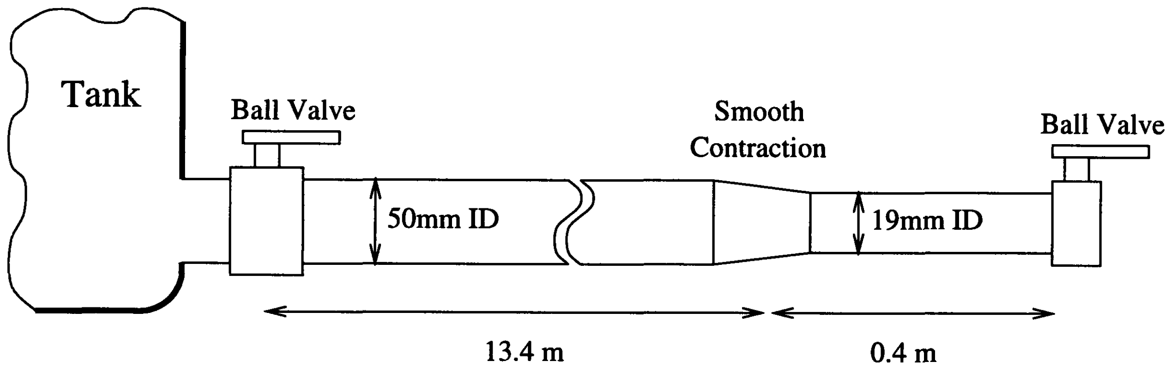

Figure 4-9: Setup proposed for best spike capturing

The major feature of the system is its simplicity. It does not have any vertical pipe sections, bends or tees. There are only two valves. The first one is used to separate hot water from rest of the pipe. The second one is the flow control valve which is used to eliminate the momentary two phase transition which appears during starting transients. The smooth contraction is the place of high pressure drop and is intended for giving rise to pressure spike. The length of pipe is chosen such that

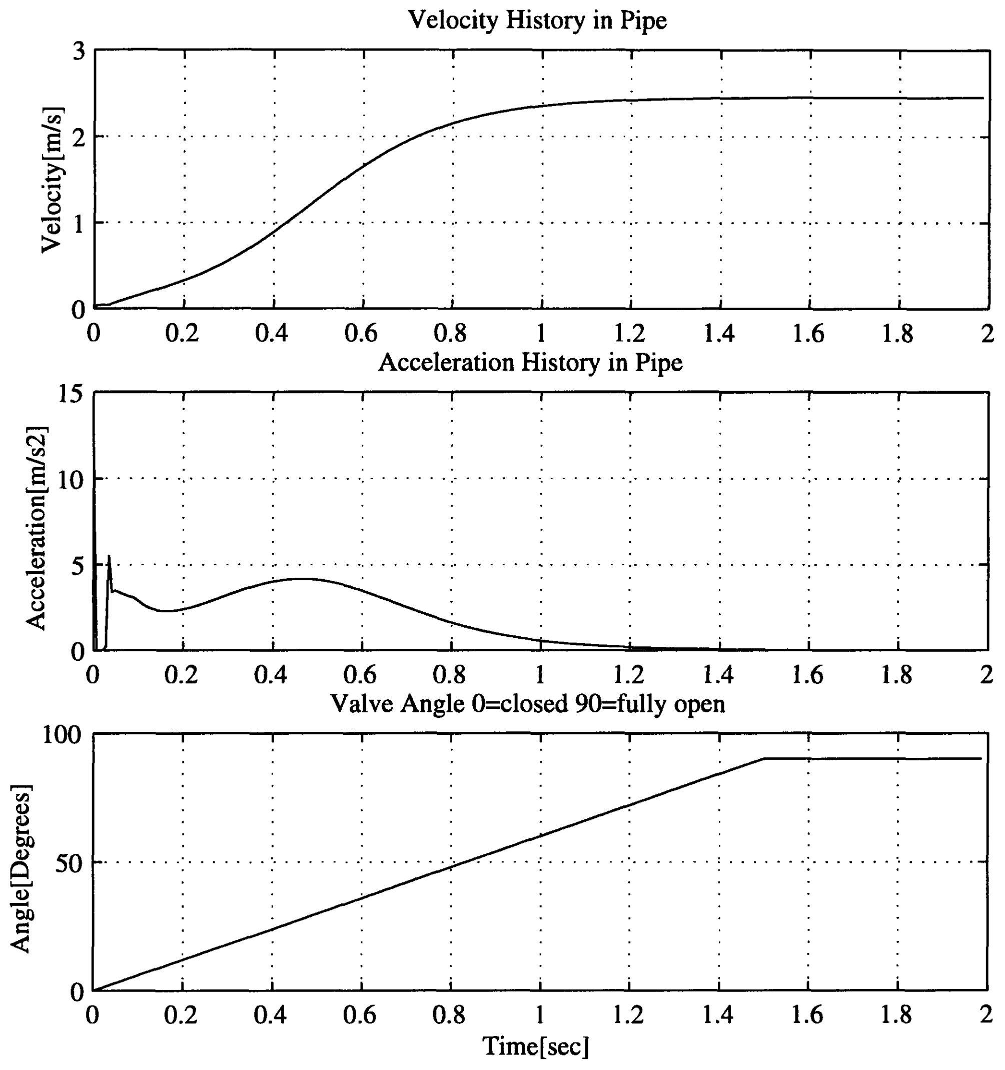

Figure 4-10: Simulation results for proposed setup

pressure waves have enough time to travel along the pipe. The pipe is not insulated because it takes about 7 seconds to hot water to clear out and this time is very short to get enough heat transfer from the water inside the pipe. The experiment will start by heating the water in the tank to just below its boiling point under atmospheric conditions. Then both valves will be opened so the pipe will be filled with water and then both valves will be closed. This ensures that there is no air trapped inside the pipe and amount of air dissolved in the water is minimized. The next step is to heat the water in the tank somewhere beyond 125°C which is the temperature in Wolf's experiment which yielded the highest pressure spike. Pressure inside the tank is kept at 300kPa, the same as in Wolf's experiments. When everything is ready, first the valve near to the tank is opened. This creates the cold/hot water interface. Then the second valve is slowly opened. We propose that it is opened linearly in 1.5 seconds. This gives enough time for water to accelerate without phase change. The velocity in the pipe will reach steady state in about 1.5 seconds. This particular valve opening

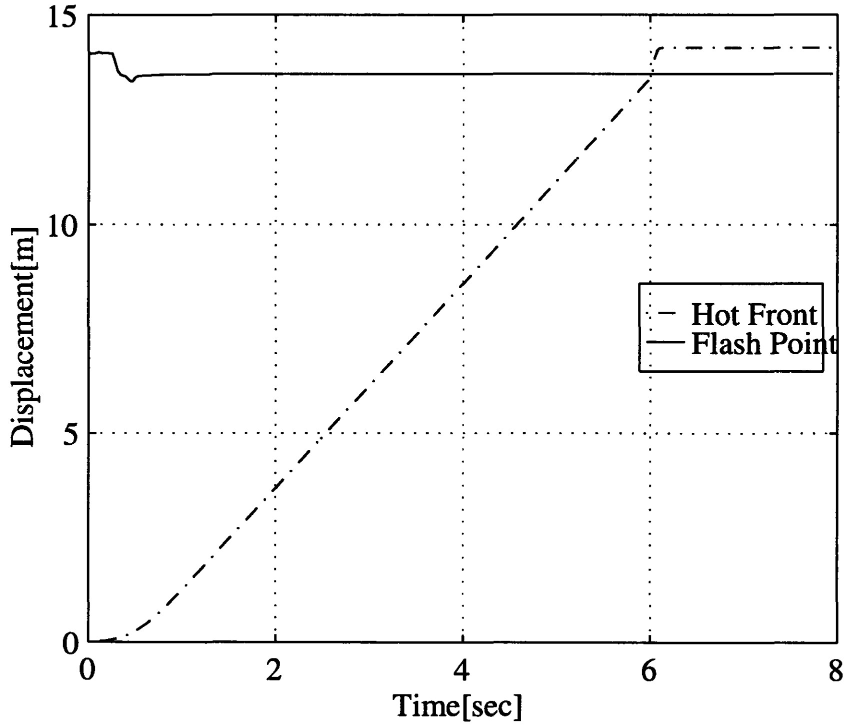

Figure 4-11: Flashing point intersecting hot water front for proposed setup

has several consequences. The first one is that at the very beginning of flow the system reaches equilibrium for a short period of time. This is probably caused by the nonlinear behavior of the valve loss coefficient. The second one is that acceleration is reduced to an average of 4 m/s^2 instead of 12 which is the basic reason for two-phase flow transition. Finally the flashing point in the pipe reaches an equilibrium point in almost 0.5 seconds and does not move. As a result hot water reaches this point when system is in steady-state.

