NTNU
Norges teknisk-naturvitenskapelige universitet

The Norwegian University of Science and Technology

Rapportnummer
Gradering

POSTADRESSE
NTNU
INSTITUTT FOR ENERGI OG PROSESSTEKNIKK
Kolbjørn Hejes vei 1A
N-7941 Trondheim-NTNU

TELEFONER
Sentralbord NTNU: 73 59 40 00
Instituttkontor: 73 59 38 60
Vannkraftlaboratoriet: 73 59 38 57

TELEFAX
Instituttkontor 73 59 35 80
Vannkraftlaboratoriet : 73 59 38 54

Rapportens tittel	Dato **14.06.2010**
Computational models for transient gas transport	Antall sider og bilag 102
Avdeling **Institutt for Energi og Prosessteknikk**	Rapportnummer
ISBN nr	Studieprogram: Produktutvikling og produksjon Fordypning: Anvendt mekanikk
Oppdragsgiver	Oppdragsgivers ref.

Norwegian gas accounts for approximately 15 % of the total consumption of natural gas in Europe. In order to deliver the gas to the marked in continental Europe, long pipelines are used to transport the natural gas.

Metering stations of the state of the gas are only found at inlets and outlets of the pipelines. Therefore the state of the gas throughout the pipeline can only be found by using computational programs.

When we refer to what we call transient flow, meaning flow that is changing over time, the calculation become more complicated. Typical events describing such an event could be the closure or opening of a valve. This will result in a change in the mass flow rate, and as a consequence the pressure is also changing.

A brief survey of available methods for calculating transient flow in a natural gas pipelines is presented, with a specialization in the method of characteristics. The method of characteristics was used when developing a computer program to solve the transient flow.

A set of different transients were modelled and the results were compared to values that had been obtained from Gassco's test section going from the processing plant at Kårstø to the island Vestre- Bokn. This section is a 12,2 km long testsection, and is the first step of the 658 km long Europipe II, that reaches from Kårstø to Dronum In Germany.

The report also gives a brief explanation of the flow inside of a natural gas pipeline, and the simplifications done in order to solve the governing equations.

Three events were simulated. Oscillation of the flow rate, closure of a valve at the inlet and the opening of the valve. The first to events were separated into two cases. For the oscillating flow we investigated both an ideal sine-oscillations with a large magnitude, and a real event where the flow was unsteady and had small oscillations. The closure of a valve involved a simulation of the rapid closure, with its following oscillations in the flow, whereas the other case investigated a stepwise closure according to real data. A simple investigation of the stepwise opening of the valve was also done. This event was however not simulated for the entire time period, due to long simulation time, and results that were not satisfactory compared to real measurements.

Due to the simplifications done in the model, and the lack of sufficient boundary conditions
resulted in calculated values that differ a lot from the measured values. The method might however prove useful when the boundary conditions are well known, and the flow is rather simple.

	Stikkord på norsk	Indexing terms English
Gruppe 1		
Gruppe 2		
Egenvalgte stikkord	**Transient strømning, Kårstø-Bokn**	**Transient flow in a natural gas pipeline, Kårstø-Bokn**

Norges teknisk-
naturvitenskapelige universitet
NTNU

Institutt for energi- og prosessteknikk

NTNU

EPT-P-2010-1

PROSJEKTOPPGAVE

for

Stud.techn. Joachim Dyrstad Gjerde

Våren 2010

Transiente beregningsmodeller for gasstransport

Transient computation models for gas transport

Bakgrunn: *Transport av naturgass i lange rørledninger på havbunnen er en stor og voksende eksportindustri – særlig i Norge. Til dette trenges det bl.a. sikre beregningsmodeller for å planlegge/optimalisere leveranser i nær fremtid, og for å overvåke rørsystemenes integritet. Strømningen i rørledningene er i mange tilfeller tidsavhengig (=transient), og dette må reflekteres i de aktuelle beregningsmodellene.*

Mål: *Å først få frem en oversikt over tilgjengelige numeriske beregningsmodeller, og spesielt vurdere bruk av karakteristikkmetoden versus andre differansemetoder. Dernest å velge/etablere en modell, med tanke på middels store rørlengder (opptil ca100km), og å gjennomføre beregninger basert på denne for typisk transiente situasjoner som oppstart/nedstengning. Det er til slutt et mål å foreta sammenligninger med resultater fra målinger på fullskala transportanlegg hos Gassco.*

Oppgaven bearbeides ut fra følgende punkter:

1. En oversikt over tilgjengelig informasjon på beregningsmodeller i gasstransport, både fra interne rapporter og fra internasjonale fagjournaler.
2. En fordypning i karakteristikkmetoden og en vurdering av denne opp mot andre aktuelle numeriske beregningsmetoder.
3. Valg av beregningsmetode og numerisk implementering av denne.
4. Numeriske beregninger av typiske transiente forløp, og med driftsparametre som tilsvarer eksempelvis strekningen Kårstø – Bokn i Gasscos transportsystem.
5. Diskusjon av oppnådde resultater og konklusjoner.

Belastningen på prosjektet utgjør 50%, eller 15 studiepoeng, under forutsetning av at kandidaten har valgt fordypningsemne(r) tilsvarende 7.5 Stp.

-- " --

Senest 14 dager etter utlevering av oppgaven skal kandidaten levere/sende instituttet en detaljert fremdrift- og evt. forsøksplan for oppgaven til evaluering og evt. diskusjon med faglig ansvarlig/ veiledere. Detaljer ved evt. utførelse av dataprogrammer skal avtales nærmere i samråd med faglig ansvarlig.

Besvarelsen redigeres mest mulig som en forskningsrapport med et sammendrag både på norsk og engelsk, konklusjon, litteraturliste, innholdsfortegnelse etc. Ved utarbeidelsen av teksten skal kandidaten legge vekt på å gjøre teksten oversiktlig og velskrevet. Med henblikk på lesning av besvarelsen er det viktig at de nødvendige henvisninger for korresponderende steder i tekst, tabeller og figurer anføres på begge steder. Ved bedømmelsen legges det stor vekt på at resultatene er grundig bearbeidet, at de oppstilles tabellarisk og/eller grafisk på en oversiktlig måte, og at de er diskutert utførlig.

Alle benyttede kilder, også muntlige opplysninger, skal oppgis på fullstendig måte. (For tidsskrifter og bøker oppgis forfatter, tittel, årgang, sidetall og evt. figurnummer.)

Det forutsettes at kandidaten tar initiativ til og holder nødvendig kontakt med faglærer og veileder(e). Kandidaten skal rette seg etter de reglementer og retningslinjer som gjelder ved fagmiljøer som kandidaten har kontakt med gjennom sin utførelse av oppgaven, samt etter eventuelle pålegg fra Institutt for energi- og prosessteknikk.

I henhold til "Utfyllende regler til studieforskriften for teknologistudiet/sivilingeniørstudiet" ved NTNU § 20, forbeholder instituttet seg retten til å benytte alle resultater i undervisnings- og forskningsformål, samt til publikasjoner.

Ett -1 komplett eksemplar av originalbesvarelsen av oppgaven skal innleveres til samme adressat som den ble utlevert fra. (Det skal medfølge et konsentrert sammendrag på maks. en maskinskrevet side med dobbel linjeavstand med forfatternavn og oppgavetittel for evt. referering i tidsskrifter).
Til Instituttet innleveres to - 2 komplette, kopier av besvarelsen. Ytterligere kopier til evt. medveiledere/oppgavegivere skal avtales med, og evt. leveres direkte til, de respektive.
Til instituttet innleveres også en komplett kopi (inkl. konsentrerte sammendrag) på CD-ROM i Word-format eller tilsvarende.

Institutt for energi og prosessteknikk, 1. februar 2010

Olav Bolland
Instituttleder

Tor Ytrehus
Faglig ansvarlig/veileder

Preface

This paper is written as a report on the project originally intended for the fall semester in 5th grade of the M.Sc. degree at the Energy and Process Department at NTNU. The purpose of the project was to give an overview over numerical methods available for computations of flow in natural gas pipelines, with a special focus on the Method of Characteristics.

This report presents the mathematical background of the method of characteristics, and the numerical implmentation of this. Results are presented and compared to measured data from a test section at Kårstø to Bokn.

The report requires some technical understanding of the concepts, but is written in the simplest way making it understandable without making too much effort.

The project was done under the supervision of professor Tor Ytrehus, and I would like to give my regards to him for excellent advice and encouragement throughout the period when the project was done.

Trondheim, 14th June 2010

Joachim Dyrstad Gjerde

Abstract

Norwegian gas accounts for approximately 15 % of the total consumption of natural gas in Europe. In order to deliver the gas to the marked in continental Europe, long pipelines are used to transport the natural gas.

Metering stations of the state of the gas are only found at inlets and outlets of the pipelines. Therefore the state of the gas throughout the pipeline can only be found by using computational programs.

When we refer to what we call transient flow, meaning flow that is changing over time, the calculation become more complicated. Typical events describing such an event could be the closure or opening of a valve. This will result in a change in the mass flow rate, and as a consequence the pressure is also changing.

A brief survey of available methods for calculating transient flow in a natural gas pipelines is presented, with a specialization in the method of characteristics. The method of characteristics was used when developing a computer program to solve the transient flow.

A set of different transients were modelled and the results were compared to values that had been obtained from Gassco's test section going from the processing plant at Kårstø to the island Vestre- Bokn. This section is a 12,2 km long testsection, and is the first step of the 658 km long Europipe II, that reaches from Kårstø to Dronum In Germany.

The report also gives a brief explanation of the flow inside of a natural gas pipeline, and the simplifications done in order to solve the governing equations.

Three events were simulated. Oscillation of the flow rate, closure of a valve at the inlet and the opening of the valve. The first to events were separated into two cases. For the oscillating flow we investigated both an ideal sine-oscillations with a large magnitude, and a real event where the flow was unsteady and had small oscillations. The closure of a valve involved a simulation of the rapid closure, with its following oscillations in the flow, whereas the other case investigated a stepwise closure according to real data. A simple investigation of the stepwise opening of the valve was also done. This event was however not simulated for the entire time period, due to long simulation time, and results that were not satisfactory compared to real measurements.

Due to the simplifications done in the model, and the lack of sufficient boundary conditions resulted in calculated values that differ a lot from the measured values. The method might however prove useful when the boundary conditions are well known, and the flow is rather simple.

Samandrag

Norsk gass utgjer kring 15 % av det totale forbruket i Europa. For å kunne levere denne gassen frå Norsk sokkel til det Europeiske fastlandet, vert det nytta lange transportrøyr frå prosess anlegga i Noreg til Europa og Storbritannia. Desse røyrstrekka strekkjer seg frå 500 til 800 kilometer.

Ettersom ein berre kan gjere målingar av gassen ved innløp og utløp av røyret, kan tilstanden gassen er i gjennom røyrstrekket berre finnast ved å bruke numeriske berekningsverkty.

Når ein referer til transient straum, meinast det at straumen i røyret endrast over tid. Eit eksempel der straumen endrast over tid, er dersom vi opnar eller lukker ein ventil i røyret på eit tidspunkt. Dette resultere i at massestraumen i røyret endrast samstundes med trykket, og gir straumen ein ny tilstand.

Ei lita oversikt over aktuelle metodar for berekning av transient gas transport i røyr er gitt. Det er deretter gjort ei fordjuping i ein metode med namn karakteristikkmetoden. Denne metoden vert og nytta når eit program vart skrive for å kunne løyse dei karakteristiske likningane numerisk. Mykje av den same prosedyren vart og gjort av Stig Grafsrønningen i 2006, og deler av hans arbeid har vore nytta som basis.

Ei rekkje transientar vart modellert, og resultata av desse modelleringane vart samanlikna med reelle måledata gjort av Gassco på strekket Kårst til Vestre Bokn. Dette 12,2 kilometer lange teststrekket er det fyrste steget av det 658 kilometer lange gassrøyret Europipe II, som strekkjer seg frå Kårstø til Dornum i Tyskland.

I hovudsak vart 3 ulike transiente hendingar modellerte. Desse hendingane var oscillerande straumrater, lukking av ventil på innløpet og ei opning av den same ventilen. For den oscillerande straumen, to ulike hendingar vart berekna. Den eine hendinga er ei simulering av ei reell hending, der resultata vart samanlikna med måledata for den same hendinga. Den andre hendinga var ein oscillerande straum med store oscillasjonar, i ein sinuskurve. Ved lukking, vart både ei fullstendig avstenging av ventilen på eit minutt, og ei hending der massestraumen vart stegvis nedtrappa simulert. Opninga av ein ventil skjer over ei særs lang til og berre kring 1/3 av tida vart simulert ettersom dei oppnådde resultat hadde eit stort avvik frå målingar.

Grunna forenklingar gjort i modellen vår og grensevilkår som ikkje strekk til, vert dei oppnådde resultata ikkje nøyaktige i høve til dei avleste resultata i dei same hendingane. Men den numeriske berekningsmetoden gjev fysisk truverdige resultat, og vil ein del høver der grense- og start vilkåra er tilstrekkeleg, truleg gje relativt fornuftege resultat.

Contents

List of Figures

List of Tables

Nomenclature

ϵ	wall roughness
κ	von Karman constant
λ	multiplier
μ	viscosity
ν	kinematic viscosity
ρ	density
ρ'	density perturbation
τ_w	wall friction
θ	pipeline angle
$\triangle t$	timestep
$\triangle x$	length of pipeline segment
^oC	temperature Celcius
^oK	temperature, Kelvin
C^+	Characteristic
C^-	Characteristic
c_f	skin friction factor
f	Darcy friction factor

p_0	density at rest
p_0	pressure at rest
R_q	Root means square
Re	Reynolds number
U^+	inner velocity
u_*	friction velocity
y^+	inner distance
A	pipeline area
B	gas wavespeed
D	pipeline diameter
g	gravity constant
M	mass flow rate
MoC	method of characteristics
p'	pressure perturbation
R	gas constant
S	wetted perimeter
U	gas velocity
u'	velocity perturbation
Z	compressibility factor
z	roughness height

1 Introduction

1.1 Background

Natural gas is a very important in energy supply world wide. And Norway are one of the main exporters of natural gas to markeds worldwide. Total gas produced on Norwegian continental shelf has more than doubled over the last 10 years, and was in 2009, the equivalent of 103,470 million Sm^3[2], which accounted for approximately 3% of the total gas production in the world. For the European marked, Norwegian gas covers around 15%. Natural gas to the European marked, including the United Kingdom, is transported through 7 pipelines. These pipelines vary from 500 km and up to 800 km in length, and is operated by the Norwegian, government owned corporation Gassco.

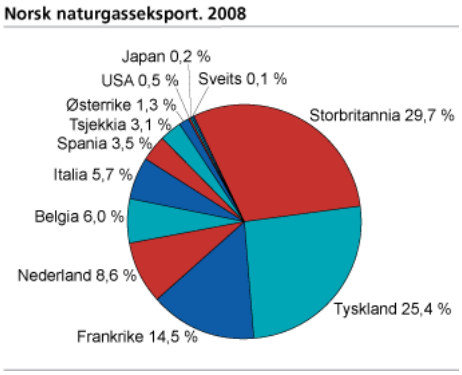


Figure 1.1.1: Natural gas export in 2008[2]

The state of the natural gas and the flow rates can not be measured throughout the whole pipeline, and computer models are used to calculate the state of the gas. These models prove very important when it comes to utilizing the full potential of the pipelines. With as much as 70 million standard cubic metres MSm^3 transported in a pipeline daily, a small error in computed results can prove very costly. Accurate models for the state of the gas may also simplify the operation of a pipeline.

1.2 Kårstø

Kårstø is a processingplant located a bit north of Stavanger in western Norway, and plays a key role in Norwegian gas export[1]. It receives rich gas and condensate from the

North sea and the Norwegian sea through the Åsgard transport system, and Statpipe. The gas is being processed and fractionated into natural gas, liquified petroleum gases and condensate. Kårstø serves as one of the main export ports for LPG's. The natural gas is transported to Dornum, Germany, through Europipe II, and to Emden, also located in Germany, through Statpipe/Norpipe.

1.3 Europipe II

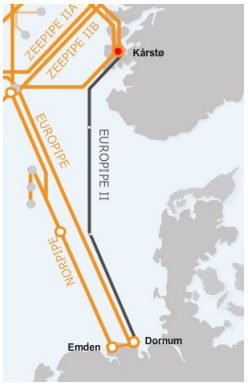


Figure 1.3.1: Euopipe II from Kårstø to Dornum[3]

Europipe II is the pipeline transporting natural gas from the facility at Kårstø to the European continent in Dornum. The length of the pipeline is approximately 660 km and it mainly goes undersea. The pipeline first became operational in 1999.

Before going undersea, Europipe II contains a test section of around 12,2 km that runs from the processing plant to Vestre-Bokn.

1.3.1 Kårstø- Bokn

When going through the test section form Kårstø to Vestre-Bokn, the pipeline crosses two islands and goes undersea between. The maximum difference in tha altitude is about 210 meters. At Vestre -Bokn, Gassco have done measurements of the gas before the pipeline submerges and travels through the North Sea. The route from Kårstø to Bokn is shown below.

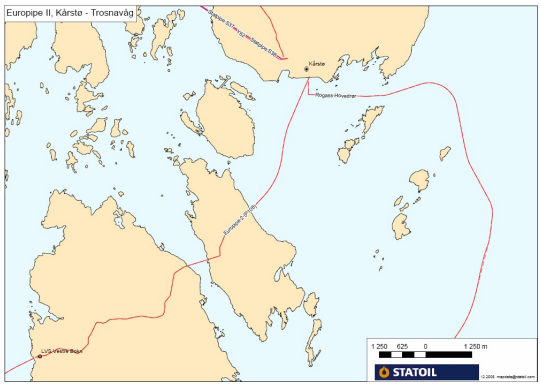


Figure 1.3.2: Section from Kårstø to Bokn

The dissertations thesis of Langelandsvik [12] also holds an elevation profile of the pipeline. This elevation profile is later used in a simplified form when the profile is used in a numerical model.

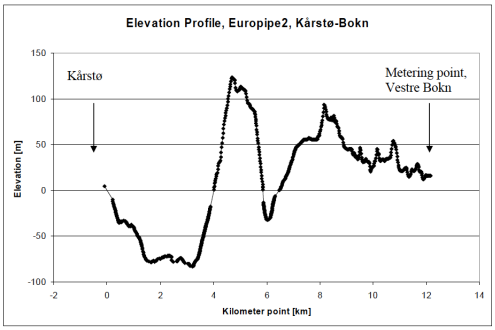


Figure 1.3.3: Elevation profile

1.4 Natural gas

The natural gas from a reservoir consists mainly of methane (CH_4), but it undergoes an extensive processing in order to remove most other species. The processing however leaves other by-products in the gas. However, methane still remains the most important specie in the gas (>90%), but other species such as ethane, propane, CO_2 and N_2 are present. . Typical values are for the dry gas transported in Europipe II were[7].

1.5 Transient-/time dependent modelling

With a daily amout of as much as 70 MSm^3 of gas flowing through the pipeline each day, as much as 25.5 billion standard cubic metres of gas can be transported each year. Exact

Natural gas composition		
C1	Methane	89.15%
C2	Ethane	7.35%
C3	Propane	0.51%
iC4	iso-Butane	0.025%
nC4	normal-Butane	0.030%
iC5	iso-Pentane	$9.0 \cdot 10^{-3}$%
nC5	normal-Pentane	$2.06 \cdot 10^{-3}$%
C6	Hexane	0%
CO^2	Carbon dioxide	2.22%
N^2	Nitrogen	0.699%

Table 1.1: Natural gas composition

knowledge of the state of the gas at different events, such as startup and shutdown of a pipeline is therefore of great value. Not only are there a lot of money involved in such an operation, time dependent changes in the conditions may results in significant changes in the flow properties, which again could cause problems with the processing. These timedependent changes will from now on be refered to as transients. Typical examples of transient flow, are closing and opening of a valve, flow oscillations or involvement of other components such as pumps.

Several methods have been used and presented in order to solve transient cases regarding a flow in a gas pipeline. Some methods can be considered rather complex but the transient flow can also be solved using simpler methods. Complex methods has the advantage of being more exact in their results, but are also more costly. The simplest method involves solving simplified partial differential equations becoming ordinary differential equations. Other methods, such as finite difference methods, are sometimes used. This part will be further discussed in chapter 3.

2 Flow in a gas pipeline

This chapter is meant to give a brief introduction to some of the parameters used to define the flow in a gas pipeline.

2.1 Introduction

Early research done by Stokes and Osbourne Reynold proved that flow at different velocities have different characteristics. In the late 19^{th} century Reynolds proved that at a certain velocity of the flow in a pipe, the flow changed from being laminar into turbulent [19]. From this he introduced the term Reynolds-number, which for a pipe is defined as

$$Re = \frac{\rho U D}{\mu} = \frac{U D}{\nu} \tag{2.1.1}$$

The transition from laminar to turbulent happens in the region $2000 \leq Re_D \leq 2300$, and from Reynoldsnumbers above this we have a turbulent flow with different characteristics than for the slower, laminar flow. The velocityprofile changes together with friction and local pressures.

2.1.1 Laminar flow

For Reynolds numbers below 2300 we say that the flow is laminar. For a pipeflow that means that the flow is following straight streamlines in the pipe direction. For a laminar flow the velocity profile inside a pipeline is parabolic.

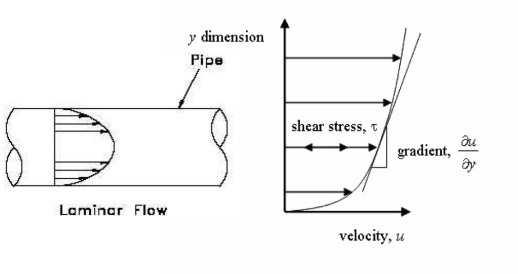


Figure 2.1.1: Laminar velocity profile

2.1.2 Turbulent flow

Turbulence is generated close to the walls and is being transported towards the centre of the pipe. After a distance the entire velocity in the pipeline is turbulent.

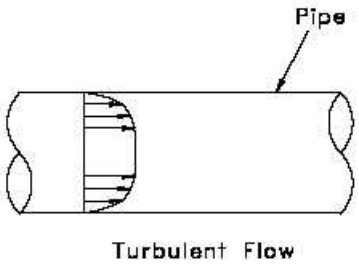


Figure 2.1.2: Turbulent flow in a pipe

A turbulent velocity profile for pipeflow. The velocity gradient is greater at the wall, and the average velocity is quite close to the highest velocity. The turbulent velocity profile is not as straight forward as the laminar one. It is common to divide it into four regions

- Visous sub-layer: Near the wall region, with small velocities and the viscous forces are dominating.
- Buffer layer: The velocities are still relatively small in this region, but turbulent forces are becoming more dominant.
- Logarithmic layer: The flow is totally dominated by turbulent forces, and the velocity only depends logarithmically on y.
- Wake region: This accounts for most of the turbulent velocity distribution.

The logarithmic layer is described

$$U^+ = \frac{1}{\kappa} ln(y^+) + B \tag{2.1.2}$$

Where κ is the con Karman constant, and the other varaibles are defined as

$$U^+ = \frac{U}{\sqrt{\tau_w/\rho}} = \frac{U}{u_*} \tag{2.1.3}$$

$$y^+ = \frac{\rho u_* y}{\mu} \tag{2.1.4}$$

The value u_* is the dimension of velocity and is called the friction velocity. The turbulent velocity profile becomes

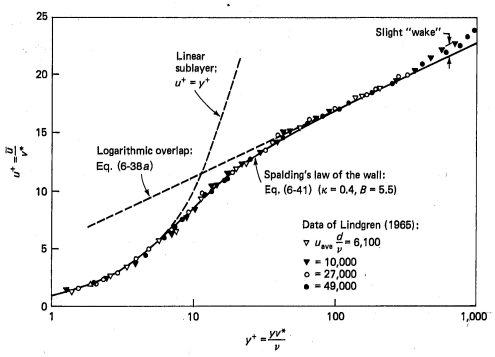


Figure 2.1.3: Turbulent velocity profile[23]

2.2 Pipeline surface

A steel pipeline, even if the pipeline has been extensively treated, never becomes completely smooth. Inside of a pipeline there will always be imperfections, and the pipeline is considered to be rough. But an exact description of the surface at every point of the pipeline is not possible. One therefore tries to obtain a typical value of the pipeline surface which is used throughout the whole pipeline.

Figure 2.2.1: Image of an idealised surface roughness

In reality the surface of a pipeline does not look like figure 2.2. Roughness measurements have been done on real pipelines, and in Dr. Ove Bratlands book on flow assurance [4] several results of measured roughness's are presented. 2.2 show measurements done on a pipeline coated with two-component epoxy coating.

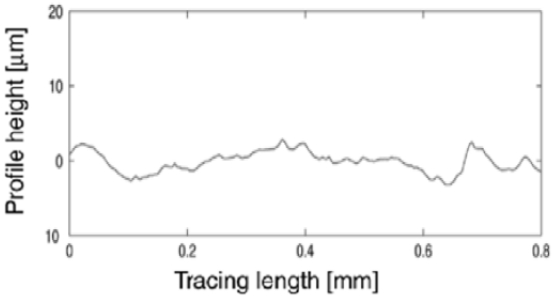


Figure 2.2.2: Measured surface roughness coated pipe[4]

In comparison with the model in figure 2.2 coated pipeline wil have imperfections that will be much more smeared. However it illustrates the principle of the surface roughness. Predictions of the roughness can be done in differents way. One way being the root mean square

$$R_q = \frac{1}{x}\int_0^x \mid z^2(x) \mid dx \tag{2.2.1}$$

where z is the amplitude. From this a dimensionless parameter, ε/D, of the pipeline can be found and used in the calculation of turbulent friction factors.

2.3 Pipeline friction

2.3.1 Darcy friction factor

In a simple pipeline segment, pressure loss is relative to the friction loss, and is denoted the Darcy friction factor.

$$f = \frac{D\frac{\partial p}{\partial x}}{\frac{1}{2}\rho U^2} \tag{2.3.1}$$

Is often used as four times the skin friction, $f = 4c_f$.

2.3.2 Laminar flow

For friction balance in a pipe segment one can write

$$A\triangle p = \tau_w S\triangle x \tag{2.3.2}$$

from this equation one can obtain the pressure drop

$$\frac{\partial p}{\partial x} = \tau_w \frac{4}{D} \tag{2.3.3}$$

Using that $\tau_w = \mu^{\partial U}/_{\partial y}$, gives

$$\frac{\partial u}{\partial y} = \frac{\partial p}{\partial x}\frac{D}{4\mu} \tag{2.3.4}$$

Integrating this over the pipeline cross section gives an average velocity, which combined with the friction factor derived in 2.3.1

$$\frac{32\mu}{D^2}U = \frac{1}{2}f\rho U^2\frac{1}{D} \tag{2.3.5}$$

leads to an expression for the laminar friction factor

$$f = \frac{64}{Re} \tag{2.3.6}$$

2.3.3 Turbulent flow

For the turbulent case it gets more complicated. The turbulent velocity profile is a bit different from the laminar. The turbulence makes the velocity gradient close to the wall larger, as shown in figure2.1.2. It is normal to assume the pipeflow as turbulent once the *Reynoldsnumber* is above 2300.

$$Re = \frac{\rho U D}{\mu} \geqslant 2300 \tag{2.3.7}$$

Which in reality means that almost all flows in gas pipelines are turbulent. A friction-factor however, cannot be given as straight forward for the turbulent case as it was for the laminar case. Prandtl proposed an equation for the friction factor for a smooth pipe

$$\frac{1}{\sqrt{f}} = 2log\left(Re\sqrt{f}\right) - 0.8 \tag{2.3.8}$$

Nikuradse[17] had previously presented a correlation for relatively high Reynoldsnumbers

$$\frac{1}{\sqrt{f}} = -2log\left(\frac{\varepsilon}{3.7D}\right) \tag{2.3.9}$$

Colebrook and White successfully combined the two equations, and giving name to the well known Colebrook equation which is still now the most established equations for calculation the friction factor[5].

$$\frac{1}{\sqrt{f} =} = -2log\left(\frac{\varepsilon/D}{3.7065} + \frac{2.523}{Re\sqrt{f}}\right) \tag{2.3.10}$$

A closer look at the equation reveals that it has the disadvantage of being implicit. We have the frictionfactor f on both sides of the equal sign, meaning that the equations has to be solved using an iterative method. Despite being implicit, the Colebrook equation, has been the most popular method of calculating the frictionfactor of turbulent pipeflow. The frictionfactor f using a logarithmic Reynoldsnumber scale was plotted by Moody in 1944 [16]. The traditional Moody diagram is shown below

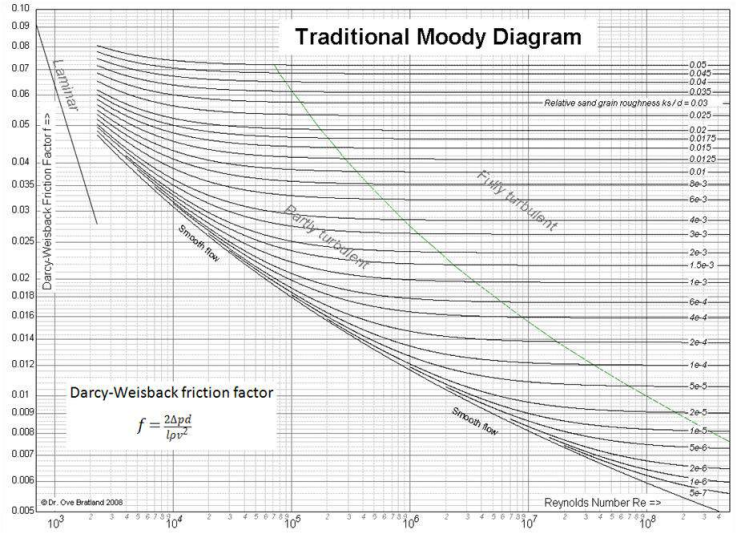


Figure 2.3.1: Moody chart[23]

Haaland later proposed an explicit version of the Coolebrook equation [10], which is quite accurate compared to the implicit version.

$$\frac{1}{\sqrt{f}} = -1.8log\left(\frac{6.9}{Re} + \left(\frac{\varepsilon}{1.11D}\right)^{1.11}\right) \tag{2.3.11}$$

2.4 Solving the turbulent pipeflow

The pipeflow is turbulent, meaning that if we were to solve the governing equations eactly using Navier Stokes equations, Direct Numerical Simualtion(DNS) have to be used. This requires an enourmus amount of computer efforts. Doing a direct numerical simulation of a pipe would require a number of gridpoints in the order of $Re^{9/4}$ according to Kolmogorovs scales[22]. That means having $Re \sim 10^7$ requires the number of nodes in our numerical scheme is in the order of $5.6 \cdot 10^{15}$. In addition comes the timescales. I practice this proves impossible. Direct numerical simulation can only be used for very low Re. Other methods in order to solve turbulent pipeflows can be used.

Large Eddy Simulation:	Soling the larger eddies, while use approximate functions for the smaller ones
Reynolds Stress models:	Solving the reynolds stresses in the flow in all three directions
Two equation models:	Introducing a set of new transport equations for quantities in order to solve the turbulent approximations
One equation models:	An even simpler model accounting for the turbulent viscosity
Algebraic models:	The simplest models, adding only a single value to the existing governing equations

Table 2.1: Methods for solving turbulent pipeflow

All of the above equations involves solving a 3 dimensional flow. We are in general only interested in the x-direction, and small fluctuations in the flow is ignored. Fully solving the turbulent pipeflow is therefore not considered and simplified, one dimensional models are used.

3 Numerical models for solving transient gas transport

We have several numerical methods that we can use in order to calculate the flow in a natural gas pipeline. The four most relevant methods are

- Method of Characteristics
- Finite Difference Method
- Finite Volume Method
- Finite Element Method

Common for all the methods are that they solve the governing equations of the flow. These governing equations are partial differential equations that must be handled in a way that makes them solvable. The methods vary in simplicity.

3.1 Method of Characteristics(MoC)

The rather simple way of solving a transient flow in a natural gas pipeline is using the method of characteristics. The method is a result of the combination of the continuity equation and the transport equations. With some mathematical manipulation the partial differential equations collapse and form ordinary differential equations along a slope in a time and space grid. Further investigation yields that the slopes, or characteristics as they are called, are equal to $C_1 = U + B$, $C_2 = U - B$ and $C_3 = U$, for the momentum equations and the energy equation, respectively. U denotes the velocity of the fluid, and B denotes the acoustic wavespeed For the simplified governing equations inertia term is often neglected, and the characteristics C_1 and C_2 simplifies and becomes $C_{1,2} = \pm B$. In a space-time grid, we see that may cause problems since the location of the solutions are not located in the same place.

Since the stability criterion used for MoC, is the Courant condition[14], $(\mid U \mid + B) \leq \triangle x / \triangle t$. We are then using this method to determine timestep, $\triangle t$. This will eventually lead to a very low timestep if the number of nodes is high.