Another use of this model is to explore the behavior in real applications. One of the feedwater systems mentioned earlier was also modeled to calculate the optimum valve opening time by the operator. Although the simulations showed that transient

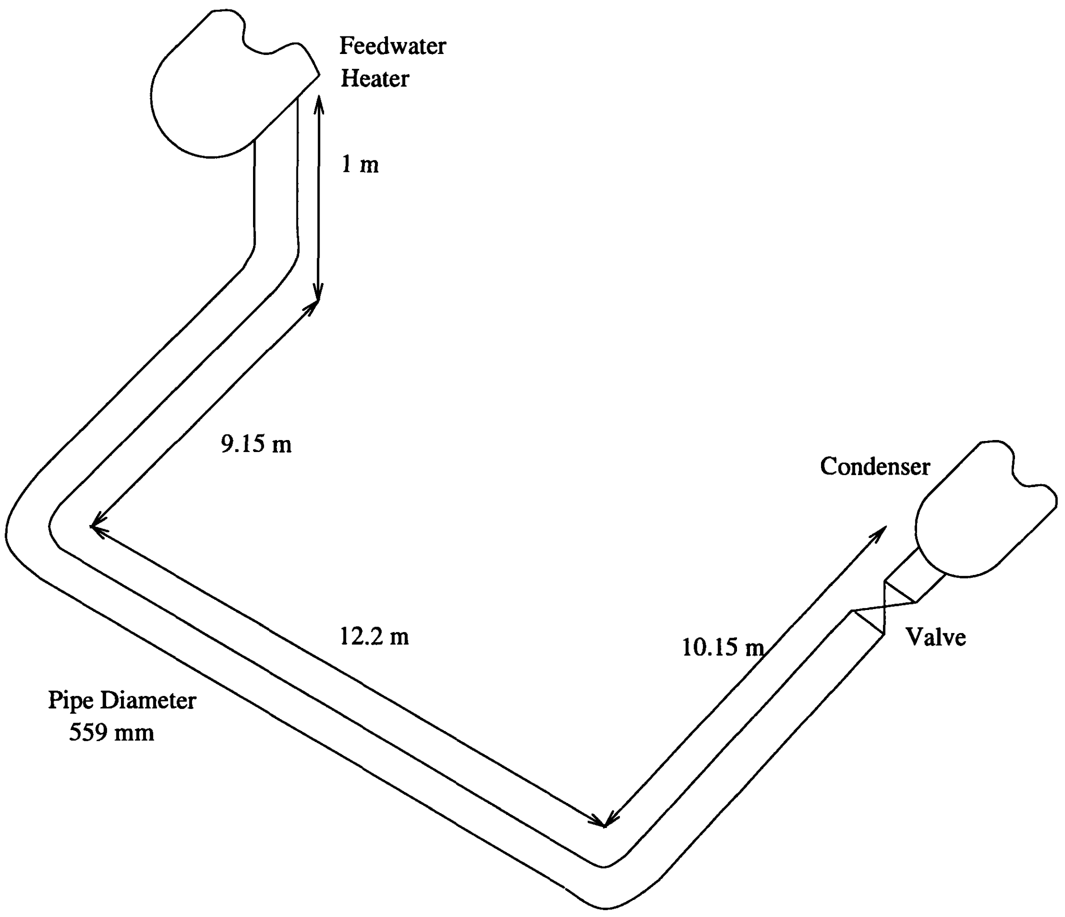

Figure 4-12: Feedwater drain line of a nuclear power plant

phase change can be avoided by opening the valve even in 5 seconds, this would be very hard if not impossible for a valve of 50cm (2 inch) diameter. On the other hand the velocity should be kept low until the hot/cold water interface clears out from the system. The solution is to crack open the valve and let the cold water clear out. After passing a sudden stoppage with small velocity, the valve can be fully open to reach steady state value. By a conservative approach we propose that valve is first opened to 20 degrees in 27 seconds. This time would be enough for cold water to leave the system. Then the valve can be opened fully in 8 seconds. The system will reach steady state in less than 40 seconds. This will limit the maximum acceleration to below $6m/s^2$ which will totally eliminate the transition to two phase flow during the starting transients. As a result of this slow opening the flow comes to an early

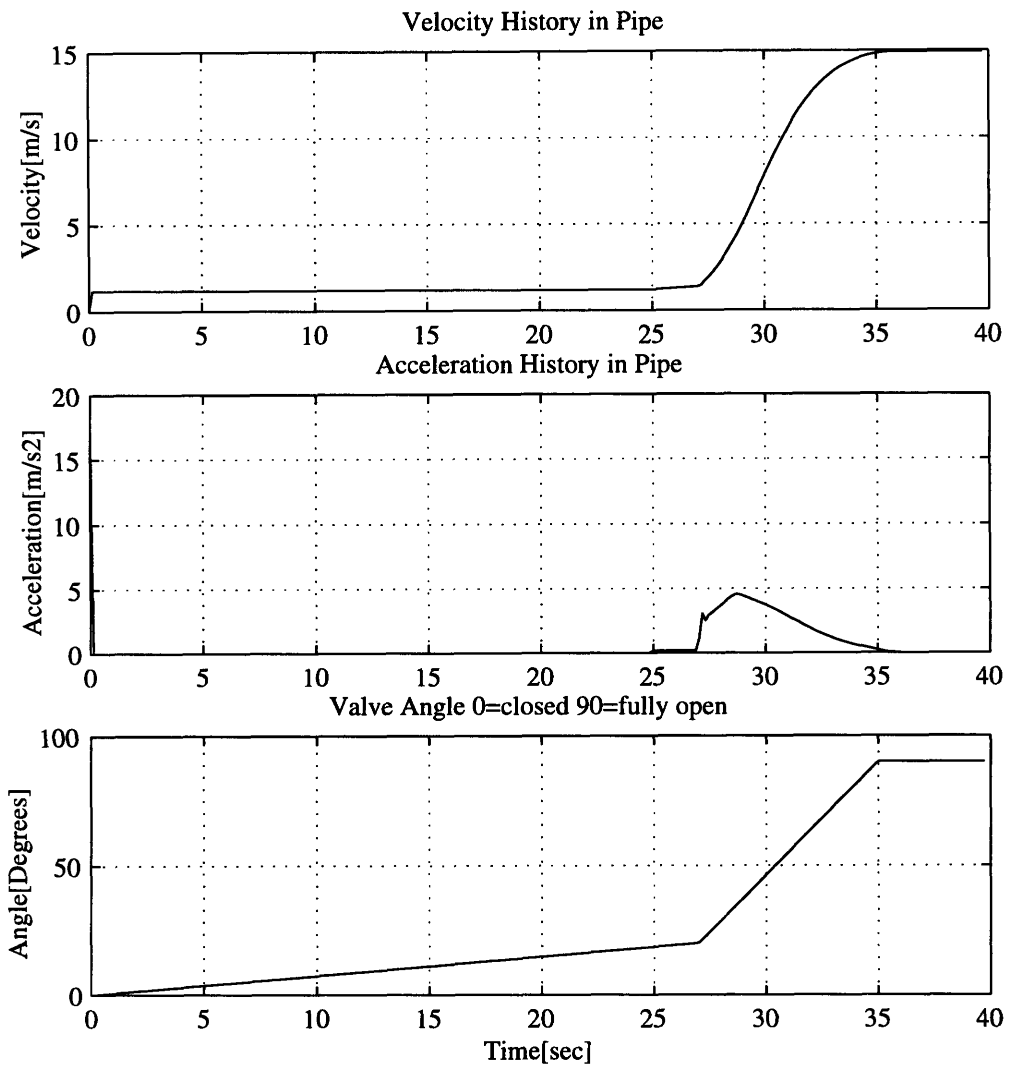

Figure 4-13: Simulation results for nuclear power plant feedwater heater

steady state velocity in the first few seconds. The acceleration graph (Figure 4-13) shows small oscillations after this pseudo steady state which should be discarded. It originates from the fact that solution of the differential equation changes very fast with respect to the numerical solver. The system will be settled in about 38 seconds giving a steady state value of 15 m/s in a pipe section of 559 mm ID. This is abnormally high but it originates from the fact that the given pressure difference is very high. (Wolf's setup was 300 kPa whereas this is 660kPa)

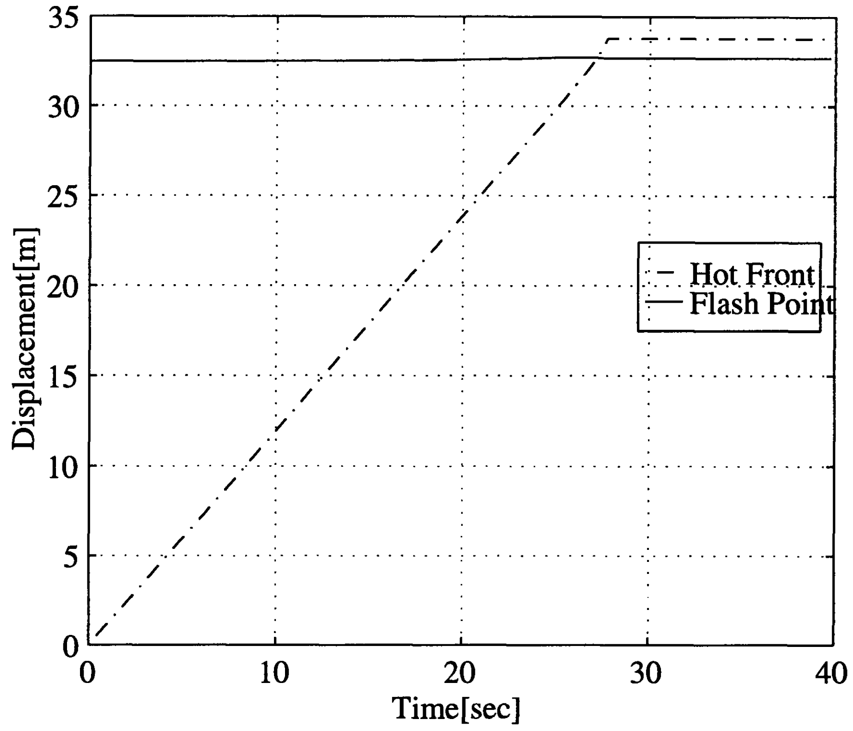

Figure 4-14: Flashing point vs. hot water front for power plant feedwater heater

4.7.3 Controlling the valve

Simulations show that to avoid two-phase flow in starting transients and passive water hammer, time needed to open the valve is a key factor. However determination of this parameter is difficult because of the complex physics involved. Calculation of exact timing is not important as the operation of a plant has much bigger time scales.