3.2 Finite Difference Method(FDM)

Perhaps the most popular way to solve the governing equations governing a flow in a gas transport pipeline is using a finite difference method. The pipeline is divided into a

finite number of pipe nodes and approximate solutions to the derivatives of the equations are sought at every node. These approximate solutions are also called differences. One therefore reduces the PDE's to an algebraic system of equations that are based on the discrete values of the solution. These differences can get the information forward, central or backwards in the old or new timestep, depending on wether the solution is implicit or explicit. The finite difference method has several advantages. It is rather simple to understand, and all of the equations can be solved at the same location. Using a finite difference method also implies that an implicit solution can be used. The implicit solution is a more complicated to implement, and requires a solving procedure. On the positive side, the method becomes more stable and longer timesteps can be utilized, which for long transients are very convenient.

3.3 Finite Volume Method(FVM)

A popular method used for CFD, or computational fluid dynamics, is the finite volume method. I the finite volume method the equations are transformed in such a way that the equations are integrated over a control volume. For a time dependent flow, integration is then performed over both the control volume and the timestep. The finite volume method can also be solved implicitly, obtaining steady solutions using longer timesteps. A method called TVD, or total variation diminishing, is often used to solve fluid flow in pipes[26]. but on larger scale pipelines this is not used to a large extent.

3.4 Finite Element Method(FEM)

Even though finite element method is most used in solid mechanics, it can also be used for fluid calculations. In the finite element method the pipeline is divided into several elements of a length. The lengths do not have to be equal. Pressure force, flow rates and temperature are approximated using a set of piecewise continuous functions by a Galerkin approach or other methods forming weighted residuals[20]. Using the finite element method involves some rather complicated algebra, but it has been successfully done for simple transient flows[21]. Finite element method is not used extensively for transient pipeline simulations.

Spectral methods

A method that is becoming more popular are so-called spectral methods. This method uses global smooth functions, such as FFT's(Fast Fourier Transforms), to represent the variables, instead of the piecewise smooth functions used in FEM.

3.5 Conclusion

In general we have four methods that can be used to solve the transient flow in a gas pipeline. For the further investigation of our problem, the Method of Characteristics will be used. This method is the simplest method of the methods presented above, but it may also prove useful and quite accurate for some cases of simple flows. The other methods requires more efforts in order to be completely understood and implemented in a computer program.

4 Method of characteristics

The equations governing a natural gas flow through pipelines are the same as for most other fluid flows. We have equations governing continuity, momentum, and energy. In a transient flow the most important ones are inertial- and pressure forces, however in a gas flow, the inertia force is of less importance. The flow is dominated by friction and pressure forces. When deriving the equations needed in the method of characteristics solution, we make the following assumptions[24]:

1. The flow is isothermal, meaning that the temperature is kept constant throughout our calculation.
2. We keep the diameter of the pipe constant.
3. The slope of the pipe remains constant over any particular reach.
4. The equation of state for the gas is given by $p = Z\rho RT$, where Z is the gas compressibility factor. Z is held constant throughout the whole computation.
5. The flow is simplified to one dimensional.
6. Frictionfactor is a function of wall roughness, and Reynoldsnumber.
7. The kinetic energy of the flow is considered unimportant, and therefore neglected.

4.1 Equation of state

The equation for the state of the natural gas is given as:

$$p = \rho ZRT \tag{4.1.1}$$

In which Z represents the compressibility factor, introduced as a factor to make up for the behaviour of a real gas compared to an ideal gas. The compressibility factor is a function of pressure and temperature. And when we assume that we have an isothermal flow, the definition of the wavespeed in the gas is given as

$$B = \sqrt{\left(\frac{\partial p}{\partial \rho}\right)_T} = \sqrt{ZRT} \tag{4.1.2}$$

Since the flow rate is given as million standard cubic metres per day, MSm^3/d, this must be converted into kilograms per second. A standard cubic meter of natural gas is defined

as the amount of gas in a cubic meter at a standardized temperature and pressure, for natural gas pressure is equal to 1 atmosphere, or $101,3\,kPa$, and temperature is 15^oC. From table1.1 the resulting molecular weight of natural gas was calculated, resuklting in a molecular weight of $18.04\,{}^{kg}/_{kmol}$. Calculation to obtain the mass flux in ${}^{kg}/_{s}$ becomes

$$PV = nRT \tag{4.1.3}$$

$$\frac{V}{n} = \frac{RT}{P} = \frac{8.314 \times 288.15}{101,325} = 23.64 \left[\frac{m^3}{kmol}\right] \tag{4.1.4}$$

$$\frac{23.64\,[{}^{m^3}/_{kmol}]}{18.04\,[{}^{kg}/_{kmol}]} = 1.3104 \left[\frac{Sm^3}{kg}\right] = 0.763 \left[\frac{kg}{Sm^3}\right] \tag{4.1.5}$$

$$M = X \left[\frac{Sm^3}{d}\right] \cdot 0.763 \left[\frac{kg}{Sm^3}\right] \cdot \frac{1}{60 \times 60 \times 24} \left[\frac{\mathrm{d}}{s}\right] = X \cdot 8.831 \cdot 10^{-6} \left[\frac{kg}{s}\right] \tag{4.1.6}$$

4.2 Continuity equation

The continuity equation states that the mass has to be conserved. Meaning that the net inflow has to be compensated by a change in volume and/or density.

$$\frac{\partial \rho A}{\partial t} + \frac{\partial \rho AU}{\partial x_i} = 0 \tag{4.2.1}$$

In a pipeline with constant diameter this can be shown

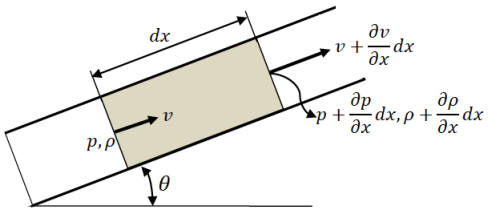


Table 4.1: Continuity

Since the sum of ρAU is equal to the total mass flux, we can rewrite the continuity equation

$$\frac{\partial \rho A}{\partial t} + \frac{\partial M}{\partial x_i} = 0 \tag{4.2.2}$$

The density can be eliminated by using the equation of state, and inserting it into the continuity equation

$$\frac{\partial(Ap/B^2)}{\partial t} + \frac{\partial M}{\partial x_i} = 0 \tag{4.2.3}$$

Now using the notation $\partial p/\partial t = p_t$,, and similar for M, we get

$$p_t + \frac{B^2}{A} M_x = 0 \tag{4.2.4}$$

4.3 Equation of motion

A force balance of a fluid motion in pipe is made

$$\frac{\partial(\rho AU)}{\partial t} + \frac{\partial(\rho AU)}{\partial x} = -A\frac{\partial P}{\partial x} - AS\tau_w - A\rho g sin\theta \tag{4.3.1}$$

Darcy Weisbachs formula for friction factor in terms of force balance is used

$$\tau_w = \frac{1}{2} f\rho U^2 \frac{1}{4} = \frac{1}{8} f\rho(\frac{M}{\rho A})^2 == \frac{f\rho}{8}\frac{M^2B^4}{p^2A^2} \tag{4.3.2}$$

Since we also neglect the changes in the velocity of the gas the part on the left also simplified

$$\frac{DU}{Dt} = U_t + UU_x \approxeq U_t = \frac{1}{A}\left(\frac{M}{\rho}\right)_t = \frac{B^2}{A\rho}(M_t + \frac{M}{p}p_t) \tag{4.3.3}$$

Substitution of the equations into the force balance

$$-p_x \triangle x A - \frac{f\rho}{8}\frac{M^2B^4}{p^2A^2}\pi D\triangle x + \rho g A\triangle x sin\theta = \frac{B^2}{A\rho}\left(M_t - \frac{M}{p}p_t\right)\rho A\triangle x \tag{4.3.4}$$

The latter term on the right will be small compared to the first term.

$$p_x + \frac{1}{A}M_t + \frac{pg}{B^2}sin\theta + \frac{fB^2M^2}{2DA^2p} = 0 \tag{4.3.5}$$

Steady state

Steady state means that the boundary conditions remains constant. Hence the time derivatives are zero, and the only independent variable is in the x direction.

$$\frac{dp}{dx} + \left(\frac{pg}{B^2}sin\theta + \frac{fB^2M^2}{2DA^2p}\right) = 0 \tag{4.3.6}$$

Integrating this equation from $x = 0$ and $p = p_1$ to $x = \triangle x$ and $p = p_2$ gives

$$dp + \left(\frac{pgsin\theta}{B^2} + \frac{fB^2M^2}{2DA^2p} \right) dx = 0 \tag{4.3.7}$$

Multiplying by p and further manipulation gives

$$\frac{1}{2}dp^2 = -\frac{gsin\theta}{B^2} \left(p^2 + \frac{fB^4M^2}{2DA^2gsin\theta} \right) dx \tag{4.3.8}$$

Integrating this equation from $x = 0$ and $p = p_1$ to $x = \triangle x$ and $p = p_2$ gives

$$\int_{p1}^{p2} \frac{1}{p^2 + \frac{fB^4M}{2DA^2gsin\theta}} dp^2 = -\int_0^{\triangle x} \frac{2gsin\theta}{B^2} dx \tag{4.3.9}$$

Simplify by using s=$(2g\triangle xsin\theta/B^2)$

$$\left[\ln(p^2 + \frac{fB^4M^2}{2DA^2gsin\theta}) \right]_{p1}^{p2} = -s \tag{4.3.10}$$

Which also can be written

$$\frac{p_1^2 + \frac{fB^4M^2}{2DA^2gsin\theta}}{p_2^2 + \frac{fB^4M^2}{2DA^2gsin\theta}} = e^s \tag{4.3.11}$$

And when we solve this equation for p_2 to get our steady state pressure after $\triangle x$

$$p_2^2 = (p_1^2 + \frac{fB^2M^2}{DA^2} \triangle x \frac{e^s - 1}{s}) / e^s \tag{4.3.12}$$

From this equation we can calculate the pressure distribution in the entire pipe, which will later on be used as initial conditions when we do transients calculations.

4.4 Solution by the method of characteristics

In section 4.2 we derived the equation of continuity

$$L_1 = \frac{B^2}{A} M_x + p_t = 0 \tag{4.4.1}$$

And we also derived the momentum equation in section 4.3

$$L_2 = p_x + \frac{\alpha^2}{A} M_t + \frac{fB^2M^2}{2DA^2p} + \frac{1 - \alpha^2}{A} M_t = 0 \tag{4.4.2}$$

Here we have introduced the inertial multiplier, α, which is mathematically correct, since

$$\frac{\alpha^2}{A} M_t + \frac{1 - \alpha^2}{A} M_t = \frac{1}{A} M_t \tag{4.4.3}$$

When $\alpha = 1$, the equations is identical to equation 4.4.2. An evaluation on the inertial multiplier will be given in section 4.5. Combining equations L_1 and L_2 using an unknown multiplier yields

$$\lambda L_1 + L_2 = \lambda \left(\frac{B^2}{A} M_x + p_t \right) + p_x + \frac{\alpha^2}{A} M_t + \frac{f B^3 M^2}{2DA^2 p} + \frac{1-\alpha^2}{A} M_t = 0 \tag{4.4.4}$$

Rearranging this further yields

$$\frac{\alpha^2}{A} \left(\frac{\lambda B^2}{\alpha} M_x + M_t \right) + \lambda \left(\frac{1}{\lambda} p_x + p_t \right) + \frac{pgsin\theta}{B^2} + \frac{f B^2 M^2}{2DA^2 p} + \frac{1-\alpha^2}{A} M_t = 0 \tag{4.4.5}$$

And then we compare the first two terms in the caracteristic equation with the substantial derivative

$$\frac{dM}{dt} = \frac{\partial M}{\partial t} + \frac{dx}{dt} \frac{\partial M}{\partial x} \tag{4.4.6}$$

And similar for the pressure. The first part of equation 4.4.5 can be written

$$M_t + \frac{\lambda B^2}{\alpha} M_x = \frac{\partial M}{\partial t} + \frac{dx}{dt} \frac{\partial M}{\partial x} \Longrightarrow \frac{dx}{dt} = \frac{\lambda B^2}{\alpha} \tag{4.4.7}$$

In the same way for the pressure we get

$$\frac{dx}{dt} = \frac{1}{\lambda} \tag{4.4.8}$$

Hence we can write

$$\frac{dx}{dt} = \frac{1}{\lambda} = \frac{\lambda B^2}{\alpha^2} \Longrightarrow \lambda = \pm \frac{\alpha}{B} \tag{4.4.9}$$

Inserting this into our equation

$$\frac{\alpha^2}{A} \left(\frac{dM}{dt} \right) \pm \frac{\alpha}{B} \left(\frac{dp}{dt} \right) + \frac{pgsin\theta}{B^2} + \frac{f B^2 M^2}{2DA^2 p} + \frac{1-\alpha^2}{A} M_t = 0 \tag{4.4.10}$$

Where $^{dx}/_{dt} = \pm ^{B}/_{\alpha}$ is valid. The two values are defined as characteristics and we integrate the equation along its specific characteristic in a finite difference grid.

$$\frac{\alpha^2}{A} \frac{dM}{dt} dx \pm \frac{\alpha}{B} \frac{dp}{dt} dx + \frac{pgsin\theta}{B^2} dx + \frac{f B^2 M^2}{2DA^2 p} dx + \frac{1-\alpha^2}{A} M_t dx = 0 \tag{4.4.11}$$

with leads to

$$\frac{\alpha B}{A} \int_{A,B}^{p} dM \pm \int_{A,B}^{P} dp + \int_{A,B}^{P} \left(\frac{pgsin\theta}{B^2} + \frac{f B^2 M^2}{2DA^2 p} \right) dx + \frac{B}{A} \int_{A,B}^{P} \frac{1-\alpha^2}{\alpha} \frac{\partial M}{\partial t} dt = 0 \tag{4.4.12}$$

The last term in the equation are what makes the inertial multiplier. As we can see from

the grid, the solution depends on adjacent values for the new timestep in the characteristic grid. The term is integrated in the following way

$$\frac{B}{A}\int_{A,B}^{p}\frac{1-\alpha^2}{\alpha}\frac{\partial M}{\partial t}dt \approx \frac{B}{2A}\left(\frac{1}{\alpha}-\alpha\right)(M_{R,P}-M_{A,C}+M_{P,S}-M_{C,B}) \tag{4.4.13}$$

Integration is done along the characteristics, from now on reffered to as $C^{+}and\ C^{-}$

$$C^{+}: \frac{\alpha B}{A}(M_p-M_A)+p_p-p_A+\frac{fB^2\triangle x}{(p_p+p_A)DA^2}\left(\frac{M_p+M_A}{2}\frac{|M_p+M_A|}{2}\right)+$$
$$\frac{p_p^2}{p_p+p_A}(e^s-1)+\frac{B}{2A}\left(\frac{1}{\alpha}-\alpha\right)(M_R-M_A+M_P-M_C)=0 \tag{4.4.14}$$

$$C^{-}: \frac{\alpha B}{A}(M_p-M_B)-p_p+p_B+\frac{fB^2\triangle x}{(p_p+p_B)DA^2}\left(\frac{M_p+M_b}{2}\frac{|M_p+M_B|}{2}\right)+$$
$$\frac{p_p^2}{p_p+p_B}(e^s-1)+\frac{B}{2A}\left(\frac{1}{\alpha}-\alpha\right)(M_p-M_C+M_S-M_B)=0 \tag{4.4.15}$$

The absolute terms are inserted so that the equations are valid for negative flow. The terms in equation4.4 will be explained in the next section. By ignoring the inertial multiplier, α 4.4.14 and **??** becomes

$$C^{+}: \frac{B}{A}(M_p-M_A)+p_p-p_A+\frac{fB^2\triangle x}{(p_p+p_A)DA^2}\frac{e^s-1}{s}\left(\frac{M_p+M_A}{2}\frac{|M_p+M_A|}{2}\right)+\frac{p_p^2}{p_p+p_A}(e^s-1)=0 \tag{4.4.16}$$

$$C^{-}: \frac{B}{A}(M_p-M_B)-p_p+p_B+\frac{fB^2\triangle x}{(p_p+p_B)DA^2}\frac{e^s-1}{s}\left(\frac{M_p+M_B}{2}\frac{|M_p+M_B|}{2}\right)+\frac{p_p^2}{p_p+p_B}(e^s-1)=0 \tag{4.4.17}$$

4.5 Inertial multiplier

Yow introduced an inertial multiplier[25] in the system of equations in order to compensate for the loss of the inertial forces when the equations were simplified. In most situations of a pipe flow, the inertial term is of less importance, however, in large transients the inertial term plays a role. This allows for larger time increments to be utilized, with a small error in the pressure. For long transients, the ability to increase the time increments is of great importance, since the restrictions due to the Courant condition in many cases leads to very low timestep.