From a conservative point of view, the valve should be crack opened to start the system. This would enable the cold water in the pipe to slowly advance and leave the sytem. As the velocity is low, the observed pressure spike resulting from the hot/cold water interface would be minimum. It is important to note that, pipe supports should still be designed to accomadate this shock. After this initial shock, valve can be fully opened and the system can come to steady state without any further problem. If further insight is needed, a rough estimate for cracking open the valve can be calculated by a computer model simulating the transients.

Chapter 5

Conclusion

Passive water hammer is an important problem in piping systems. It occurs because flashing at the flow control valve at the end of the pipe increases the pressure differential and reduces the velocity. It generally occurs during transients so the use of simulation is necessary to analyze it. Simulation results have demystified some of the important aspects of the passive water hammer. There are several reasons why Wolf's experiments were not successful to identify reasons for the water hammer.

First of all there were different time constants associated with different parts of the problem. The fluid part had a very large settling time about 2 seconds due to large inertance of water involved. Thermodynamics of flashing had a settling time that was almost one tenth of the fluid part. Wave dynamics had a time constant which was even smaller than thermodynamic side. The experimental setup was designed to resolve some of these but not all.

Bubble formation and condensation occur at different rates. Vapor formation takes several milliseconds in creating a cavity. Air trapped in water diffuses into these cavities. This makes a bubbly air-water mixture. When pressure rises above saturation pressure, tending to collapse the bubble, this air has to diffuse back to water. But mass and heat diffusion processes proceed at greatly differing rates. It takes more time for bubbles to collapse as the diffusion of mass(air) takes much larger than the diffusion of heat. So once bubbles are formed they do not have enough time to collapse redissolve before they reach the end of the pipe.

When experimental setup did not show bubbles it was concluded that flow was single phase. However, due to the starting transients, flashing occurred upstream and there were apparently small unobservable bubbles which turned the single phase flow into a two-phase flow. This was the reason for pressure spikes being small. The small bubbles have so little volume that they are scarcely visible but have a substantial effect on the velocity of sound.

For constructing a new experiment some points may be helpful to avoid potential problems. Although longer pipe sections were chosen to have higher wave traveling time, pipes having large numbers of fittings have a higher transient response time. To decrease transient response time less fittings should be used. Using a vertical pipe section was mandatory because of space limitations but this made it necessary to use more fittings. So vertical pipe sections should be eliminated. To avoid two-phase flow at start-up valve should be operated slowly with respect to system acceleration parameters. At the same time it should be fast enough so that when hot/cold water interface reaches the valve the flow has considerable velocity. This is necessary to ensure that passive water hammer can be observed.

For a real application like a feedwater system of a power plant trying to avoid passive water hammer, valve operation is very important. The time for cold water to clear the system is the dominating factor. The valve should be cracked opened so little that any stoppage occurring before the cold water that is flushed out would be tolerable to the piping supports. After the cold water has been flushed out the valve can be fully opened to allow the system to come to steady state.

It is important to note that this research is not finished at this stage. There are areas remaining to be understood. New experiments should be carried out to identify the physics behind which will yield mathematical models. This will enable us to calculate and predict the effects of passive water hammer events.

Appendix A

Computer Programs

A.1 Pressure element declerations `presdef.h`

```
/******************* presdef.h *****************/
/********* Definitions for Constants ***********/

#define UNKNOWN −1
#define RPIPE 1
#define SPIPE 2
#define ELBOW 11
#define BALL_VALVE 21
#define VAR_BALL_VALVE 22
#define CONTRACTION 31
#define ENLARGEMENT 32
#define TANKEXIT 41
#define PIPE_EXIT 42
#define TEE_BRANCH 43
#define TEE_THROUGH 44

/***** Definitions for Setup structure *********/

struct PressureElem
{
 char typename[50];
 int type,direction;
 float p1,p2,p3;
 float rho,mu,length;
};
```

A.2 Pressure element library header file `preselem.h`

```
/************** preselem.h ********************/

#define MAXELEMENT 25

/***** Function Declerations *****************/

extern void DisplayElements(struct PressureElem *p,int n);
extern float DPRpipe(float V,struct PressureElem p);
extern float DPSpipe(float V,struct PressureElem p);
extern float DPElbow(float V,struct PressureElem p);
extern float DPContraction(float V,struct PressureElem p);
extern float DPTeeBranch(float V,struct PressureElem p);
extern float DPBallValve(float V,struct PressureElem p);
extern float DPVarBallValve(float V,struct PressureElem p,float angle);
extern float DPTankExit(float V,struct PressureElem p);
extern float DPEnlargement(float V,struct PressureElem p);
extern float DPPipeExit(float V,struct PressureElem p);
extern float Area(float d);
```

A.3 Pressure element function library preselem.c

```
/************* preselem.c ***********************/

#include <stdio.h>
#include <string.h>
#include "presdef.h"
#include "transdef.h"
#include "frictlib.h"
#include "frict.h"

/*************Function Definitions **************/

void DisplayElements(struct PressureElem *p,int n)
{
 int i;
 char str1[20],str2[20];
 struct PressureElem pe;
  for (i=0;i<n;i++)
   {
    pe=*(p+i);
    switch (pe.direction)
     {
      case -1:strcpy(str1,"Down       ");break;
      case  0:strcpy(str1,"Horizontal");break;
      case  1:strcpy(str1,"Up         ");break;
     }
    switch (pe.type)
     {
      case RPIPE:
         printf(" %d) %s :  %s L=%2.2fm D=%2.2fmm\n",i+1,pe.typename,
             str1,pe.p1,pe.p2*1000);break;
      case SPIPE:
         printf(" %d) %s :  %s L=%2.2fm D=%2.2fmm\n",i+1,pe.typename,
             str1,pe.p1,pe.p2*1000);break;
      case UNKNOWN: printf(" %d) Unknown Type:  %s\n",i+1,pe.typename);break;
    case CONTRACTION: printf(" %d) %s :  From D1=%2.2fmm to D2=%2.2fmm\n",i+1,
                    pe.typename,pe.p1*1000,pe.p2*1000);break;
    case ENLARGEMENT: printf(" %d) %s :  From D1=%2.2fmm to D2=%2.2fmm\n",i+1,
                    pe.typename,pe.p1*1000,pe.p2*1000);break;
    case ELBOW:printf(" %d) %s :  Radius=%2.2fmm D=%2.2fmm\n",i+1,pe.typename,
            pe.p1*1000,pe.p2*1000);break;
    case TEE_BRANCH:printf(" %d) %s :  D=%2.2fmm\n",i+1,pe.typename,
            pe.p2*1000);break;
    case BALL_VALVE:printf(" %d) %s :  D=%2.2fmm\n",i+1,pe.typename,
            pe.p2*1000);break;
    case VAR_BALL_VALVE:printf(" %d) %s :  D=%2.2fmm\n",i+1,pe.typename,
            pe.p2*1000);break;
    default :printf(" %d) %s :  %2.2f %2.2f\n",i+1,pe.typename,
           pe.p1,pe.p2);break;
    }
  }

}
/***************************************************/

float DPRpipe(float V,struct PressureElem p)
{
 float R,f;
 R=Re(V,p.p2,p.rho,p.mu);
 f=fc(R,p.p2);
 return(f*p.p1/p.p2*p.rho*V*V/2.0);
}

/***************************************************/

float DPSpipe(float V,struct PressureElem p)
{
 float R,f;
 R=Re(V,p.p2,p.rho,p.mu);
 f=fs(R);
 return(f*p.p1/p.p2*p.rho*V*V/2.0);
```

```
}

/***************************************************/

float DPElbow(float V,struct PressureElem p)
{
 return(KStandardElbow*V*V*p.rho/2.0);
}

/***************************************************/

float DPContraction(float V,struct PressureElem p)
{
 /*This includes both the contraction loss and
   Momentum Change Loss
 */
 float Aratio;
 Aratio=p.p2/p.p1;
 return((0.306*(1-Aratio)+(1-Aratio*Aratio))*V*V/2.0*p.rho);
}

/***************************************************/

float DPTeeBranch(float V,struct PressureElem p)
{
 return(KStandardTee*V*V*p.rho/2.0);
}

/**************************************************/

float DPBallValve(float V,struct PressureElem p)
{
 return(KBallvalve*V*V*p.rho/2.0);
}

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