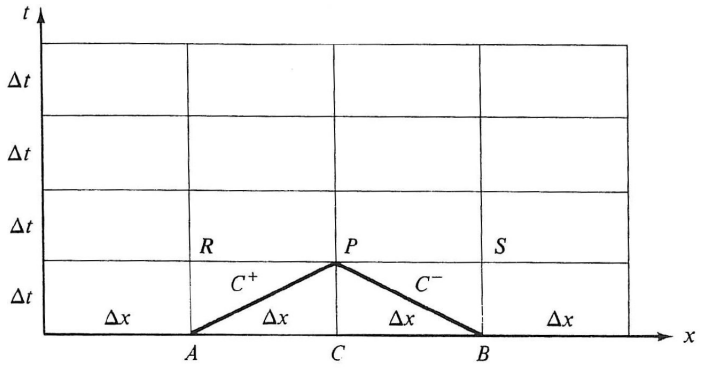


Figure 4.5.1: space and time grid

When $\alpha = 1$ the equations remains the same as they were before introducing the term, and also in the steady state case, when $M_A = M_B = M_P = M_R = M_C = M$ we get the same equations as. The α value is determined by the following equation

$$\alpha^2 = 1 - \frac{(\sigma m h)^2}{3p_d} + \psi \frac{\sqrt{p_d \left(\sigma m \omega^*\right)^2}}{m\omega^* \triangle q} \tag{4.5.1}$$

The quantities are given in dimensionless form as

$$m = \frac{V_0}{B},\ \sigma = \frac{fL}{2D},\ p_d = \sqrt{1 - 2\sigma m^2},\ h = \frac{\triangle x}{L},\ \omega^* = \frac{\omega L}{B}$$

The ψ value denotes the acceptable error of the pressure. $\psi = {}^{\triangle p}/_{p}$. Meaning that if we accept a 5% error in the pressure, the ψ value becomes 0.05. Higher value of ψ results in an increased α.

Depending on the type of transient, the $\triangle q$ and ω^* are defined the following way

Transient	BC	ω	$\triangle q$
Oscillating flow	Massflow	ω	$\triangle q = \frac{\triangle M}{M_0}$
Valve opening/-closing	Massflow	$\omega = \frac{\pi}{2}\frac{dM/dt}{\triangle M}$	$\triangle q = \frac{\triangle M}{M_0}$
	Pressure	$\omega = \frac{\pi}{2}\frac{dp/dt}{\triangle p}$	$\triangle q = \frac{A}{B}\frac{\triangle p}{M_0}$

Table 4.2: ω^* and $\triangle q$ for different types of transients

Solving the equations utilizing the inertial multiplier means that we first have to solve the equations by an iterative method, without including the iertial term,α. The equations are then corrected by introducing the complete set of equations for the inner nodes. As

a consequence we can introduce longer timesteps, $\triangle t_{new} \leq \alpha \triangle t_{old}$. The new equations are also iterative and an iteration process must be done.

4.6 Boundary conditions

We have two characteristic equations for each node we want to calculate in the next timestep. At the boundaries, only one of the equations remains valid. C^+ characteristic at the right boundary, and C^-at the left. A Dirichlet type of boundarycondition must therefore be given at each of the boundaries, since we have two unknowns in our equations, p_p and M_p.

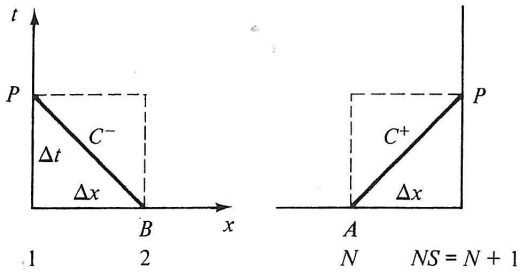


Figure 4.6.1: Characteristics at boundaries

The boundary conditions can be given in the following combinations

1. Pressure is given both at the inlet and the outlet of the pipe
2. Flow rate is given at both pressure and inlet
3. Pressure is given at the inlet, while the flow rate is known at the outlet
4. Flow rate is given at the inlet, and pressure is known at the outlet

4.7 Stability

The method of characteristics suffers under the Courant criterion for stability[14] ($\mid V \mid + a) \leq \frac{\triangle x}{\triangle t}$. In our case, with the simplified equations neglecting the convective term the CFL coundition yields,

$$B \leq \frac{\triangle x}{\triangle t} \tag{4.7.1}$$

Graphically this can be shown by

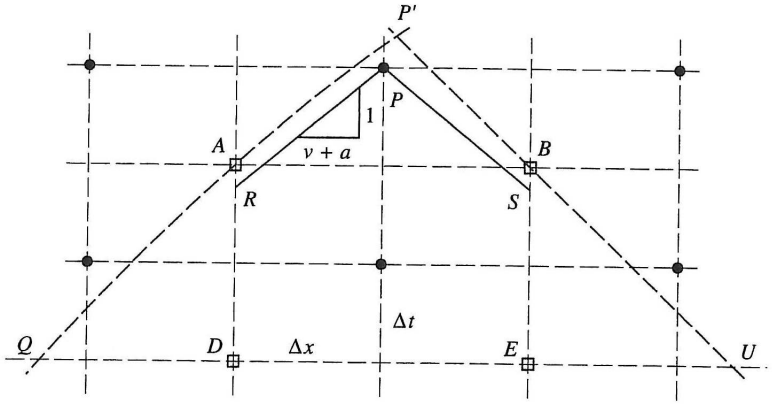


Figure 4.7.1: Stability Method of Characteristic[24]

4.8 Discussions on the Method of Characteristics

As seen from the results above, the information regarding a flow inside a pipeline travels through the pipeline at the speed of sound. This results also proves physically reasonable, as a disturbance or a shock at on one side of the pipeline will create a wave that travels through the pipe until it reaches the other other end after a certain time. These disturbances could for instance be boundary conditions, since an altering of mass flux or pressure at the inflow creates som kind of a discontinuity in the numerical scheme. The waves that form are also called weak shock waves, or acoustic waves.

A brief explanation on the formation of characteristics

If we say that a shock wave is isotropic, meaning that the temperature does not change and the pressure is only is a function of density, $p = p(\rho)$. We have

$$\frac{\partial p}{\partial x} = \frac{\partial p}{\partial \rho}\frac{\partial \rho}{\partial x} = B^2\frac{\partial \rho}{\partial x} \tag{4.8.1}$$

If we then consider a fluid which does not move, a shock can be described as small perturbations in the fluid

$$\rho = \rho_0 + \rho' \tag{4.8.2}$$

$$p = p_0 + p' \tag{4.8.3}$$

$$u = 0 + u' \tag{4.8.4}$$

Where the subscript 0 means that the value is a constant. Then by conisdering our governing equations

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + \rho\frac{\partial u}{\partial x} = 0 \tag{4.8.5}$$

and

$$\rho\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = -B^2\frac{\partial \rho}{\partial x} \tag{4.8.6}$$

using that $\rho' \ll \rho_0$, $p' \ll p_0$ and $u' \ll B$, and inserting 4.8.2to 4.8.4 the equations above becomes

$$\frac{\partial}{\partial t}\left(\rho_0 + \rho'\right) + \rho_0\frac{\partial}{\partial x}\left(0 + u'\right) = 0 \tag{4.8.7}$$

$$\rho_0\frac{\partial}{\partial t}\left(0 + u'\right) + B^2\frac{\partial}{\partial x}\left(\rho_0 + \rho'\right) = 0 \tag{4.8.8}$$

which again is equal to

$$\frac{\partial \rho}{\partial t} + \rho_0\frac{\partial u}{\partial x} = 0 \tag{4.8.9}$$

$$\rho_0\frac{\partial u}{\partial t} + B^2\frac{\partial \rho}{\partial x} = 0 \tag{4.8.10}$$

Then, by multiplying the equations by $1/\rho_0$ and $1/B\rho_0$ respectively, we obtain a dimensionless form. Adding and subtracting the equations yelds

$$\frac{\partial}{\partial t}\left(\frac{u}{B} + \frac{\rho}{\rho_0}\right) + B\frac{\partial}{\partial x}\left(\frac{u}{B} + \frac{\rho}{\rho_0}\right) = 0 \tag{4.8.11}$$

$$\frac{\partial}{\partial t}\left(\frac{u}{B} - \frac{\rho}{\rho_0}\right) - B\frac{\partial}{\partial x}\left(\frac{u}{B} - \frac{\rho}{\rho_0}\right) = 0 \tag{4.8.12}$$

From the equations above one can observe that they are of the form of the substantial derivative of $\left(\frac{u}{B} + \frac{\rho}{\rho_0}\right)$, with the acoustic wavespeed as convection velocity. Integrating these equations reveals

$$\frac{u}{B} + \frac{\rho}{\rho_0} = constant\, along\, x - B \cdot t = constant \tag{4.8.13}$$

$$\frac{u}{B} - \frac{\rho}{\rho_0} = constant\, along\, x + B \cdot t = constant \tag{4.8.14}$$

From these results we get the characteristics lines $x - B\cdot t$ and $x + B\cdot t$, and the quantities $u/B \pm \rho/\rho_0$ are called the Riemann invariants[6].

5 Solving the characteristic equations

The equations derived in the earlier sections are implicitly given. Meaning that the solution is not given explicitly, but as a function of the solution. That could be that one variable is a function of a second variable, but the second variable is also a function of variable number one. Hence we need a method to solve the equations. The method used to solve these equations is Newton's method (also known as the Newton-Rhapson's method) , suggested by Wylie et. al [24].

5.1 Newton's method

Newton's method is a method for solving equations of the form $f(x) = 0$. The function is assumed to have continuous first derivative$f^{'}(x)$, and we start by "guessing" a value and then iterate until the residual is below a acceptable value for the error. In our case the "guessed" value will be the value for the previous timestep, or for the first iteration, our initial conditions.

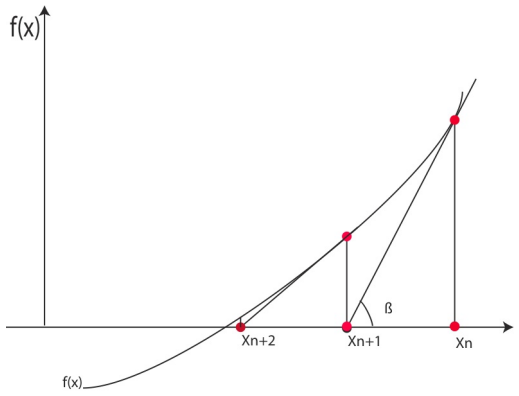


Figure 5.1.1: Newtons method[11]

As seen from figure.5.1 we see that the derivative $f^{'}$ can be defined as

$$f^{'}(x_n) = tan(\beta) = \frac{f(x_n)}{x_n - x_{n+1}} \tag{5.1.1}$$

which gives us

$$x_{n+1} = x_n - \frac{f(x_n)}{f^{'}(x_n)} \tag{5.1.2}$$

As seen from the figure and the equations, this requires and iterative process. the value of x_{n+1} has to be calculated enough times so that $f(x_n) \to 0$ and the error becomes smaller than an accepted value ϵ.

$$x_{n+1} - x_n = -\frac{f(x_n)}{f'(x_n)} \leq \epsilon \tag{5.1.3}$$

Newton's method has the advantage of simple implementation, easy to understand and very fast convergence in many cases, however a problem emerges when the $f'(x_n) \to 0$. Then $f(x_n)/f'(x_n) \to$ inf, and the solution no longer converges. Therefore a maximum number of iterations using the Newton's method is given.

5.2 Solution at the boundaries

At the boundaries one of the values M_p or p_p is given, meaning that we have to solve the equation either along C^+ or C^-. At the inlet, our values ar calculated by either eq... or by eg..., depending on the propery given av the boundary.

$$p_1^{new} = p_1^{old} - \frac{\frac{B}{A}(M_1 - M_2) - p_1^{old} + p_2 + \frac{fB^2 \triangle x}{\left(p_1^{old}+p_2\right) DA^2} \frac{e^s-1}{s} \left(\frac{M_1+M_2}{2} \frac{|M_1+M_2|}{2} \right) + \frac{p_1^{old\,2}}{p_1^{old}+p_2}(e^s-1)}{-1 - \frac{fB^2 \triangle x}{\left(p_1^{old}+p_2\right)^2 DA^2} \frac{e^s-1}{s} \left(\frac{M_1+M_2}{2} \frac{|M_1+M_2|}{2} \right) + \frac{p_1^{old\,2}+2p_1p_2}{(p_1^{old}+p_2)^2}(e^s-1)} \tag{5.2.1}$$

$$M_1^{new} = M_1^{old} - \frac{\frac{B}{A}\left(M_1^{old} - M_2\right) - p_1 + p_2 + \frac{fB^2 \triangle x}{\left(p_1^{old}+p_2\right) DA^2} \frac{e^s-1}{s} \left(\frac{M_1^{old}+M_2}{2} \frac{|M_1^{old}+M_2|}{2} \right) + \frac{p_1^2}{p_1+p_2}(e^s-1)}{\frac{B}{A} + \frac{fB^2 \triangle x}{(p_k+p_{k-1}) DA^2} \frac{e^s-1}{s} \left(\frac{|M_k^{old}+M_{k-1}|}{2} \right)} \tag{5.2.2}$$

At the outlet our boundary equations becomes

$$p_k^{new} = p_k^{old} - \frac{\frac{B}{A}(M_k - M_{k-1}) + p_k^{old} - p_{k-1} + \frac{fB^2 \triangle x}{\left(p_k^{old}+p_{k-1}\right) DA^2} \frac{e^s-1}{s} \left(\frac{M_k+M_{k-1}}{2} \frac{|M_k+M_{k-1}|}{2} \right) + \frac{p_k^{old\,2}}{p_k^{old}+p_{k-1}}(e^s-1)}{1 - \frac{fB^2 \triangle x}{\left(p_1^{old}+p_2\right)^2 DA^2} \frac{e^s-1}{s} \left(\frac{M_1+M_2}{2} \frac{|M_1+M_2|}{2} \right) + \frac{p_1^{old\,2}+2p_1^{old}p_2}{(p_1^{old}+p_2)^2}(e^s-1)} \tag{5.2.3}$$

$$M_k^{new} = M_k^{old} - \frac{\frac{B}{A}\left(M_k^{old} - M_{k-1}\right) + p_k - p_{k-1} + \frac{fB^2 \triangle x}{(p_k+p_{k-1}) DA^2} \frac{e^s-1}{s} \left(\frac{M_k^{old}+M_{k-1}}{2} \frac{|M_k^{old}+M_{k-1}|}{2} \right) + \frac{p_k^2}{p_k+p_{k-1}}(e^s-1)}{\frac{B}{A} + \frac{fB^2 \triangle x}{(p_k+p_{k-1}) DA^2} \frac{e^s-1}{s} \left(\frac{|M_k^{old}+M_{k-1}|}{2} \right)} \tag{5.2.4}$$

5.3 Newtons method for a system of equations

Newton's method can also be applied to a system of dependet equations, as is the case with the characteristic equations above. Instead of dividing $f(x_n)$ on its derivative, we divide the system of functions on its Jacobi matrix,

$$J = \begin{bmatrix} f_{\prime}^{'}(x_1) & \dots & f_1^{'}(x_n) \\ \vdots & \ddots & \vdots \\ f_n^{'}(x_{\prime}) & \dots & f_n^{'}(x_n) \end{bmatrix} \tag{5.3.1}$$

The complete expression for the Jacobi matrix becomes

$$J = \begin{bmatrix} \partial f_1(M_p,p_p)/\partial M_p & \partial f_1(M_p,p_p)/\partial p_p \\ \partial f_2(M_p,p_p)/\partial M_p & \partial f_2(M_p,p_p)/\partial p_p \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\alpha B}{A} + \frac{fB^2 \triangle x}{(p_p+p_A)DA^2} \frac{e^s-1}{s} \left(\frac{|M_p+M_A|}{2} \right) + \frac{B}{2A} \left(\frac{1}{\alpha} - \alpha \right) & 1 - \frac{fB^2 \triangle x}{(p_p+p_A)^2 DA^2} \frac{e^s-1}{s} \left(\frac{M_p+M_A}{2} \frac{|M_p+M_A|}{2} \right) + \frac{p_p^2+2p_p p_A}{(p_p+p_A)^2} (e^s-1) \\ \frac{\alpha B}{A} + \frac{fB^2 \triangle x}{(p_p+p_B)DA^2} \frac{e^s-1}{s} \left(\frac{|M_p+M_B|}{2} \right) + \frac{B}{2A} \left(\frac{1}{\alpha} - \alpha \right) & -1 - \frac{fB^2 \triangle x}{(p_p+p_B)^2 DA^2} \frac{e^s-1}{s} \left(\frac{M_p+M_B}{2} \frac{|M_p+M_B|}{2} \right) + \frac{p_p^2+2p_p p_B}{(p_p+p_B)^2} (e^s-1) \end{bmatrix} \tag{5.3.2}$$

Newton's method for a system of equations then becomes

$$x_{1,2}^{t+\triangle t} = x_{1,2}^t - J^{-1} \cdot f\left(x_{1,2}^t\right)$$

The $f_{1,2}(x_n)$ refers to the characteristic equations given in where 1 denotes being the C^+ characteristic and 2 is referring to the C^-. $x_{1,2}$ refers to the two variables, M_p and p_p.

$$f(x_{1,2}) = \begin{bmatrix} \frac{B}{A}(M_p - M_A) + p_p - p_A + \frac{fB^2 \triangle x}{(p_p+p_A)DA^2} \frac{e^s-1}{s} \left(\frac{M_p+M_A}{2} \frac{|M_p+M_A|}{2} \right) + \frac{p_p^2}{p_p+p_A}(e^s-1) \\ \frac{B}{A}(M_p - M_B) - p_p + p_B + \frac{fB^2 \triangle x}{(p_p+p_B)DA^2} \frac{e^s-1}{s} \left(\frac{M_p+M_B}{2} \frac{|M_p+M_B|}{2} \right) + \frac{p_p^2}{p_p+p_B}(e^s-1) \end{bmatrix} \tag{5.3.3}$$

5.4 Initial conditions

In order to start our iterations using the Newton's method we need to "guess" starting values. In this case we use the values at the previous timestep as our initial values. But at $t = 0$ we we need existing values of pressure and massflow. And for this we use the steady state values. An inital massflow in the pipe is equal a steady massflow. The steady massflow can either be determined from a given value, or it can be found from the boundary conditions. The initial pressure distribution is determined from the steady steady state equation described in section 4.3. The pressure at the next node can be

found by

$$p_2 = \sqrt{(p_1^2 + \frac{fB^2M^2}{DA^2} \triangle x \frac{e^s - 1}{s}) / e^s} \tag{5.4.1}$$

Pressure at our next node can be found by using the value at the previous node, and correct accoring to the slope of the pipe and the fluid properties. Hence we need an initial value of the pressure at the inlet in order to obtain the initial pressure distribution at the pipe.