```
float DPTankExit(float V,struct PressureElem p)
{
 return(KTankExit*V*V*p.rho/2.0);
}

/**************************************************/

float DPEnlargement(float V,struct PressureElem p)
{
 float K;
 K=KSuddenEnlargement(p.p1/p.p2);
 return(K*V*V*p.rho/2.0);
}

/**************************************************/

float DPPipeExit(float V,struct PressureElem p)
{
 return(V*V*p.rho/2.0);
}

/**************************************************/

float DPVarBallValve(float V,struct PressureElem p,float angle)
{
 return(Kballtheta(angle)*p.rho*V*V/2.0);
}

/**************************************************/

float Area(float d)
{
 return(d*d*PI/4.0);
}
/************** End of preselem.c ****************/
```

A.4 Pressure profile calculation program pres2.c

```
/***************** pres2.c *************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "frictlib.h"
#include "presdef.h"
#include "transdef.h"
#include "output.h"
#include "input.h"
#include "frict.h"
#include "preselem.h"

struct PressureElem pe[MAXELEMENT];

/******************** Main *************************/

void main(int argc,char **argv)
{
 int i,j,n,ret,Terminate=0;
 float Pi[100],Pa[100],Ps[100],Length[100],Mass[100];
 float Phigh=660E3;
 float Plow=100E3;
 float Tlength=0,Ticklength,Tmass=0,dp=0,Vel=0,Vel2,inertance=0,ma,
      increment=1,massflow;

 Pi[0]=Phigh;
 Pa[0]=Phigh;
 Ps[0]=Phigh;
 Length[0]=0;
 printf("\nPressure Profiler for Pipe Systems\n");
 printf("By Murat Erenturk October 1995\n");
 if (argc<2)
 {
  printf("Usage:\npres2 <filename>\n");
  exit(0);
 }

/*************** Reading Data File ****************/

printf("Reading Data from %s ...",argv[1]);
ret=ReadSetup(argv[1],pe,&n);
if (ret) exit(-1);
printf("Done\n");

/******* Initialization  of Variables *************/

DisplayElements(pe,n);
for (i=0;i<n;i++)
 {
  if (pe[i].type==RPIPE || pe[i].type==SPIPE)
   {
    Tlength+=pe[i].p1;
    Mass[i]=Area(pe[i].p2)*pe[i].p1*pe[i].rho;
   }
   else Mass[i]=0;
   pe[i].rho=1000;pe[i].mu=1020E-6;
/*All elements are set to have same density and viscosity*/
 }
for (i=0;i<=n;i++)
 {
  if (pe[i].type==RPIPE || pe[i].type==SPIPE)
    Length[i+1]=Length[i]+pe[i].p1;
  else Length[i+1]=Length[i]+Tlength*0.01;
  Tmass+=Mass[i];
 }
printf(" Total length of Pipe :  %2.2f m\n",Tlength);
printf(" Total Mass of Water  :  %2.3f kg\n",Tmass);
```

```
/***********Static pressure Distribution*********/

 printf("Calculating Initial Press.  Dist.  ...");
 for (i=0;i<n;i++)
   {
    if ((pe[i].type==RPIPE || pe[i].type==SPIPE) && pe[i].direction!=0)
         Pi[i+1]=Pi[i]+pe[i].p1*G*pe[i].rho*-pe[i].direction;
    else Pi[i+1]=Pi[i];
   }
 printf("Done\n");

/*********** Acceleration pressure Drop **********/

 printf("Calculating Accel.  Press.  Drop ...\n");
 for (i=0;i<n;i++)
   {
    if (pe[i].type==RPIPE || pe[i].type==SPIPE)
    inertance+=pe[i].p1/Area(pe[i].p2);
    dp+=-pe[i].direction*pe[i].rho*G*pe[i].p1;
   }
   dp+=Phigh-Plow;
   ma=dp/inertance;
  printf(" Initial  Mass acc.  rate :  %2.3f kg/s2\n",ma);
  for (i=0;i<n;i++)
    {
     if (pe[i].type==RPIPE || pe[i].type==SPIPE)
       {
        if (i<(n-1)) j=i+1;
        else j=i;
        dp=pe[i].p1/Area(pe[i].p2)*ma+pe[i].direction*pe[i].rho*G*pe[i].p1;
        Pa[i+1]=Pa[i]-dp;
       }
      else Pa[i+1]=Pa[i];
    }
 printf("Done\n");
```

```
/******* Steady State Pressure Drop ***************/

 printf("Calculating Steady State Pressures...");
 j=0;while(pe[j].type!=RPIPE && pe[j].type!=SPIPE && j<n) j++;
 do {
 Vel+=increment;
 for(i=0;i<n;i++)
  {
   Vel2=Vel*Area(pe[j].p2)/Area(pe[i].p2);
   switch (pe[i].type)
   {
    /* These constants are defined in presdef.h */
    case SPIPE          : dp=DPSpipe(Vel2,pe[i]);break;
    case RPIPE          : dp=DPRpipe(Vel2,pe[i]);break;
    case CONTRACTION    : dp=DPContraction(Vel2,pe[i]);break;
    case ELBOW          : dp=DPElbow(Vel2,pe[i]);break;
    case TEE_BRANCH     : dp=DPTeeBranch(Vel2,pe[i]);break;
    case BALL_VALVE     : dp=DPBallValve(Vel2,pe[i]);break;
    case TANKEXIT       : dp=DPTankExit(Vel2,pe[i]);break;
    case ENLARGEMENT    : dp=DPEnlargement(Vel2,pe[i]);break;
    case PIPE_EXIT      : dp=DPPipeExit(Vel2,pe[i]);break;
    case VAR_BALL_VALVE : dp=DPBallValve(Vel2,pe[i]);break;
    default: printf("\nI do not know how to handle %s",pe[i].typename);break;
   }
   Ps[i+1]=Ps[i]-dp-pe[i].direction*pe[i].rho*G;
  } /*End of For loop*/
    if (Ps[n]<Plow)
     {
      if (Plow-Ps[n]<100) Terminate=1;
      Vel-=increment;
      increment*=0.10;
     }
  } while (!Terminate);
 printf("Done\n");

/*Printing and writing the results*/
```

```
printf("Velocity in first Pipe Section :  %2.3f m/s\n",Vel);
massflow=pe[j].rho*Area(pe[j].p2)*Vel;
printf("Mass Flow Rate :  %2.3f kg/s\n",massflow);
printf("Printing Results...");
outmatlab2d("pdi.m",Length-1,Pi-1,n+1,64);
outmatlab2d("pda.m",Length-1,Pa-1,n+1,65);
outmatlab2d("pds.m",Length-1,Ps-1,n+1,66);
printf("Done\n");
}
/************** End of pres2.c **************/
```

A.5 Transient system declerations transdef.h

```
/************* transdef.h *******************/
/*** Definitions for Pressure drop elements **/

#define KTankExit 1.0
#define KStandardElbow 0.6
#define KStandardTee 1.14
#define KBallvalve 0.95
#define G 9.81
#define PI 3.14159265
#define TANK 0
#define TANK_EXIT 1
#define LARGE_PIPE_START 2
#define BRANCH 3
#define LARGE_PIPE_END 4
#define SMALL_PIPE_START 5
#define FLASHING 6
#define SMALL_PIPE_END 7
#define EXIT 8

/******* Definition for Array Bound *******/

#define MAXPOINTS 300

/********* Structure Definitions ***********/
struct Water
{
 float P;
 float T;
 float Vel;
 float rho;
 float mu;
 float h;
};
```

```
struct PlotParam
{
 int transvel;
 int transacc;
 int transpre1;
 int transvalve;
 int transdis;
 int transflash;
 int transpre2;
};

struct ValveParam
{
 float time1;
 float angle1;
 float Ttime;
 float angle[MAXPOINTS];
};

/* End of transdef.h */
```

A.6 Transient system calculation program trans3.c

```
/***************** trans3.c ***********************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "transdef.h"
#include "frictlib.h"
#include "input.h"
#include "output.h"
#include "watertab.h"
#include "maths.h"
#include "presdef.h"
#include "preselem.h"
#include "frict.h"

float X1,Y1,Y2; /* Only to be used for Math library */

/*******************************************************/

float Trapezoidal(float(*dydx)(float,float,float,
            struct PressureElem *,float,int,int,float),
            float c1,float c2,float c4,struct PressureElem *pe,
            float yn,int ne,int j,float angle,float h)
  /*
   General Second order numerical solution function.
   */
{
 float y1n,ynp1;
 y1n=yn+(*dydx)(c1,c2,c4,pe,yn,ne,j,angle)*h;
 ynp1=yn+h/2*((*dydx)(c1,c2,c4,pe,yn,ne,j,angle)+
          (*dydx)(c1,c2,c4,pe,y1n,ne,j,angle));
 return(ynp1);
}

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

```
int index(int i,int j,int n)
{
 return(j+i*n);
}

/*******************************************************/

float Flashing_Point(float *Phistory,int n,int tinterval,int ne,float Psat,
                    struct PressureElem *pe)
{
  /*
   This function finds the point where saturation pressure is equal
   to the local pressure in the pipe. It is called in every time step.
   */
 int i=0,j;
 float inter,P2,P1,length=0;
 while (Phistory[index(i+1,n+1,tinterval+1)]>Psat && i<ne) i++;
 for (j=0;j<i;j++) length+=pe[j].length;
 P2=Phistory[index(i+1,n+1,tinterval+1)];
 P1=Phistory[index(i  ,n+1,tinterval+1)];
 inter=pe[i].length/(P2-P1)*(Psat-P1);
 return(length+inter);
}

/*******************************************************/

float AdvanceWaterInPipe(float x_now,struct PressureElem *pe,float v,
                         float dt,int j,int ne)
{
  /*
    This function calculates the place of hot/cold water interface
    inside the pipe. All the complications come from the fact that
    pipe has different diameters.
   */
 float inter,ret,Lt=0,left,vol_displaced,volleft,x_covered;
 int i=0,k;
 x_covered=x_now;
 while (Lt<=x_now)
 {
   if (i>=ne) return(x_now);
   Lt+=pe[i].length;
   i++;
 }
 i--;
 left=Lt-x_now;
 vol_displaced=(v*Area(pe[j].p2)/Area(pe[i].p2))*dt*Area(pe[i].p2);
for(k=i;k<ne;k++)
{
 volleft=left*Area(pe[k].p2);
 if (volleft>=vol_displaced) {
                             inter=(vol_displaced)/Area(pe[k].p2);
                             return(x_covered+inter);
                            }
 vol_displaced-=volleft;
 x_covered+=left;
 if (k<(ne-1)) left=pe[k+1].length;
 else return(x_covered);
} /*end of loop*/
 return(0);
}

/*******************************************************/

float ff(float B)
{
  /*
   First order system approximation that can not be written in closed
   form. It is solved by a numerical method.
   */
 return(log(Y2/(Y2+Y1*B))+B*X1);
}
```

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

void FitExponential(float xx1,float yy1,float yy2,float *A,float *B)
{
  /*
    Finds the coefficents for first order system approximation.
    The BrentMethod function is in maths.h and definition is similar
    to that given in Recipies in C.
   */
 X1=xx1;
 Y1=yy1;
 Y2=yy2;
 *B=BrentMethod(ff,0.1,15.0,0.0);
 if (*B!=0) *A=Y2/(*B*exp(-(*B)*X1));
 else *A=0;
}

/*********************************************************/

float FrictionPDelement(struct PressureElem pe,float Vel,float angle)
{
  /*
    Branches the pressure drop calculation to appropriate element.
   */
 float dp;
    switch (pe.type)
     {
      case SPIPE          : dp=DPSpipe(Vel,pe);break;
      case RPIPE          : dp=DPRpipe(Vel,pe);break;
      case CONTRACTION    : dp=DPContraction(Vel,pe);break;
      case ELBOW          : dp=DPElbow(Vel,pe);break;
      case TEE_BRANCH     : dp=DPTeeBranch(Vel,pe);break;
      case BALL_VALVE     : dp=DPBallValve(Vel,pe);break;
      case TANKEXIT       : dp=DPTankExit(Vel,pe);break;
      case ENLARGEMENT    : dp=DPEnlargement(Vel,pe);break;
      case PIPE_EXIT      : dp=DPPipeExit(Vel,pe);break;
      case VAR_BALL_VALVE : dp=DPVarBallValve(Vel,pe,angle);break;
      default : printf("\nI do not know how to handle %s",pe.typename);break;
     }
  return(dp);
}