6 The program

In order to solve the equations, a computer program was developed. Much of the same was done by Grafsrønningen[9] in 2006, however the new program was written from scratch but involved many of the same functions as the previous program. The previous program however was written to determine startup and shut-down of a 250 km slope from a reservoir in Heidrun to the gas processing plant on Tjeldbergodden at the mainland. This program was written in order to solve gas transport for medium length pipelines, and testing it up against the existing testresults for the section etween Kårstø and Vestre Bokn.

Assumptions made in the code

As listed in chapter 4, some assumptions were made when deriving the characteristc equations. Point 1 and 4 stated that both compressibility factor and temperature were held constant, and since wavespeed is defined as $B = \sqrt{ZRT}$, it also becomes constant. In reality, both temperature and compressibility changes over the pipeline. Measurements from the test section suggest that the temperature of the gas decreases approximately $\sim 14 - 15$ degrees Celcius over the 12 km from Kårstø to Bokn. Section5.4 also shows that the pressure does not remain constant throughout the pipeline. This will in reality result in a change in both viscosity and compressibliltyfactor.

A comparison of wavespeed and compressibilityfactor at initial conditions shows

6.1 Natural gas compressibility

The compressibility factor is a factor describing the behavioural deviation of the natural gas from the ideal gas law. Under large pressure and extreme temperatures the gas no longer acts according to the law of an ideal gas, and the introduction of the compressibility factor is a way of dealing with the behavioural deviation from the ideal gas law

$$p = \rho RT \tag{6.1.1}$$

Other, more complex numerical models dealing with gas transport often uses a complicated state equation, i.e TGNet[12]. These equations can be difficult to solve. For the method of characteristics a single variable called the compressibility factor, Z, is used.

$$p = Z\rho RT \tag{6.1.2}$$

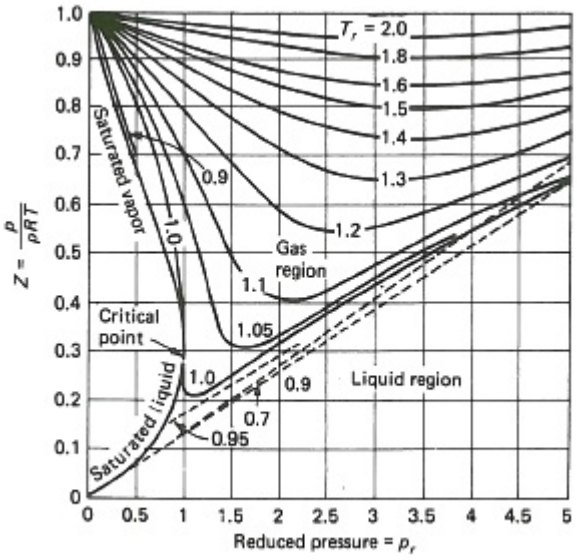


Figure 6.1.1: Compressibility chart[23]

Rewriting 6.1.2, shows that the compressibilityfactor can be written as sa function of pressure and temperature. The pressure and temperatures used are the reduced pressure and temperature. These values are given as $p_r = {}^{p}/_{pcr}$ and $T_r = {}^{T}/_{T_{cr}}$. T_r and p_p denotes the reduced temperature and pressure, and T_{cr} and p_{cr} represents the critical values. Obtaining a critical value for a specific gas is a rather complex procedure, and for this calculation values for methane gas are used. This is likely to deviate some from the real answer, but since the gas contains approximately 90% methane gas this is considered good enough.

The computational method used to approximate the Standing Katz compressibility chart, is the method introduced by Gopal[8]. This method provides rather well approximations when the values when $T_r \neq 1.05$ and $1.0 < p_r < 1.5$ [15].

6.2 Viscosity of natural gas

In order to calculate the viscosity of natural gas, a function *visosity.m* has been developed. Lee Gonzalez and Eaking presented as early as 1966 a method to calculate the viscosity of natural gases[13]. This method has been usend to a large extent in commercial codes, such as the TGNet[12]. Different versions of the method has been presented. The LGE method used in the code is implemented as follows:

$$X = \frac{x_1 + x_2}{T + x_3 \cdot MW_g} \tag{6.2.1}$$

$$K = \frac{(k_1 + k_2 \cdot MW_g) \cdot T^{k_3}}{k_4 + k_5 \cdot MW_g + T} \tag{6.2.2}$$

$$Y = y_1 - y_2 \cdot X \tag{6.2.3}$$

$$\mu = 10^{-4} K exp\left[X \cdot \rho^Y\right] \tag{6.2.4}$$

The unknowns are defined as

k	Value	x	Value	y	Value
k_1	9.379	x_1	3.448	y_1	2.447
k_2	0.016007	x_2	986.4	y_2	0.2224
k_3	1.5	x_3	0.010009		
k_4	209.2				
k_5	19.26				

Table 6.1: LGE correlation factors

The temperature is given in *Rankine* and the gas density is given in g/cc. The answer is given in the unit *centipose*, and must be corrected according to standard values. One centipose is equal to $0.001^{N \cdot s}/m^2$.

6.3 Friction factor

In section 2.3 we derived the friction factor for the pipeflow. This frictionfactor is a function of Reynoldsnumber, diameter and surface roughness. Since the Reynoldsnumber changes as the velocity component changes, it does not remain constant, and needs to be calculated for the different velocities. For instance, a shut down of the pipeline, will result in zero velocity at one end, whereas the other end still holds the initial velocity. A function *frictionfactor.m* were developed. This function first calculates the respective Reynoldnumber. It then determines if the flow is laminar or turbulent. If the flow is laminar, given that $Re < 2300$, the frictionfactor is calculated according to equation 2.3.6. If the flow is turbulent, $Re \geq 2300$, the function uses Colebrooks equation, and an iterative process to calculate the frictionfactor. The *frictionfactor.m* function is called first using the steady state velocity. In the iterations process it is called for each node using the old flow rate at the node.

6.4 The code

The program can mainly be divided into three sections. The first section is the Pre-processing, where we build up the problem, with all of its variables and necessary data.

The second part of the program is the main processing which consists of the solver. The third part is the post processing, here data are saved and presented.

6.4.1 Pre-processing

The program starts with a pre-processing sequence, in which one obtain and sets the necessary data and variables needed to do the calculations. The pre-processing gives the data for the gas, pipeline, initial- and boundary conditions. At first the program asks for which case to calculate, wether it's a transient or steady state case. Based on this information the program selects which documents to read. The pre-processing then follows the following sequence:

Setting universal constants: The universal constants R and g are set.

Reading data: The data necessary for the calculation is read from excel-sheets. Boundary conditions are read from excel documents created from the *Data.xls* document provided by Gassco. The pipeline profile is also read from an excel sheet based on an approximation from a figure in Langelandsvik's phd thesis[12].

Pipeline properties: The values for the pipeline are then set. Diameter is fixed, and other properties are calculated from these values and the pipeline profile.

Object: Pipeline	
Property	**Function**
.diameter	diameter of pipeline
.area	calculated area
.roughness	wall roughness
..height	heights at different nodes at the pipeline
.length	total length of pipeline under consideration
.initial_nodes	number of measuring points
.dx_initial	the length of each initial pipeline segment
.dx	$\triangle x-$length of each segment
.nodes	total number of nodes
.nodes_pr_dx	number of nodes pr initial pipeline segment
.alfa	calculated angle of inclination for the original pipeline segments
.s	$\mathrm{s}=\left(2\mathrm{g}\triangle\mathrm{x}\sin\theta/\mathrm{B}^2\right)$

Natural gas properties: Depending on the case, different initial pressures are given. The other properties of the gas can be calculated from the pressure using the equations of state given in section 4.1. Critical temperature and pressure are initially given. The section calls the functions *compressibilityfactor.m* and *viscosity.m*

Object: gas	
Property	**Function**
.p_init	Initial pressure of the gas
T_init	Initial temperature
.pc	Critical pressure of gas(methane gas used)
.Tc	Critical temperature(methane gas used)
.Tr	Reduced temperature(T_{init}/T_r)
.pr	Reduced pressure(p_{init}/p_r)
.MW	Mole weight of gas
.R	Spesific gas constant($R_{universal}/MW$)
.wavespeed	Acoustic wavespeed of gas($B = \sqrt{ZRT}$)
.Z	Compressibility factor calculated from6.1
.rho	Gas density with inital pressure and temperature
.my	Gas viscosity calculated as in6.2

Initial conditions/ steady–state: We need starting values in our grid in order to have convergence in our solution. The initial boundary condition for the mass flow rate is used as the steady state massflow in the pipeline. The pressure however must be calculated according to 4.3. In 5.4.1, p_2 refers to the pressure at the next node. We therefore need to keep track of the value for the slope of the pipeline. This is done by utlizing a node-counter for the nodes and an if sentence.

Object: Initial	
Property	**Function**
.massflow	Steady state massflow
.pressure	The calculated steady state pressure distribution

Timeevaluation: the program requires an input of the total simulation time in seconds. When the number of seconds is set, the total number of timesteps must be determined. In order to calculate the timesteps we use the Courant condition[14]. Equation 4.7.1 gives us the timestep, and by inserting a total simulation time the total number of timesteps for a given simulation time becomes $t_{total}/\triangle t$.

Boundary conditions: Depending on the case, different boundary conditions must be given. The program runs two types of cases, either from real values from real values taken from the Gassco excel file[7]. When that is the case, the inlet boundarycondition consists of a vector of massflows with a fixed time interval of one minute. In order to obtain the correct value at the given timestep, and interpolation using is carried out in the solver.m code. When the transient is an imagined case such as a sine-oscillation in massflow, or a rapid closure of a valve, only steady flow value is given,values are given and the values at the given time are calculated. The outlet boundary condition is set as the initial pressure that we calculated at the last node according to equation 5.4.1.

6.4.2 Main processing

The main processing starts by initializing a new object called fluid. This is a $[2 \times nodes]$ matrix containing the pressure and massflow. It's first values are set as the steady state massflow and the initial pressure. One are then asked if the inertial multiplier should be utilized. Two matrices to save data are pre-allocated. Since the model produces vast quantities of data, restrictions on the number of nodes and timesteps saved must be given.

Inertial multiplier

If the inertial multiplier is utilized, a method *inertial_ multiplier.m* is called. This method calculates the alpha value accoring to equation 4.5.1. With the new alpha value our time steps can be calculated according to the following

$$t_{new} = t_{old} + \alpha \cdot \triangle t \tag{6.4.1}$$

Iterations

If the inertial multiplier is not utilized, the alfa value is set equal to one. A *for-loop* iterates from 1 to the total number of timesteps that we calculated in the pre-processing.

For each timestep it calls the function *solver.m*. The function *solver.m* solves the characteristic equations according to section 4, and returns fluid object with all of the pressures and flow rates at each node.

When the inertial multiplier is utilized, the same iteration can no longer be used. The alfa value are changing throughout the flow, and a *while-loop* is used to iterate in time. The while loop calls the same function, *solver.m* as it was when α was set equal to one. but it now uses the new time as presented in 6.4.1. The alfa value as also no longer equal to one, and at the end of *solver.m*, a new function, *inertial_solver.m* is called. The *inertial_solver.m* uses both the old and new value of the fluid. It then corrects the solution at the new nodes according to equations 4.4.14 and 4.4.15.

6.4.3 Post processing

The main part of the post processing is saving the datas that we have kept from our calculations. These files are saved as .dat files and a name relating them to the case they are representing. Presenting the results in plots are also a part of the post processing.

7 Simulations

In order to make simulations of any value one need to compare the results to already measured values, only in that way one can see if the results one gets are of any value. The method of characteristics is a simplified methods to calculate gas transport in pipes. Several variables can not be examined using this method. However it may prove useful for simple applications and calculations. The excel-sheet from Gassco were used for the testcases. The hydrostatic pressureloss from Kårstø to Bokn is negliglible.

7.1 Steady state calculations

Steady state calculations of a constant flow rate of 70.6 MSm^3/d.

Measurements from Gassco			
2/4/2009 2:56:08 AM			
Inlet	Pressure	180.8623	Bar
	Massflow	70.6025	$^{MSm^3}/_{d}$
	Temperature	33.183	^oC
Outlet	Pressure	179.1083	Bar
	Temperature	19.005	^oC
Pressureloss		1.754	Bar

Table 7.1: Initial conditions

Measured inlet and outletpressure at Kårstø-Bokn

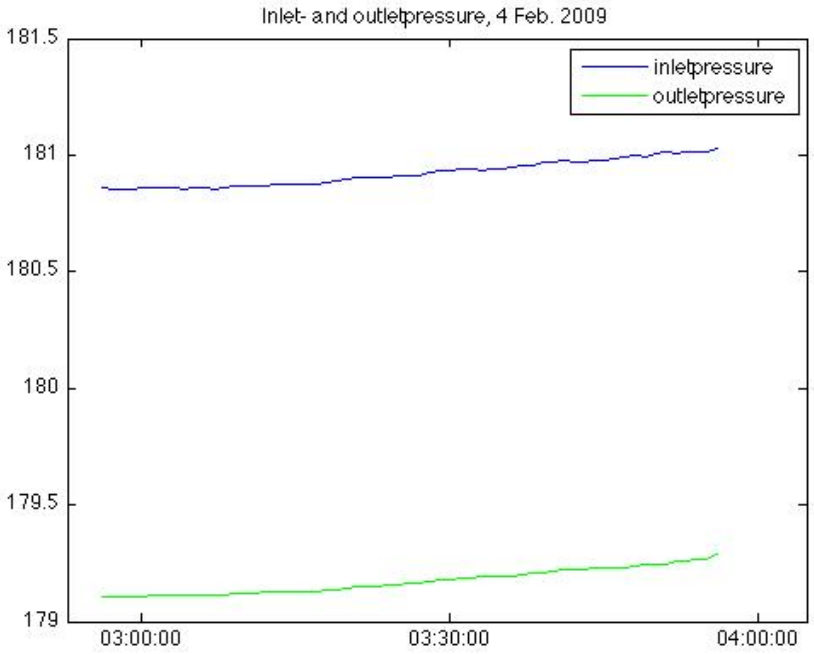


Figure 7.1.1: Inlet and outletpressure

Since the altitude at the boundaries are approximately the same, the pressureloss due to friction in the pipe can be represented as the difference between the inletpressure and outletpressure.

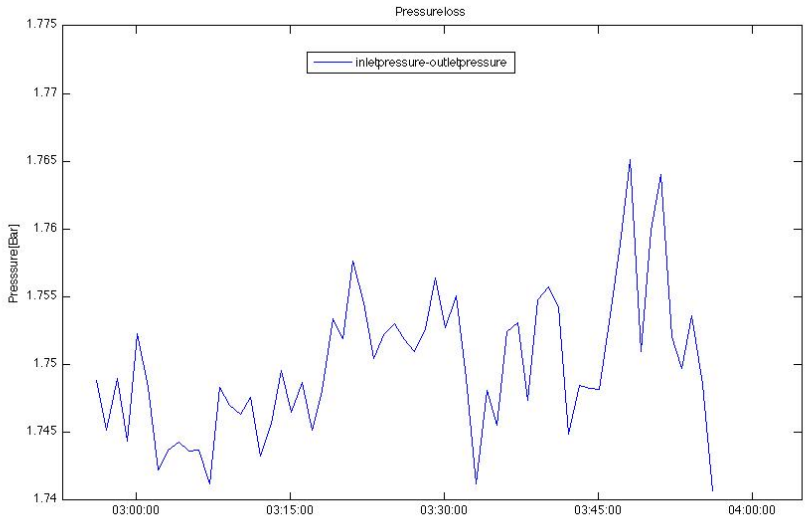


Figure 7.1.2: The corresponding pressureloss

Average friction pressureloss over the time interval shown in 7.1 is $1.7499\,bar$.
The steady state calculations gives a pressure distribution as shown below

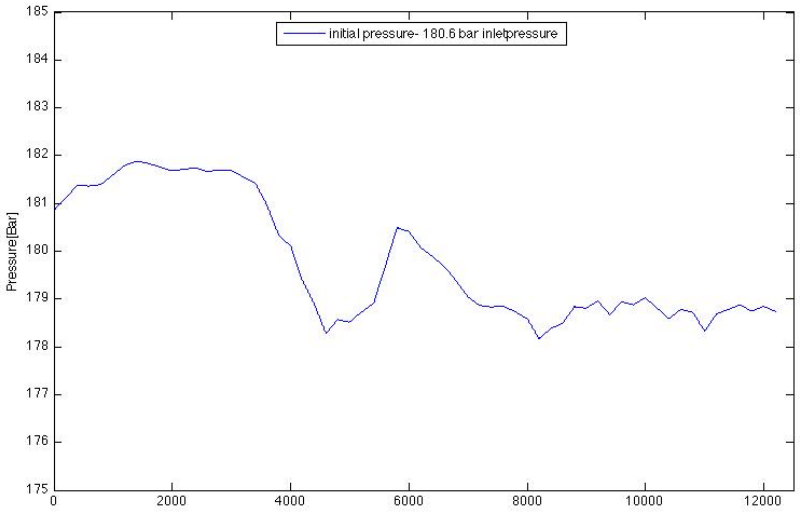


Figure 7.1.3: Initial pressure distribution

The calculations of the initial pressure distribution shows that the pressure difference between the highest and the lowest point in our pipeline is approximately $3.564\,bars$,

which corresponds rather well to given hydrostatic pressure difference for the two points. The altitude difference of about $212\,meters$, giving a hydrostatic pressure of $3,24\,bar$,

Steady state calculations			
Inlet	Pressure	180.8623	Bar
Outlet	Pressure	178.7419	Bar
Pressureloss		2.1204	Bar
Error		17.27	%

Table 7.2: Steady state results

As the results shows, the MoC tends to overpredict the pressureloss. This will be discussed further in 8.3.

7.2 Simulation of cases

In addition to the steady state calculations other cases are also discussed. These cases involves transients, $M = M(t)$. Scenarios of interest are

- Opening of a valve ("start-up")
- Closing of a valve ("shut-down")
- Flow oscillations ("altering mass flow rates")

Based on this we can define some cases. Using cases from the data-sheet from Gassco gives us results that we can compare to real values. A series of calculations will be defined:

case-number	*Steady state massflow[kg/s]*	*△M [kg]*	*Inlet Pressure [Bar]*	*Valve opening/ closing time[s]*	*Oscillation period[min]*	*Number of Nodes*	*Simulation Time[s]*
			Oscillating flow cases				
1	*623.4925*	*2.98*	*180.86*	-	-	*489*	*3600*
2	*623.4925*	*2.98*	*180.86*	-	-	*977*	*3600*
3	*623.4925*	*50*	*180.86*	-	*10*	*489*	*3600*
4	*623.4925*	*50*	*180.86*	-	*10*	*977*	*3600*
5	*623.4925*	*2.98*	*180.86*	-	-	*4942*	*90*
			Shut-down				
6	*626.44*		*185..43*	*60*	-	*489*	*3600*
7	*626.44*		*185..43*	*60*	-	*977*	*3600*
8	*626.44*		*185..43*	-	-	*489*	*1800*
9	*626.44*		*185..43*	-	-	*977*	*1800*
10	*626.44*		*185..43*	*60*	-	*4942*	*70*
			Start-up				
11	*12.861*		*147.25*	*39600*	-	*245*	*39600*

Table 7.3: Test scenarios

Cases 5 and 6 were used as comparison in order to validate the accuracy of our results using less nodes.

7.2.1 Oscillating flow

Even though we often refer to the steady state case4.3, the reality is that our case is not a steady state but time dependent. We have small changes in the massflow and pressure at the inlet. An increase in the massflow will increase the compressibility, and as a result the pressure increases.

Two scenarios with oscillating massflow were simulated. One based on a real life example from the data received from Gassco, using the measured values for the given values in[7] as a reference. The calculated pressures from the program were compared to the pressure measurements made by Gassco. The other case is a simulated case with larger oscillations in the flow. This simulation had no reference and is only an indication of how pressure and massflow relates to one another when the magnitudes of the oscillations becomes large.

7.2.2 Shut down/ valve closing

Another interesting case is the event of a valve closing at one end. Closing a valve will reduce the mass fluxt and as a consequence the pressure also drops. Depending on the speed of the shut down different problems emerges. In case of a rapid shut down, large waves are likely form inside the pipe.

Two cases were also investigated for this event. One rapid shut down from maximum massflow to complete blockage in 1 minute. The other case involved a partly stepwise reduction of massflow, not completely blocking the valve. The second case is also a case from Gassco.

7.2.3 Start-up/ valve opening

Another case of transient flow is a startup of a flow in a closed pipeline. The startup takes place for a long time period. In the case evaluated using the program, the startup process happened over a period of 11 hours, the equivalent of 39600 seconds, resulting in a huge number of timesteps.

8 Results

8.1 Oscillating flow

8.1.1 Case 1 and 2

The first case of the oscillating flow is a steady-state case from real life measurements. The steady state in this matter is however, not exactly correct. The flow rate at the inlet of the pipeline has small oscillations, and corresponding pressure changes. The real life case is presented below

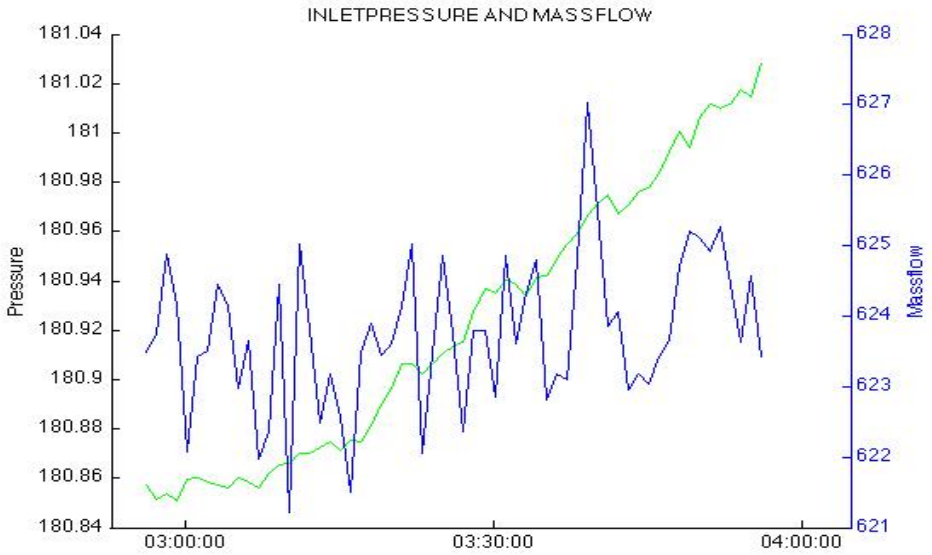


Figure 8.1.1: Measured pressure and massflow at inlet

As we can see from the case above pressure and massflow are not very well correlated. The inletpressure increases over our calculation time. A closer look at figure 7.1 reveals that the outletpresures is also increasing in that case, and inlet and outletpressure follows one another rather well. In our case we have a constant outletpressure, and a simulation of the same case is presented in figure 8.1.3.

In order to determine sufficient number of nodes a simulation using 500, 1000 and 5000 nodes were done for the unsteady flow case for 90 sec. The simulation using 5000 nodes

were not carried out for the entire time of 1 hour, due to computational time. The difference between 5000 nodes and 500 and 1000 nodes after 60 seconds is presented below

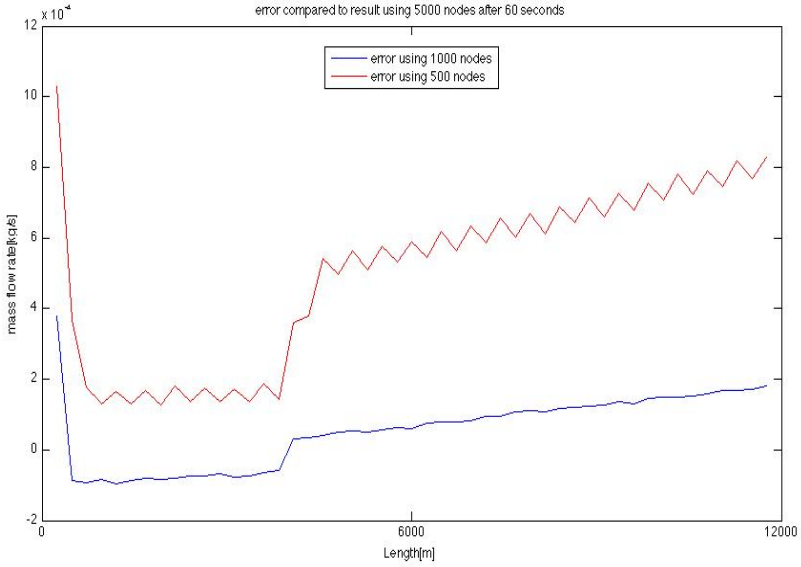


Figure 8.1.2: Comparison 500, 1000 and 5000 nodes after 60 seconds

From figure **8.1.1** we see that both using 500 and 1000 nodes rather good results. Mean error between 1000 nodes and 5000 nodes is equal to $5.4426 \cdot 10^{-5}$. The case using 1000 nodes is used for the presentation below.

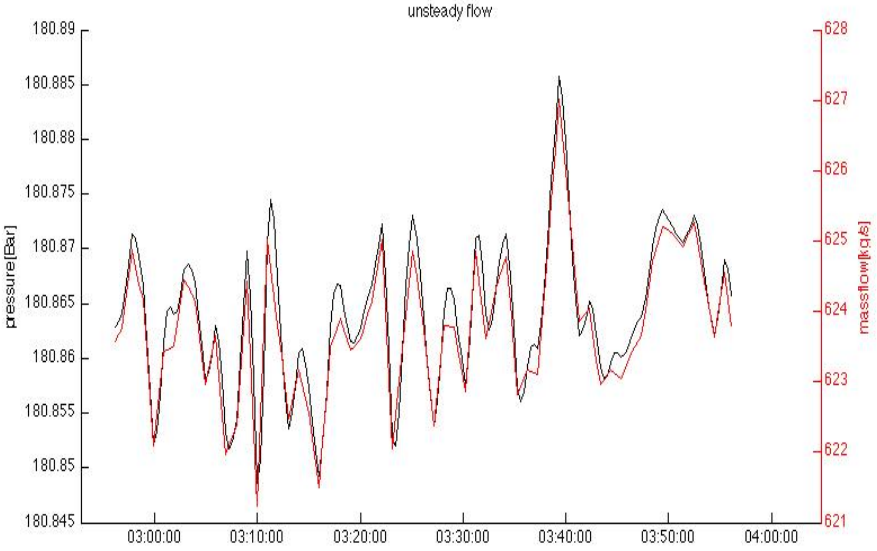


Figure 8.1.3: Case2: Pressure and massflow at the inlet

As we clearly see from the resulting plot, massflow and pressure are strongly correlated. However the results are not correct according to the measured quantities. Reasons for this will be further discussed in section 8.3. The measured results compared to the calculated pressure at the inlet is presented in figure below

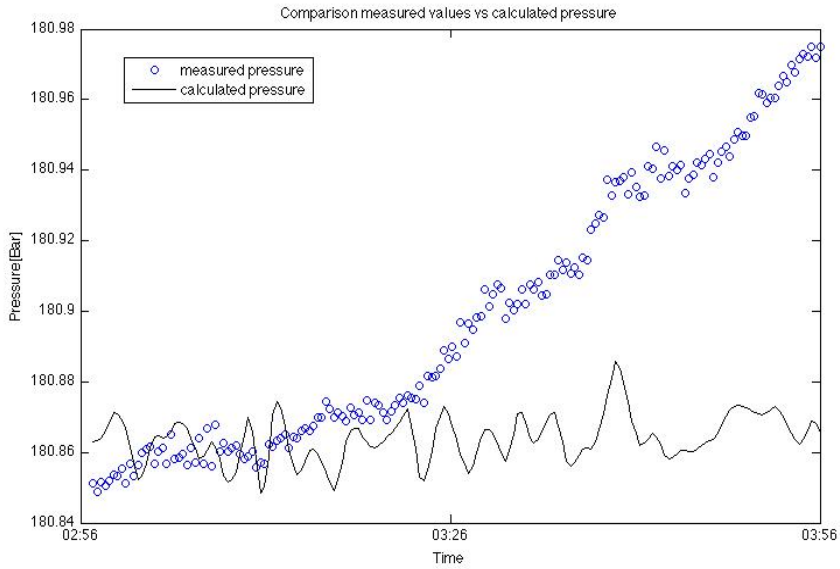


Figure 8.1.4: Case2 : Comparison to measured values

8.1.2 Case 3 and 4

A testcase with larger oscillations were also evaluated, using a sine-function representing steady oscillations. Magnitude of the oscillations were 50 $^{kg}/_{s}$, and the oscillation period was set to 10 minutes.

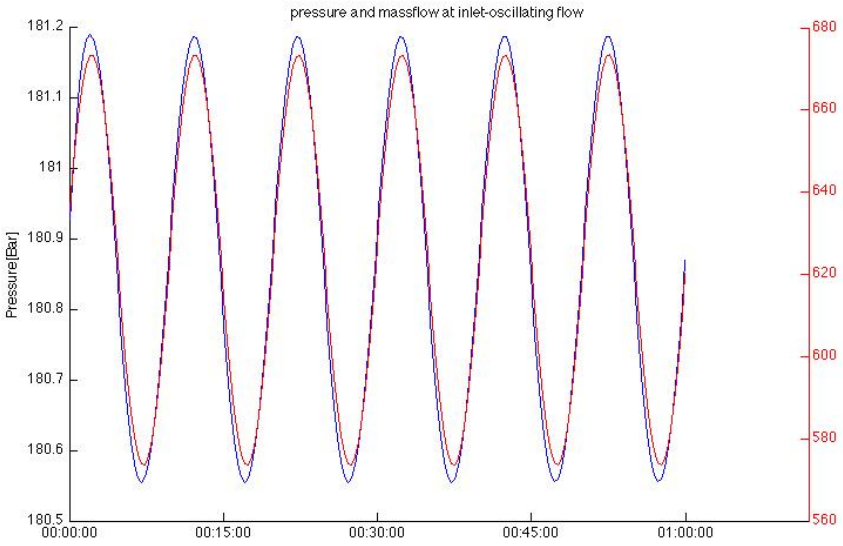


Figure 8.1.5: Case4: Steady oscillations

8.2 Shut-down

8.2.1 Complete shut-down

The first of the shut-down cases involves shutting down the valve completely in a very short time. The valve closing time in this case was set to be minute, or 60 seconds. This is a very rapid shut-down and in reality a complete shut down this quickly would not occur. However, it can be of interest to see what happens with both pressure and massflow when the valve closes this quickly. Intuitively one sees that the pressure must drop quickly, and as a result of this, backflow might occur as will be shown in the figures. A comparison of numer of node was also done for the shut down case.