/*********************************************************/

float AverageC3term(struct PressureElem *pe,int ne,int j,float *c3p)
{
 /*
  This function calculates the coeffiecent in front of the velocity
  term in the differential equation. It approximates the K factors
  requiring velocity. The K of Variable ball valve is taken as standard
  ball valve.
  The c3 consists of:
  c3=c3Av+c3p*KvariableBallvalve
  This coeffient includes everything except the velocity itself
  unlike the simulation functions which calculates the term itself.
 */
 int i;
 float Aratio=0,tmp,c3=0;
 *c3p=0;
  for(i=0;i<ne;i++)
    {
     Aratio=Area(pe[j].p2)/Area(pe[i].p2);
     tmp=pe[i].rho*AverageKfactor(pe[i])*Aratio*Aratio;
     if (pe[i].type!=VAR_BALL_VALVE) c3+=tmp;
     else *c3p=pe[i].rho*Aratio*Aratio;
   } /*End of loop i*/
   c3=c3/(pe[ne-1].rho*pe[ne-1].length*2);
   *c3p=*c3p/(pe[ne-1].rho*pe[ne-1].length*2);
 return(c3);
}
```

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

float FrictionPDTerm(struct PressureElem *pe,float Vel,int ne,
                 int j,float angle)
{
 /*
   Calculates the c3 term in every time step.
  */
 float Vel2,dp,Dp=0,c3=0;
 int i;
  for(i=0;i<ne;i++)
    {
     Vel2=Vel*Area(pe[j].p2)/Area(pe[i].p2);
     Dp+=FrictionPDelement(pe[i],Vel2,angle);
   } /*End of loop i*/
   c3=Dp/(pe[ne-1].rho*pe[ne-1].length);
 return(c3);
}

/*******************************************************/
/* The coeffients c1, c2 and c4 are calcualted once at   */
/* the beginning of program to speed up simulation       */
/* Cases of negative acceleration are filtered out       */

float dvdt(float c1,float c2,float c4,struct PressureElem *pe,
         float Vel,int ne,int j,float angle)
{
 /*
   This is the differential equation with variable coeffiencts
   to be solved. If c3 term is averaged, there is a rather
   strange closed form solution involving tan(x) in
   imagenary domain. But averaging degrades solution considerably.
  */
 float presterm,fricterm,c3;
 c3=FrictionPDTerm(pe,Vel,ne,j,angle);
 presterm=-c2/c1-c4/c1;
 fricterm=c3/c1;
 if (presterm>fricterm)
   return(presterm−fricterm);
 else return(0.0);
}

/**********************************************************/

float de2(float v,float dvdt,float Ar,struct PressureElem pe,float angle)
{
 float dp1,dp2;
 dp1=FrictionPDelement(pe,Ar*v,angle);
 dp2=(dvdt*Ar+pe.direction*G)*pe.rho*pe.length;
 return(dp1+dp2);
}

/**********************************************************/

int CheckVariableValve(struct PressureElem *p,int ne)
{
 int i;
 for (i=0;i<ne;i++) if (p[i].type==VAR_BALL_VALVE) return(1);
 return(0);
}

/******************** MAIN ***************************/

void main(int argc, char **argv)
{
 struct Water w[10],Hotw,Coldw;
 struct PlotParam p;
 struct ValveParam v;
 struct PressureElem pe[MAXELEMENT];
 float t[MAXPOINTS],xfront[MAXPOINTS],xflash[MAXPOINTS],mdot[MAXPOINTS],
       dydt[MAXPOINTS],y[MAXPOINTS],dy2dt2[MAXPOINTS],
       Phistory[MAXPOINTS*MAXELEMENT],
    Length[MAXELEMENT],Mass[MAXELEMENT],elem[MAXELEMENT],paramf[MAXELEMENT];
int ret,tinterval,n,i,j,ne;
float time,Psat,dt,Inter,TurbinL,TurbinS,A,B,fhw,fcw,Ar,TotalK=0,
Tlength=0,Tmass=0,c1=0,c2=0,c4=0,c3Av=0,c3p=0;

printf("Transient Response Simulation of Piping Systems\n");
printf("by Murat Erenturk March 1996\n");
if (argc<2) {
        printf("Usage:\ntrans3 <setupfile> <datafile>\n\n");
        exit(0);
        }

/*************** Reading Data files ****************/

printf("Reading Setup from %s...",argv[1]);
ret=ReadSetup(argv[1],pe,&ne);
if (ret) {printf("Exiting Program\n");exit(-1);}
printf("Done\n");
printf("Reading Data from %s...",argv[2]);
ret=ReadData(argv[2],&Hotw,&Coldw,&time,&tinterval,&xfront[0],
        &v,&p);
if (ret) {printf("Exiting Program\n");exit(-1);}
printf("Done\n");
dy2dt2[0]=0;
/***************Initialization******************/
printf("Initializing Variables...");
if (tinterval>MAXPOINTS)
  {
  printf("Predefined MAXPOINTS is less than Tinterval\n");
  exit(0);
  }
Hotw.rho=WaterSubDen(Hotw.P,Hotw.T+273);
Hotw.h=WaterSubEnt(Hotw.P,Hotw.T+273);
Hotw.mu=WaterSatVisP(Hotw.P);
Coldw.rho=1/WaterSatSpecDenP(Coldw.P);
Coldw.T=60.0;
```

```
 Coldw.mu=WaterSatVis(Coldw.T+273);
 Psat=WaterSatPress(Hotw.T+273);
/* Above functions are in watertab.c which is a water
properties library. The data can be given from water
tables instead*/

 dt=time/tinterval;
 y[0]=0;
 Length[0]=0;
 for (i=0;i<(ne+1);i++) {
                      Phistory[index(i,0,tinterval+1)]=Hotw.P;
                      elem[i]=i+1;
                     }
 for (i=0;i<=tinterval;i++)
  {
   t[i]=i*dt; /*Setting Up Time series*/
   /*Setting Up boundary Conditions*/
   Phistory[index(0,i,tinterval+1)]=Hotw.P;
   if (i!=0) xfront[i]=0;
   xflash[i]=0;
  }
 for (i=0;i<=ne;i++)     /*Calculating mass and Length*/
   {
     pe[i].rho=Coldw.rho;pe[i].mu=Coldw.mu;
     /***All elements have the same viscosity and density*/
     if (pe[i].type==RPIPE || pe[i].type==SPIPE)
        {
         Tlength+=pe[i].p1;
         pe[i].length=pe[i].p1;
         Mass[i]=Area(pe[i].p2)*pe[i].p1*pe[i].rho;
        }
    else {
          Mass[i]=0;
         }
   }
  for (i=0;i<=ne;i++)    /*Calculating total mass and length*/
 {
  if (pe[i].type==RPIPE || pe[i].type==SPIPE)
    Length[i+1]=Length[i]+pe[i].p1;
  else if (pe[i].type==TANKEXIT || pe[i].type==PIPE_EXIT)
    {
    Length[i+1]=Length[i]+Tlength*1E-6;
    pe[i].length=Tlength*1E-6;
    }
  else {
        Length[i+1]=Length[i]+Tlength*0.01;
        pe[i].length=Tlength*0.01;
       }
  Tmass+=Mass[i];
 }
 else if (!CheckVariableValve(pe,ne)) {
                                  for (i=0;i<tinterval;i++)
                                   v.angle[i]=90;
                                  }
 else v.angle[0]=con.ValveStartAngle;

for (i=0;i<ne;i++)
 TotalK+=AverageKfactor(pe[i]);
 printf("Done\n");fflush(stdout);
 if (!CheckVariableValve(pe,ne))
   printf("WARNING! There is no Control Valve in the system\n");

/*****************Differential equation Coeffiecients*************/

j=0;while(pe[j].type!=RPIPE && pe[j].type!=SPIPE && j<ne) j++;
printf("Calculating Diff. Equ. Coefficents...");fflush(stdout);
  for (i=0;i<ne;i++)
  {
   c1+=pe[i].rho*pe[i].length/(pe[ne-1].rho*pe[ne-1].length)*
       Area(pe[j].p2)/Area(pe[i].p2);
   c4+=pe[i].rho*pe[i].length/(pe[ne-1].rho*pe[ne-1].length)*
       pe[i].direction;
 }
 c2=(Coldw.P-Hotw.P)/(pe[ne-1].rho*pe[ne-1].length);
 c3Av=AverageC3term(pe,ne,j,&c3p);
printf("Done\n");fflush(stdout);
DisplayElements(pe,ne);
printf("Time interval :%2.4f sec %% %2.2f of total\n",
      dt,dt/time*100);
printf("Saturation pressure:  %3.2f kPa\n",Psat/1000);
printf("Hot Water D,V: %3.2f kg/m3, %3.2E Pa.s\n",Hotw.rho,Hotw.mu);
printf("Cold Water D,V: %3.2f kg/m3 %3.2E Pa.s\n",Coldw.rho,Coldw.mu);
printf("Total length of Pipe:  %2.2f m\n",Tlength);
printf("Total mass of water :  %2.3f kg\n",Tmass);
printf("Total of K factors  :  %2.3f\n",TotalK);
if (con.active) printf("Setting Valve Closure Time to :  %2.3f sec\n",con.tclosure);
printf("Differential Equation Coeff:\n");
printf("c1=%f,c2=%f,c3=%f+%f*Kbv,c4=%f\n",c1,c2,c3Av,c3p,c4);

/****************** Time Loop **************************/

printf("Approximating Differential Equation...");fflush(stdout);
for(n=0;n<tinterval;n++)
  {
   if (n>0 && con.active) v.angle[n]=ValveController(v.angle[n-1],dydt[n-1],dt*n,dt,con,&(paramf[n-1]));
   if (n==tinterval/2) {printf("1/2...");fflush(stdout);}
   dydt[n]=dvdt(c1,c2,c4,pe,y[n],ne,j,v.angle[n]);
   y[n+1]=Trapezoidal(dvdt,c1,c2,c4,pe,y[n],ne,j,v.angle[n],dt);
   if (n>0) dy2dt2[n]=(dydt[n]-dydt[n-1])/dt;
   for (i=0;i<ne;i++)
    {
     Ar=Area(pe[j].p2)/Area(pe[i].p2);
     Phistory[index(i+1,n+1,tinterval+1)]=Phistory[index(i,n+1,tinterval+1)]
                                -de2(y[n],dydt[n],Ar,pe[i],v.angle[n]);
    }
   xfront[n+1]=AdvanceWaterInPipe(xfront[n],pe,y[n+1],dt,j,ne);
   xflash[n+1]=Flashing_Point(Phistory,n,tinterval,ne,Psat,pe);
 } /*End of Loop*/
```

```
/****** Printing Initial Results ***************/

 printf("Done\n");fflush(stdout);
 printf("Estimating Steady State...\n");
 FitExponential((n-1)*dt,y[n-1],dydt[n-1],&A,&B);
 if (B!=0) {
 printf(" Estimated S.S. Velocity :  %3.2f m/s\n",A);
 printf(" Estimated S.S. Time :  %3.2f sec\n",(1/B)*5);
        }
 else printf(" Error in Exponential Fitting!\n");
 xflash[0]=xflash[1]=xflash[2];

/********** Writing results to files ****************/

 printf("Printing Results...");
 outmatlab2d("transdis.m",t-1,xfront-1,tinterval-1,p.transdis);
 outmatlab2d("transfla.m",t-1,xflash-1,tinterval-1,p.transflash);
 outmatlab2d("transvel.m",t-1,y-1,tinterval-1,p.transvel);
 outmatlab2d("transacc.m",t-1,dydt-1,tinterval-1,p.transacc);
 outmatlab3d("transpre.m",t,elem,Phistory,tinterval+1,ne+1,65);
 outmatlab2d("transval.m",t-1,v.angle-1,tinterval-1,p.transvalve);
 outmatlab2d("transjer.m",t-1,dy2dt2-1,tinterval-1,71);
 outmatlab2d("transprf.m",t-1,paramf-1,tinterval-1,71);
printf("Done\n\n");
}
/************* End of trans3.c **********************/
```

A.7 Speed of Sound function header file spsoulib.h

```
/********************* spsoulib.h *************************/

/**************** Function Declerations ****************/

extern float SpeedofSoundofWaterInElasticPipe(float Ew,float Ep,float rhow,
                        float Do,float Di);
extern float IdealGasSpeedofSound(float k,float R,float T);
extern float SpeedCompressivePresWaveinSatMixture(float v,float vl,float vg,
    float Cl,float Cg);

/************** Constant Definitions **********************/

#define Est 207E9 /*Young's Modulus of Elasticity of Steel [Pa]*/
#define Ew 2E9   /*Young's Modulus of Elasticity of Water [Pa]*/
#define Ecu 119.3E9 /*Young's Modulus of Elasticity of Copper [Pa]*/
#define Epl 2.4E9  /*Young's Modulus of Elasticity of Plastic [Pa]
            This is a representative value ranges between 0.76-7*/
```

A.8 Speed of sound function library `spsoulib.c`

```
/***************** spsoulib.c ***************/

#include <stdio.h>
#include <math.h>

/**************** Function Definitions *********/

float SpeedofSoundofWaterInElasticPipe(float Ew,float Ep,float rhow,
                        float Do,float Di)
{
 float tmp;
 if (Do<Di || Di<0) {
                printf("Do=%2.2f and Di=%2.2f not proper in speed of sound calculation\n",Do,Di);
                return(0);
               }
  tmp=1+(Ew*(Do+Di))/(Ep*(Do-Di));
  if (rhow<=0) printf("Warning:  Density is Zero in Sound Calculation\n");
  if (tmp<=0) printf("Warning:  Denominator zero in Sound Calculation\n");
  return(sqrt(Ew/(rhow*tmp)));
}

/*****************************************************************/

float IdealGasSpeedofSound(float k,float R,float T)
{
 return(sqrt(k*R*T));
}

/******************************************************************/

float SpeedCompressivePresWaveinSatMixture(float v,float vl,float vg,
     float Cl,float Cg)
{
 float tmp1,tmp2,tmp3,tmp4,tmp;
 tmp=vl*Cg/(vg*Cl);
 tmp1=v/vg*sqrt(1-(vl/vg));
 tmp2=v/vg*(1-tmp*tmp);
 tmp3=vl/vg-tmp*tmp;
 tmp4=sqrt(tmp2-tmp3);
 return(tmp1/tmp4*Cg);
}
/********* End of spsoulib.c ********************/
```