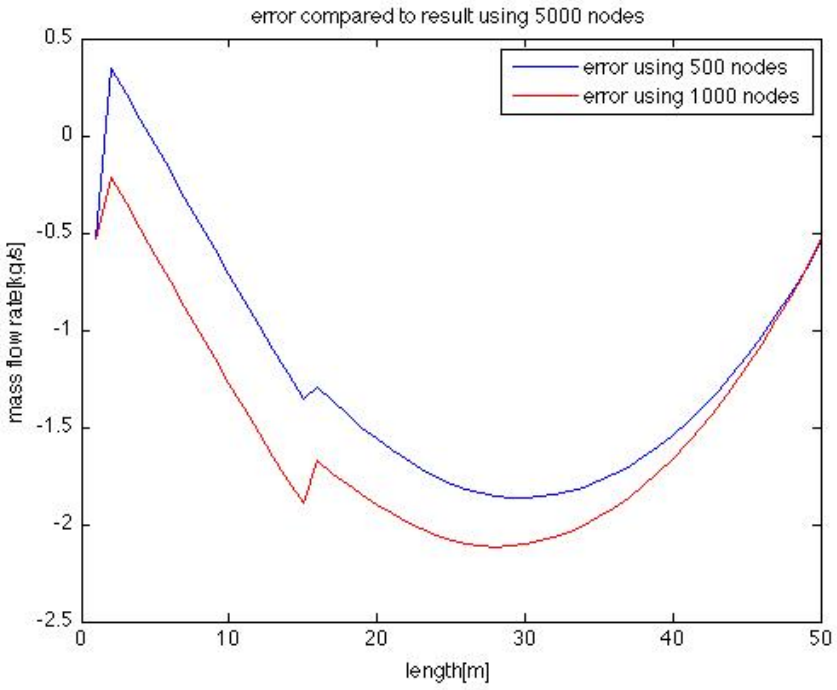


Figure 8.2.1: Error vs 5000 nodes solution after 60 seconds

The average error was $-1.211\,{}^{kg}/_{s}$ using 500 nodes, and $-1.505\,{}^{kg}/_{s}$. This shows that the solution using 500 nodes is more accurate than using 1000 nodes. This contradicts the fact that our solution becomes more accurate as the number of nodes increase. However, both cases provides accurate results with an average of less than one percentage error, provided that the results at the first node are not considered since mass flow rate at this point is close to 3or equal to zero. The case using 1000 nodes will be used in the further presentations.

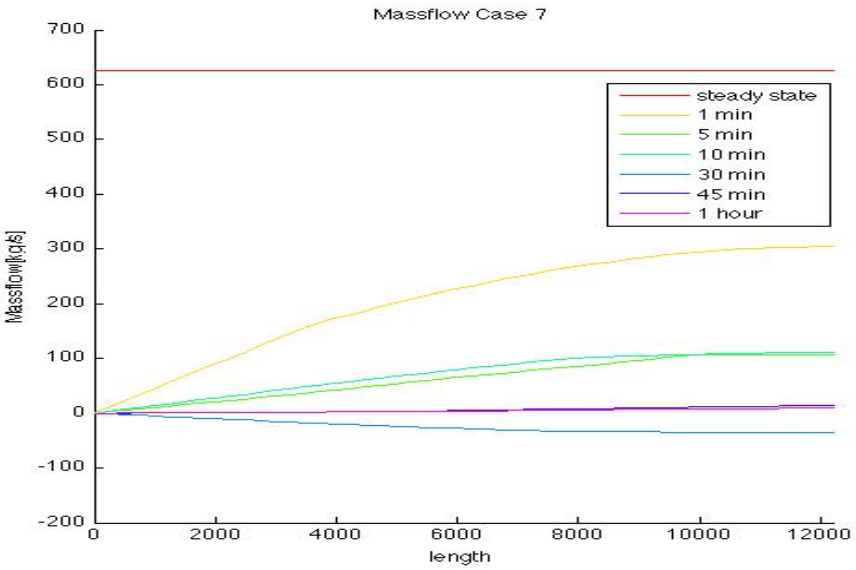


Figure 8.2.2: Case 7: Massflow

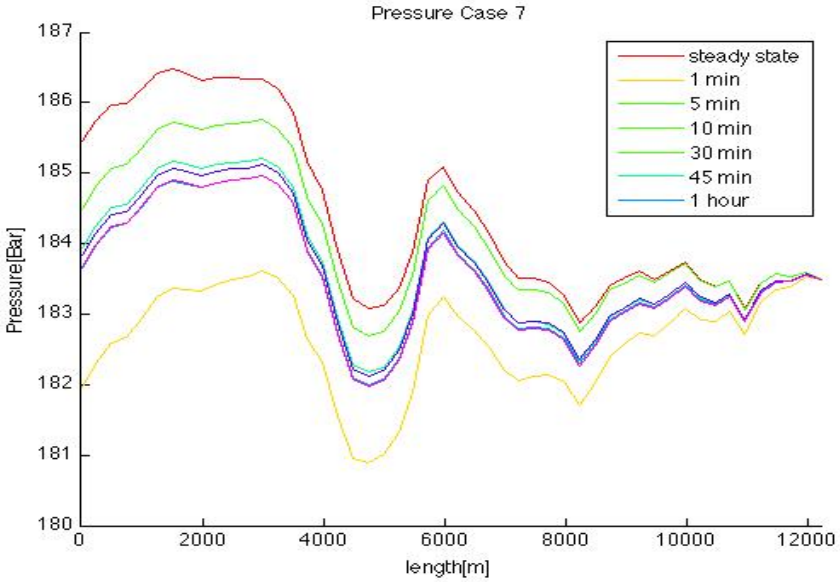


Figure 8.2.3: Case 7: Pressure

A closer look at the figures above reveals that we have oscillations in our flow. These oscillations decrease with time, and eventually die out, with a steady state mass flow rate equal to zero. The damping is a result of friction in the pipeline.

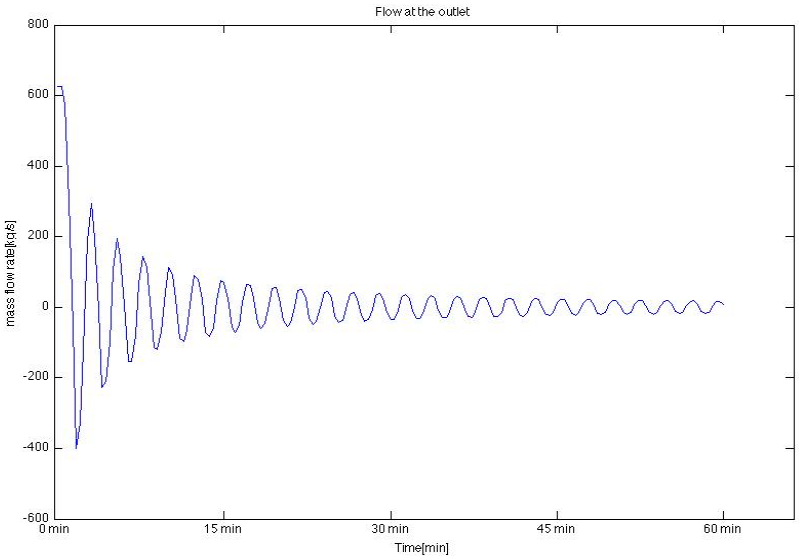


Figure 8.2.4: Flow oscillations at the outlet

8.2.2 Stepwise shut-down

A case from the datasheet[7] was also simulated. This was a case where the mass flow in the pipeline was decreased from 70.85 $^{MSm^3}/_{d}$ to 8.85 $^{MSm^3}/_{d}$ stepwise over a period of 30 minutes.

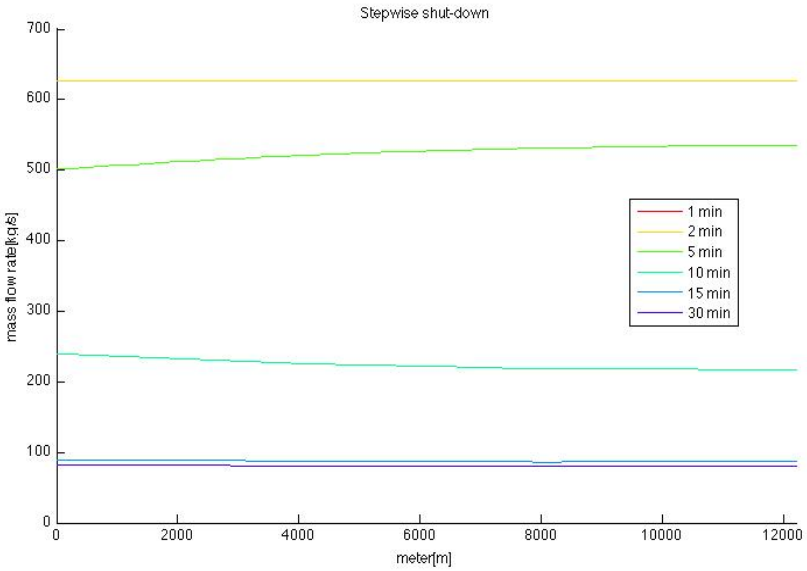


Figure 8.2.5: Case 9: Massflow

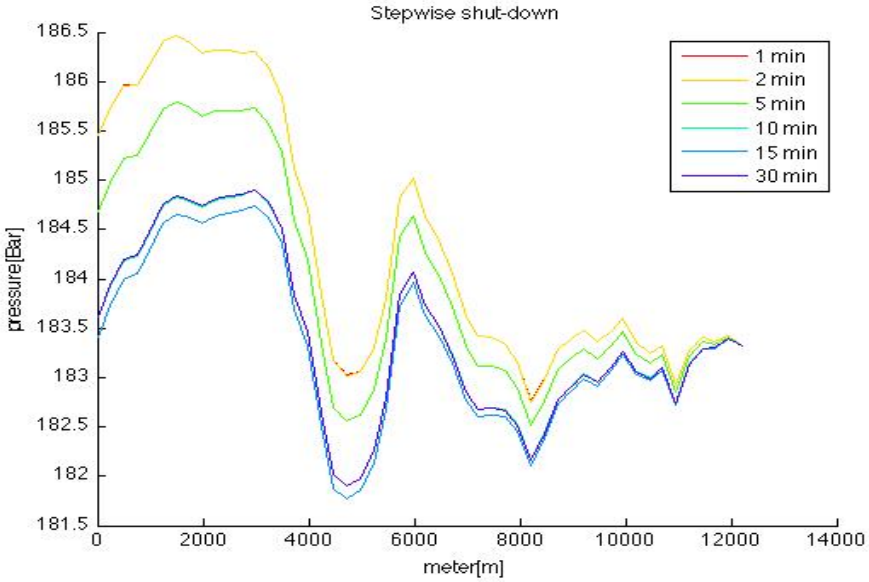


Figure 8.2.6: Case 9: Pressure

A comparison of the pressure at the inlet was done with the measured data for the same case

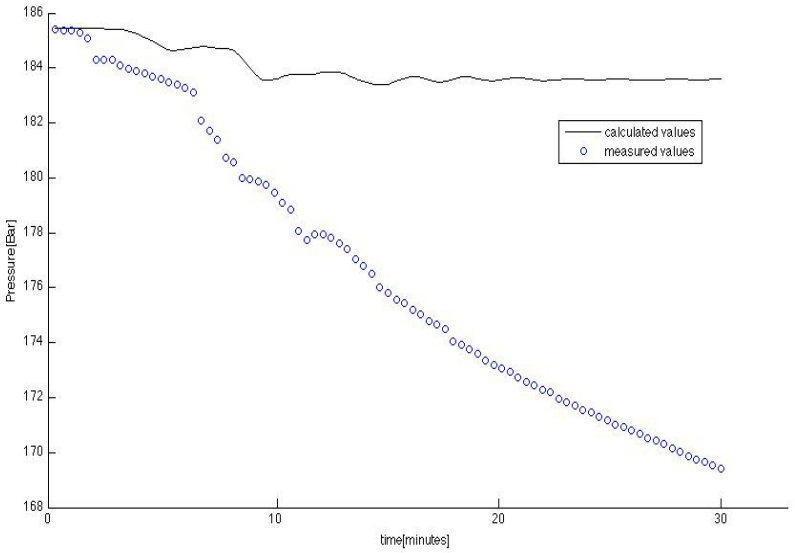


Figure 8.2.7: Comparison measured results and calculated

8.2.3 Startup/valve opening

A simulation of the start up process was done using 250 nodes. The simulation was terminated after 4 hours because of time.

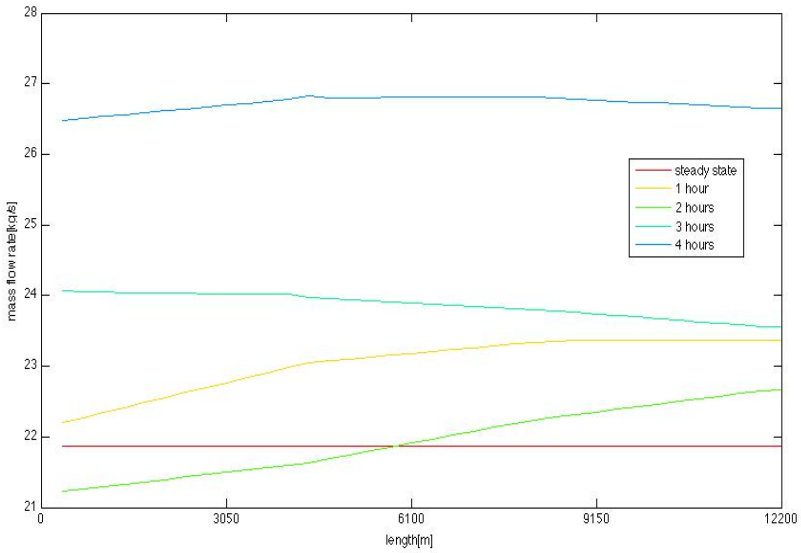


Figure 8.2.8: Case 11: Massflow

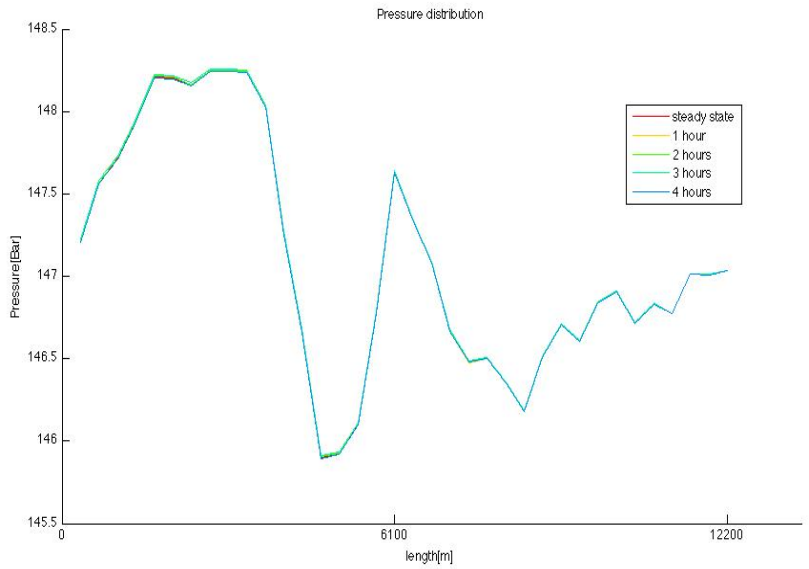


Figure 8.2.9: Case 11: Pressure

From the figure above, we see that the changes pressure is not significant. A comparison of the calculated result compared to measured values at Bokn is done

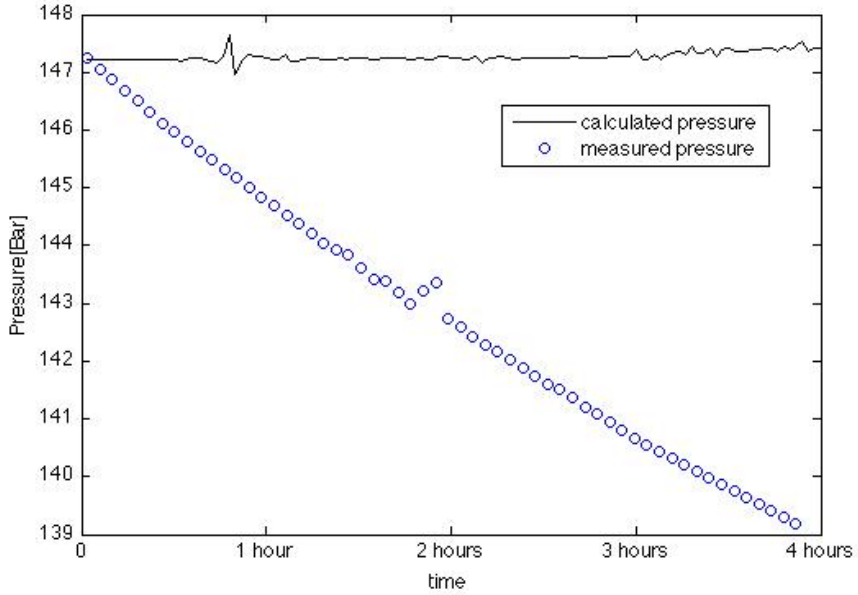


Figure 8.2.10: Comparison case 11: inlet pressure

8.3 Discussion of the results

Figures 8.1.1 and 8.2.1 was done in order to validate the results using 500 and 1000 nodes. Numerical methods such as the Method of Characteristics has the property that it becomes more accurate as the mesh becomes finer. However there must be a equilibrium between accuracy and cost, which in this case is computation time. Both figures 8.1.1 and 8.2.1 shows that results using 500 and 1000 nodes for the simulation provides sufficiently accurate results. Some of the error in figure 8.2.1, might be a result of a difference in the timesteps saved, since the program tries to save each 20 seconds, but the actual time might deviate some.