A.9 Speed of sound calculation program ssound.c

```
/***************** ssound.c *********************/

#include <stdio.h>
#include <stdlib.h>
#include "spsoulib.h"
#include "input.h"
#include "output.h"
#include "watertab.h"
#define G 9.81
#define PI 3.1415926535
#define Rw 462
#define kw 1.33
#define XINTERVAL 40

/*********** Quick and Dirty Variables *********/
/************ Bad programing practice *********/
float D1,Do1,rhow,Tw,Pw,rhost,cw1,cw2,cw3,cIdealGas,x,vl,vg,vmix,
    cMixcomp1,cMixcomp2,cMixcomp3;
FILE *in;
char s1[120];
/************************MAIN*****************************/
void main(int argc,char **argv)
{
 int i;
 float vf[100],vr[100],inc,SS[100],X[100],Spike1[100],Spike2[100],Spike3[100];
 printf("\nSpeed of Sound Calculation in Pipes\n");
 printf("By Murat Erenturk          November 1995\n");
 if (argc<2)
  {
   printf("Usage:\nssound <datafile>\n\n");
   exit(-1);
  }
/****************Reading Data**********************/
printf("Reading Data...");
```

```
in=fopen(argv[1],"r");
if (in==NULL) {
          printf("Unable to open file %s for reading\n",argv[1]);
          exit(-1);
         }
ReadToEndofLine(s1,80,in);
 fscanf(in,"%f ",&Tw);ReadToEndofLine(s1,80,in);
 fscanf(in,"%f ",&D1);ReadToEndofLine(s1,80,in);
 fscanf(in,"%f ",&Do1);ReadToEndofLine(s1,80,in);
 fscanf(in,"%f ",&x);ReadToEndofLine(s1,80,in);
 fclose(in);
 printf("Done\n");
/**************General Considerations*****************/
Do1=D1+2*Do1/1000;

Pw=WaterSatPress(Tw+273);
vl=WaterSatSpecDenP(Pw);
rhow=1/vl;
rhost=SteamSatDenP(Pw);
vg=1/rhost;
inc=x/XINTERVAL;
X[1]=0;
cw1=SpeedofSoundofWaterInElasticPipe(Ew,Est,rhow,Do1,D1);
cw2=SpeedofSoundofWaterInElasticPipe(Ew,Ecu,rhow,Do1,D1);
cw3=SpeedofSoundofWaterInElasticPipe(Ew,Epl,rhow,Do1,D1);
Spike1[1]=1*cw1/vl/1E5;
Spike2[1]=1*cw2/vl/1E5;
Spike3[1]=1*cw3/vl/1E5;

/******* Looping over Qulaity *********/

for (i=2;i<=(XINTERVAL+1);i++)
{
vmix=vg*i*inc+(1-i*inc)*vl;
vf[i]=1/vmix;
vr[i]=vmix/vg;
X[i]=i*inc;
cIdealGas=IdealGasSpeedofSound(kw,Rw,Tw+273);
cMixcomp1=SpeedCompressivePresWaveinSatMixture(vmix,vl,vg,cw1,cIdealGas);
cMixcomp2=SpeedCompressivePresWaveinSatMixture(vmix,vl,vg,cw2,cIdealGas);
cMixcomp3=SpeedCompressivePresWaveinSatMixture(vmix,vl,vg,cw3,cIdealGas);
SS[i]=cMixcomp1/cIdealGas;
Spike1[i]=1*cMixcomp1/vmix/1E5;
Spike2[i]=1*cMixcomp2/vmix/1E5;
Spike3[i]=1*cMixcomp3/vmix/1E5;
}

/********** Writing and printing the results ***************/

printf("Water Temperature    :  %3.2f C\n",Tw);
printf("Water Pressure       :  %3.2f kPa\n",Pw/1000);
printf("Quality              :  %3.2f \n",x);
printf("vmix, vl, vg :  %3.2e %3.2e %3.2e \n",vmix,vl,vg);
printf("Pipe Diameter :  %3.2f mmID %3.2fmmOD\n",D1*1000,Do1*1000);
printf("Speed of sound in Water in Steel pipe:  %3.2f m/s\n",cw1);
printf("Spike in One phase flow :  %3.2f Atm\n",cw1*5/vl/1E5);
printf("Speed of s.  in steam IG : %3.2f m/s\n",cIdealGas);
printf("Speed of s.  in Mixture  :  %3.2f m/s\n",cMixcomp1);
printf("Writing Results...");
outmatlab2d("ssound1.m",X,Spike1,XINTERVAL,64);
outmatlab2d("ssound2.m",X,Spike2,XINTERVAL,65);
outmatlab2d("ssound3.m",X,Spike3,XINTERVAL,66);
printf("Done\n");
}
/************* End of ssound.c *************/
```

A.10 Output library header file output.h

```
/******************* output.h ***************************/
/******** Writes results to files in different formats *******/

/*********** Function Declerations **********************/

void outresult(char *filename,int n,float *b,float *rval,
          int digit,int suppress);
void outmatlab3d(char *filename,float *x,float *y,float *u,long n,long n1,int options);

/*Options bits are
1 : figure 2
2 : figure 3
4 : figure 4
16 : (on) surface plot (off) mesh plot
32: output absolute values
*/

void outmatlab2d(char *filename,float *x,float *y,long n,int options);

/*Options bits are
1 : figure 2
2 : figure 3
4 : figure 4
8 : X− range Axis (on) logorithmic (off) linear
16: Y− range Axis (on) logorithmic (off) linear
*/
```

A.11 Output library functions `output.c`

```
/****************** output.c ***********************/

#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<math.h>

/********* Function Defintions *********************/

void outresult(char *filename,int n,float *b,float *rval,int digit,int suppress)
{
 FILE *fp;
 int i,j,sup;
 char stars[10],number[10],zero[10],pv[60];

 sup=0;
 if ((fp=fopen(filename,"w"))==NULL) {
                          printf("Solution file could not be opened\n");
                          exit(1);
                                      }

 switch (digit)
   {
   case 0:
         strcpy(number," %+2.0f ");
         strcpy(zero,"  0 ");
         break;
   case 1:
         strcpy(number,"% -2.1f ");
         strcpy(zero,"   0 ");
         break;
   case 2:
         strcpy(number,"% -2.2f ");
         strcpy(zero,"    0 ");
```

```
        break;
    case 3:
        strcpy(number,"% -2.3f ");
        strcpy(zero,"      0 ");
        break;
    default:
        strcpy(number,"% -2.5f ");
        strcpy(zero,"        0 ");
        break;
    }


for (i=1;i<=n;i++)
 {
  if ((b[i]<1E-4 && b[i]>-1E-4) && suppress) sup=1;
  else sup=0;
  if (!sup) {
        strcpy(pv,"u(%-2d)=");
        if (b[i]==0) strcat(pv,zero);
        else  strcat(pv,number);
        strcat(pv,"ur=");
        if (rval[i]==0) strcat(pv,zero);
        else strcat(pv,number);
        fprintf(fp,pv,i,b[i],rval[i]);
        if ((b[i]>1E-3 || b[i]<-1E-3) && (rval[i]>1E-3 || rval[i]<-1E-3))
          fprintf(fp," err=%% %2.1f\n",(b[i]-rval[i])/rval[i]*100);
          else fprintf(fp,"\n");
         }
 }
fclose(fp);
}

/***************************************************************************/

void outmatlab3d(char *filename,float *x,float *y,float *u,long n1,long n2,int options)
{
```

```
FILE *fp;
int i,j,figno;
long tmp;
char e;

figno=(options & 7)+1;
tmp=n1*n2;
 if ((fp=fopen(filename,"w"))==NULL) {
                    printf("Matlab 3-D output file could not be opened\n");
                              exit(1);
                       }
fprintf(fp,"%% Matlab output file - for %ld global elements\n",tmp);
fprintf(fp,"x3d%d=[\n",figno);
for (i=0;i<n1;i++) fprintf(fp,"%f \n",x[i]);
fprintf(fp,"]\n");
fprintf(fp,"y3d%d=[\n",figno);
for (i=0;i<n2;i++) fprintf(fp,"%f \n",y[i]);
fprintf(fp,"]\n");

fprintf(fp,"%% n=%ld, columns=%ld rows=%d\n",tmp,n1,n2);
fprintf(fp,"z%d=[\n   ",figno);
for (i=0;i<n2;i++) {
             if (options & 32) for (j=0;j<n1;j++)
                          fprintf(fp,"%f  ",fabs(u[i*n1+j]));
             else for (j=0;j<n1;j++) fprintf(fp,"%f  ",u[i*n1+j]);
             fprintf(fp,"\n   ");
            }
fprintf(fp,"]\n");
if (!(options & 64))
{
fprintf(fp,"figure(%d)\n",figno);
if (options & 16) fprintf(fp,"surface(x3d%d,y3d%d,z%d)\n",figno,figno,figno);
else fprintf(fp,"mesh(x3d%d,y3d%d,z%d)\n",figno,figno,figno);
fprintf(fp,"view(-37.5,45)\n");
}
fclose(fp);
```

```
}

/*************************************************************************/

void outmatlab2d(char *filename,float *x,float *y,long n,int options)
{
 FILE *fp;
 int i,j,figno;

 if ((fp=fopen(filename,"w"))==NULL) {
                        printf("Matlab 2-D output file could not be opened\n");
                                exit(1);
                          }
 figno=(options & 7)+1;
 fprintf(fp,"%% Matlab output file - for %ld global elements\n",n);
 fprintf(fp,"x%d=[\n",figno);
 for (i=1;i<=n;i++){
                if (fabs(x[i])>1E-3) fprintf(fp," %f \n",x[i]);
                else fprintf(fp," %E \n",x[i]);
               }
 fprintf(fp,"]\n");
 fprintf(fp,"y%d=[\n",figno);
 for (i=1;i<=n;i++){
                if (fabs(y[i])>1E-3) fprintf(fp," %f \n",y[i]);
                else fprintf(fp," %E \n",y[i]);
               }
 fprintf(fp,"]\n");
if (!(options & 64))
{
 fprintf(fp,"figure(%d)\n",figno);
 if ((options & 8) && !(options & 16)) fprintf(fp,"semilogx(x%d,y%d)\n",figno,figno);
 else if (!(options & 8) && (options & 16)) fprintf(fp,"semilogy(x%d,y%d)\n",figno,figno);
 else if ((options & 8) && (options & 16)) fprintf(fp,"loglog(x%d,y%d)\n",figno,figno);
 else fprintf(fp,"plot(x%d,y%d)\n",figno,figno);
 fprintf(fp,"grid;\n");
}
```

```
fclose(fp);

}

/************** End of output.c **************/
```

Appendix B

Data Files

B.1 Steady State analysis data file

```
% Setup Dimensions File
Tank_Exit 0 0 50E-3 0
Ball_Valve 0 0 50E-3 0
Rough_Pipe 0 6.4 50E-3 0
Elbow 1 75E-3 50E-3 0
Rough_Pipe 1 1.194 50E-3 0
Elbow 1 75E-3 50E-3 0
Rough_Pipe 0 3.302 50E-3 0
Tee_Branch 0 0 50E-3 0
Ball_Valve 0 0 50E-3 0
Rough_Pipe -1 1.194 50E-3 0
Contraction 0 50E-3 19E-3 0
Smooth_Pipe -1 0.4 19E-3 0
Enlargement 0 19E-3 50E-3 0
Ball_Valve 0 0 50E-3 0
Pipe_Exit 0 0 50E-3 0
```

B.2 Transient analysis - data file

```
% Data file for Transient system analysis program
300E3    P2 – Pressure at the beginning of large pipe(Pa)
100E3    P7 – Pressure at the end of Small pipe(Pa)
125      T  – temperature of water(C)
2.0      time – Time span of Simulation(sec)
120      tinterval – number of time slices
0.00     xfront[0] – Point of Hot water initially(m)
0.75     Valve opening time for first position(sec)
45.0     Valve opening for first position(degrees)
1.50     Total Valve opening time(sec)
64       Plot properties for V
65       Plot properties for Acceleration
68       Plot properties for v.angle
69       Plot properties for Xfront
70       Plot properties for Xflashing
```

B.3 Transient analysis - Simple setup Tank and Pipe

```
% Simple data file
Rough_Pipe 0 13.4 50E−3 0
Smooth_Pipe 0 0.4 19E−3 0
```

B.4 Transient analysis - Wolf's setup without fittings

```
% Wolf's setup without fittings
Rough_Pipe 0 6.4 50E−3 0
Rough_Pipe 1 1.194 50E−3 0
Rough_Pipe 0 3.30 50E−3 0
Rough_Pipe −1 1.194 50E−3 0
Smooth_Pipe −1 0.381 19E−3 0
```

B.5 Transient analysis - Wolf's setup full elements

This file is the same as steady state analysis data file.

B.6 Transient analysis - Proposed setup

```
% Proposed setup data file
Tank_Exit 0 0 50E−3 0
Ball_Valve 0 0 50E−3 0
Rough_Pipe 0 13.4 50E−3 0
Contraction 0 50E−3 19E−3 0
Smooth_Pipe 0 0.4 19E−3 0
Var_Ball_Valve 0 0 50E−3 0
Pipe_Exit 0 0 50E−3 0
```

B.7 Transient analysis - Real Application setup

```
% Setup Dimensions for actual Plant
Tank_Exit 0 0 559E−3 0
Rough_Pipe −0.5 1.0 559E−3 0
Elbow 0 750E−3 559E−3 0
Rough_Pipe 0 9.15 559E−3 0
Elbow 0 750E−3 559E−3 0
Rough_Pipe 0 12.2 559E−3 0
Elbow 0 750E−3 559E−3 0
Rough_Pipe 0 9.15 559E−3 0
Var_Ball_Valve 0 0 559E−3 0
Rough_Pipe 0 1.0 559E−3 0
Pipe_Exit 0 0 559E−3 0
```

Bibliography

[1] Allievi, L. "Theory of Water Hammer." (translated by E.E. Halmos), Rome, Ricardo-Garoni, 1925.

[2] Avallone, Baumeister III. *Mark's Standard Handbook for Mechanical Engineers.* New York, McGraw Hill, pp. 3–71, 1987.

[3] Brown, F.C. and W.L. Kranich "A Model For The Prediction of Velocity and Void Fraction Profiles in Two-Phase Flow.", *Advances In Chemical Engineering Journal*, Vol. 14, pp. 233–237, 1968.

[4] Crane Company Technical Paper No 410M *Flow of Fluids.* Crane Co. 1982.

[5] Henry R., Fauske. "The Two-Phase Critical Flow of One-Component Mixtures In Nozzles, Orifices and Short Tubes." *Journal of Heat Transfer, Trans. ASME, Series C*, Vol 93, pp. 179–187, May 1971.

[6] Jelev, I. "The damping of flow and pressure oscillations in water hammer analysis." *Journal of Hydralic Research*, Vol 27, pp. 91–114, No 1, 1989.

[7] Joukowsky, N. "Water Hammer." (translated by O. Simin), *Proceedings American Water Works Association*, Vol. 24, 1904.

[8] Levy,S. "Prediction of Two-Phase Critical Flow Rate." *Journal of Heat Transfer, Trans. ASME, Series C*, Vol 87, pp. 53–58, February 1965.

[9] Lobo, C. and P. Griffith "Avoiding Steam Bubble Collapse-Induced Water Hammer in the Auxiliary Piping of Steam Power Plants." *Journal of Pressure Vessel Technology, Trans. ASME*, Vol 116, pp. 49–56, February 1994.

[10] Moody, Frederick J. *Introduction To Unsteady Thermofluid Mechanics.* New York, John Wiley & Sons, pp. 70–86, 1990.

[11] Moody, Frederick J. "Maximum flow rate of a single component two-phase mixture." *Journal of Heat Transer, Trans. ASME, Series C*, Vol 87, pp. 134–42, February 1965.

[12] O'Brien,M. and G. Hickox *Applied Fluid Mechanics.* New York, Mc Graw-Hill, pp. 231–259, 1937.

[13] Parmakian,J. *Waterhammer Analysis.* New York, Dover, 1963.

[14] Press, W., S. Teukolsky, W. Vetterling and B. Flannery *Numerical Recipies in C.* New York, Cambridge University Press, 1992.

[15] Rohsenow, WM. and J.P. Hartnett. *Handbook of Heat Transfer.* New York, McGraw Hill, 1st Edition, pp. 14-1–14-22, 1973.

[16] Sharp,B.B. *Water Hammer, Problems and Solutions.* London, Edvard Arnold, 1981.

[17] Starkman,E.S., V.E. Schrock, K.F. Neusen and D.J. Maneely "Expansion Of A Very Low Quality Two-Phase Fluid Through A Covergent-Divergent Nozzle." *Journal of Basic Engineering*, pp. 247–256, June 1964.

[18] Stone & Webster Engineering Corporation *Water Hammer Handbook for Nuclear Plant Engineers and Operators.* 1992.

[19] Streeter, V.L. and E.B. Wylie *Fluid Mechanics.* New York, McGraw Hill, 8th edition, 1985.

[20] Sweeney, E. "Water Hammer Transients In Two-Phase Flow." Masters Thesis in ME, MIT, 1989.

[21] Tangren, R.F., C. H. Dodge and H.S. Seifert "Compressibility Effects in Two-Phase Flow." *Journal of Applied Physics*, Vol. 20, pp. 637–645, July 1949.

[22] Wolf,Arne "Passive Water Hammer Due to Flashing Flows In Pipes." Senior project presented at Technische Universitaet Muenchen, Germany, 1991.

[23] Wylie, E.B. and V.L. Streeter *Fluid Transients.* New York, McGraw Hill, 1978.

Biograpghy

Murat Erentürk was born in Ankara, Turkey in 1971. He went to primary and secondary school in TED Ankara Collage. He graduated in May 1989. After entering the university enterance examinations in June 1989, we was placed in Middle East Technical University Mechanical Engineering program, his first choice.

During his four years of education he spend time in different parts of the country learning different diciplines of Mechanical Engineering. In 1990 he visited Karabürk Iron and Steel plant for one month where he learned the basics of steel production. During the summer of 1991 he worked as a researcher in TÜBİTAK Defense Research and Development Institute. In 1992 he worked in Oyak-Renault Automobile factories as a part of his internship. He also visited Eskişehir Locomotive Factory for 2 weeks.

He graduated as a Mechanical Engineer in June 1993. During the summer of 1993 he worked in ERENTÜRK Consulting Engineering as a Mechanical Engineer. He was chosen to be a Fulbright Scholar in 1994 and was accepted to MIT for a Master of Science program. He graduated in June 1996 from the Mechanical Engineering department with a Master of Science in Mechanical Engineering degree. During his years at MIT he was a member of the Turkish Students Association Executive Commitee.

Apart from visiting major US cities he travelled extensively in Europe including United Kingdom, Netherlands, Italy, Germany and Switzerland. Other than his mother tongue Turkish he speaks fluent English and is proficent in German.

He is interested in philosophy, pshcology, history and art, especially music. He plays classical guitar and piano. He likes trekking, cyling and playing tennis.