Figures 8.1.3 and 8.1.5 shows the measured values of pressure and flow rates at the inlet. Flow rate and pressure seems to correlate well. This is also supported by simulations done by Langelandsvik [12] using TGNet. It can also be explained physically, since increased flow rate will increase densitey and as a consequence also pressure. Compared to the measured flow rate and pressure at the inlet for the same time frame in figure 8.1.1, we see that the results of the measured values does not follow eachother.

A comparison of the measured values and the calculated pressure is given in figure 8.1.4. From this we see that the measured values does not correspond to the calculated values. The actual pressure increases, whereas our calculations suggests that the pressure more or less will remain constant, with minor deviations from the average. The reason for this can be found in the fact that our boundary condition at the outlet will become very wrong as time passes. In reality the pipeline under evaluation is 658 km long, giving an additional 646 km of pipeline after Bokn, that functions as our outlet boundary. Following the result that we proved when deriving the equations, that the information travels through the pipeline at the acoustic wavespeed. A wavespeed in the order of 340 m/s, means that the information at the inlet of the pipeline does not arrive at the end of the pipeline until 32 minutes have passed. What happens at the end is unknown, but a reflection wave will start to travel in the opposite direction. This wave will use approximately 33 minutes before arriving at the inlet boundary, since it travel at the speed of sound minus the velocity of the flow. As a result oft this, we in fact have to wait for more than on hour in order to get response from changes done at the inlet. Using only the length from Kårstø to Bokn means that the response from will arrive only a minute later. from a boundary condition that is physically wrong, since a reduction in mass flux at Bokn would result in a corresponding reduction of the pressure.

A closure of the valve in 60 seconds produces physical reasonable results. According to figure 8.2.2 , it is seen that after 60 seconds the mass flux at the outlet is also decreasing. A closer look at the flow-rate curve after 1 minute shows that we have a contour of a wave at about 3000 metres in the pipeline. This wave comes as a reflection from the outlet boundary, and compared to wavespeed and time the distance of the discontinuity agrees well with its location. Comparing this plot to the pressure curve in figure 8.2.3 we see that the lowest pressure at the inlet is also after 1 minute. We then have a positive pressure gradient from inlet to outlet, and a negative flow could be expected.

The mass flux at the outlet will start to oscillate. The oscillating flow at the outlet boundary is shown in figure 8.2.4. As we can see from the curve, the magnitude of the flow is decreasing in time. This damping can pe physically explained as a result of the wall friction, and the flow will eventually reach steady state, with zero mass flux.

As we can see from figures 8.2.5 and 8.2.6, the stepwise closure of the valve gives a more stable flow, similar to the steady state solutions. The large oscillations that we saw when the pipeline was completely shut down in a short time interval does not occur. Comparing the result of the calculated inletpressure for the stepwise shutdown of the pipeline in cases 8 and 9, shows that the results once again are completely erroneous when compared to measured vaules. Figure 8.2.7 reveals that as the calculated values tends to become stabile, the measured values continues to decrease. As explained earlier, this results is due to the insufficient boundary conditions at the outlet. A decrease in flow rate will naturallly result in a lower pressure. This is however not reflected in the outlet boundary condition, and it affects he solution at the inlet.

When we have the opening of the valve we see that the flowrate is increased in steps over a long period of time. Only results by the hour are presented in figures 8.2.8 and 8.2.9. We see from this that we have tendencys of flow oscillations inside of the pipeline, due to the fluctuating mass flux. The contours of several relfection waves can also be seen. Pressure shows more or less no change. The pressure tends to be scimilar to the steady state pressure, since the gradients are low.

When comparing the results of our calculations to the real values in figure 8.2.10, it can be seen that the measured values contradicts our previous results and comparisons. An increase in the masflow, should intuitively result in a higher pressure. However this is not the case. A further investigation of the measured values, and imposed boundary conditions, reveals that the valve had been closed for about an hour and half. What happens at the end of the pipeline remains unknown, and our results is of more or less no value.

Time

Calculations on long pipelines, or with a high number of nodes are very costly in computer resources and time. Since the simulations were performed on a laptop computer, the long transients with numerous nodes were in practice impossible to calculate. Using the stopwatch, *tic/toc*, function in matlab one can find the expected calculation time. An approximate computation time for different cases is presented in table 8.3.

Nodes	s	T_{total}
500	3600	$1.5\,h$
1000	3600	$5.3\,h$
5000	70	$2.5\,h$
5000	3600	$135\,h$

Table 8.1: Time evaluation

Using 500 and 1000 nodes to do the computations is realizable. Increasing the number of nodes, increases the computiation time by square. The simulations using 5000 nodes, and one hour simulation time takes about five and a half days to perform. These simulations were done on a 2 year old Macbook with a 2.4 GHz intel core 2 Duo processor, and 2 Gigabytes 667 MHz SDRAM. A new stationary more powerful personal computer would manage these operations faster. The Energy and Process department also have computer clusters available. Weither or not these would be capable of doing the calculations using matlabscripts were not investigated.

Assumptions when deriving the MoC

When we derived the method of characteristics in section 4, we made a few assumptions in order to simplifying the equations. The most significant assumption that we made, was that flow was considered isotherma. From the equation of stat it's seen that ρ and p , are temperature dependent. The measured temperature from the inlet to the outlet decreases by approximately 14^oC. An analysis comparing the resulting wavespeed and compressibility factor for the different temperatures shows.

Variable	min	max	$\triangle$	%
Temperature, T	$291.9494°K$	$306.15°K$	$14.2°K$	4.46
Compressibliltyfactor Z	0.7930	0.8207	0.0277	3.38
Wavespeed, B	$326.9151^m/_s$	$340.5661^m/_s$	$13.651^m/_s$	4.01

Table 8.2: Errors due to isothermal assumption

As we can see from the table above, the errors of compressibility, wavespeed and temperature are in the same order of magnitude. An exponential funtion to represent the temperaturedistribution, instead of solving the energy equation, was developed and tested in the program. This gave new values to Z, ρ, B and μ at the different nodes. The temperature function is given in equation 8.3.1

$$T(x) = 306.36 \cdot exp\left(-3..893 \cdot 10^{-6} \cdot x\right) \tag{8.3.1}$$

This method however, proved unstable. Ths method was therefore ignored, and temperature were considered constant throughout the pipeline. This assumption will also be one source of errors when comparing the results to measured values.

Boundary conditions

Two boundary conditions must be given according to section 4.6. One at each of the boundaries. For the tests and simulations performed in this paper, flow rate at the inlet and the calculated initialpressure at the outlet were used as boundary conditions. As proved in section 7.1, the steady state equation tends to overpredicted the pressureloss. The pressure at the outlet is in reality is higher than calculated pressure.

The flow rates and inletpressure provided by Gassco were used as the boundary and initial conditions. Gassco have Paroscientific digiquartz pressure transmitters[12] at each end of the testsection to read the pressure. In this way, using the measured pressure at the outlet could in theory be used as a boundary condition. However the measured values are not correct according to the equations, and it is therefore not ideal to impose boundary conditions that contradicts the equations.

In general the main problem for this case is that the boundary conditions are only known at one end, and using a constant boundary condition at the other end provides erroneous results at the inlet.

A solution to this can be to evaluate the time that we find the outletboundary depending on the time that we are considering. I.e. if we are simulating for 10 minutes, an acoustic wave will at the same time travel approximately 200 km. If the boundary conditions at the outlet is unknown at the time, outletboundary can be placed 200++ km away. The conditions at the outlet boundary will then not affect our solution at the inlet, and the response due to the transient will be found at the inlet.

Other cases that could be of interest, where we would have valid boundary conditions could be

- Steady state with a closed valve: Flow rate at inlet would then be equal to zero, a reduction of pressure at the outlet would then result in a mass flux.
- Pressure control in order to have constant mass flux.

Initial conditions

For our initial conditions we use the steady state values, giving a constant mass flux, and the steady state pressue. In reality we already have an unknown unsteady flow inside of the pipeline. This could however be solved if we did the simulation for the period that the acousticwave spends in order to travel from inlet to outlet. And from this time start

saving our result. In this way we could make sure that the initial conditions are correct accourting to our model.

Inertial multiplier

The inertial multiplier has been presented in the derivation of the equations. The use inertial multiplier is done in order to increase the timesteps which is useful since the calculations of longe pipelines is very timeconsuming. The method involves solving the characteristic equations for the inner nodes both using values from old and new timestep. The method were implemented using Newton's iteration method. Unfortunately solver using the inertial multiplier became unstable as $\alpha \geq 3$, and a solution to this problem were not found in time.

8.4 Suggestions for further work

As we have ssen above, the simplified Method of Charcteristics provides a simple, and in some situations, rather accurate solution to the problem of transient gas transport in large pipelines. However, the MoC has disadvantages. The limitations of the timestep is quite strict, this has partly been solved by [25] and [24]. However, the validity of the results using the inertial multiplier has been questioned and in some cases proved very misleading [18].

Expansion of the model

For instance, the use of the simplified equations results in a loss of accuracy. If the complete equations had been used, our result are likely to become more accurate. In addition, a characteristic solution of the energy equations could be utilized. Since the characteristic of the energy equation requires a higher number of nodes if solved using the same timestep, the energy equation could be solved for fewer timesteps, and therefor requre less nodes. The results could be interpolated to fit the nodes for the characteristics for the momentum equation.

In the model, the critical pressure and density for methane is used. This is off course erroneous since the gas is a composition of different species, even though methane accounts for 90% of the gas. Obtaining critical values for pressure and temperature of the natural gas composition would also be helpful in improving our results.

Using a different programming language

Even though Matlab is a much used programming language for scientific computing, providing a simple and understandable environment for its users, it has the major disadvantage of being slow. That becomes clear as the computational efforts are increasing.

In general we separate between two types of programming languages, compiled and interpreted. Matlab falls under the last category. The interpreted code is not in need of a compiler and is stored the way it is written. A program is therefore changed into binary code each time it runs, whereas a compiled program is stored in its binary form and therefore has the advantage of faster execution.

Previously, nearly all scientific programs were written in the language FORTRAN. This is a rather old language, and in recent years C/C++ has become more and more popular in combination with object-oriented programming. Therefore, if a more extensive program is implemented, it should be in form of either C/C++ or FORTRAN.

Finite difference model

In order to make a more accurate and usable program, introducing a finite difference model instead should be considered. A finite difference model has the advantage that the complex set of equations would be easier to solve completely. Introducing a finite difference model would also have the advantage that it can be implemented in an implicit form, giving a more stable and faster solution since the timestep restrictions no loger would be as strict as for the MoC. Other methods such as finite element or finite volume method could also be used.

Improvements on boundary conditions

Our main source of the erroneous results when compared to measured values comes from the fact that the initial condition are unknown and that boundary condition at the outlet does not change as the flow rates change. Obtaining some kind of mathematical correlation function for our outlet pressure as the pressure, or flow rate, changes at the inlet would significantly improve the results of our calculations.

9 Conclusions

The project was meant to give a brief introduction on available computational methods computing the flow in a gas transport pipeline. A specialization in the method of characteristics, and a comparison on the method compared to other methods was done. Finally, a numerical method using the method of characteristics was implemented, computing transient flow in a natural gas pipeline. Results were then compared to real measurements done by Gassco.

This paper proves that the numerical solution of the Method of Characteristics provides realistic solutions of the partial differential equations governing the flow in a natural gas pipeline. The pure physics of the method can also be quite easily understood. The method is rather simple, compared to more extensive models such as finite difference models. But due to its simplicity, it cannot account for many of the parameters of the flow as would be the case for more extencive models.

The method itself provides physically reasonable results, but due to insufficient boundary conditions imposed at the outlet, the results does not correspond well to the measurements done by Gassco. The reason for the lack of sufficient boundary conditions is that the calculated initial pressure at the outlet is set as the outlet bounday condition. The boundary at the outlet that we are using in this case is only a part of the pipeline, and it provides no physical boundary. The use of a constant pressure at this point is therefore not correct, and in reality the pressure at this point will show the same behaviour to a change in flow rate as the inlet. This behaviour was shown in figures 8.1.3 and 8.1.5, which proved a very good correlation between pressure and flow rate. Introducing a mathematical function that correlated the outletpressure either according to massflow or pressure at the inlet would likely increase the accuracy of the calculations significantly.

Due to its simple nature, errors may also be a result of other simplification that we did when we derived the equations.

- The use of isothermal flow.
- Constant compressibility and wavespeed.
- Neglecting the convective term.
- Simplification of our pipeline profile

Even though tests have not been performed, it is likely to believe that the method will provide rather good results, when we have set boundary conditions, for instance the case of a closed valve at the inlet, and the request for a constant mass flux at the outlet.

It can also be concluded that for simulations over a long period of time on longer pipelines, the use of an explicit solution in Matlab is not ideal. The process is to time consuming, and we would be better off with an explicit solution using a compiler language, such as FORTRAN or C/C++.

As a final conclusion it can be said that the program using the method of characteristics provides reasonable results, but the main problem is the implementation of sufficient boundary conditions. With the use of constant pressure at the boundary, the results does not prove valid for the cases that we have investigated.

Bibliography

[1] Kårstø processing plant.

[2] Statistisk sentralbyrå, olje og gass.

[3] Gassco consepts, May 2010.

[4] Ove Bratland. *Pipe Flow 1: single-phase Flow assurance.* 2009.

[5] C.F Colebrook. Turbulent flow in pipes, with particular reference to the transition regime between smooth and rough pipe laws. *Institution of Civ. Eng. Journal,11:133-156, paper No. 5204.*, 1939.

[6] I. G. Currie. *Fundamental Mechanics of Fluids.* Marcel Dekker, INC, 1993.

[7] Gassco. Measured values of kårsø-bokn.

[8] V. N. Gopal. Gas z-factor equations developed for computer. *OGJ*, Aug. 8, 1977.

[9] Stig Grafsrønningen. Transient flow at shutdown/startup of haltenpipe. Master's thesis, NTNU, 2006.

[10] S.E. Haaland. Simple and explicit formulas for the friction factor in turbulent flow. *Journal of Fluids Engineering (ASME) 103 (5): 89–90*, 1983.

[11] Erwin Kreyzig. *Advanced Engineering Mathematics.* John Wiley and Sons, 1999.

[12] Leif Idar Langelandsvik. *Modeling of natural gas transport and friction Modeling of natural gas transport and friction factor for large-scale pipelines.* PhD thesis, NTNU, 2008.

[13] Gonzalez M.H. Eakin B.E. Lee, A.L. The viscosity of natural gases. *Journal of Petroleum Technology, August 1966, pp. 997-1000.*, 1966.

[14] R. Courant* K. Friedrichs" H. Lewy. Über die partiellen differenzengleichungen der mahtematischen physik. *Mathematische annalen vol 100,pages 32-74*, 1928.

[15] John McKetta. *Encyclopedia of processing and design.* Marcel Dekker, INC, 1999.

[16] L.F. Moody. Friction factors for pipe flow. *ASME Trans , vol. 66, pp. 671-684*, 1944.

[17] J. Nikuradse. Stromungsgesetze in rauhen rohren. *VDI-Forschungsheft 361. Beilage zu "Forschung auf dem Gebiete des Ingenieurwesens" Ausgabe B Band 4*, 1933.

[18] H.H. Rachford and Jr.Todd Dupont. Some applications of transient flow simulation to promote understanding the perormance of gas pipeline systems. *Pet. Eng. J. 179-186*, 1974.

[19] Osborne Reynolds. *The sub mechanics of the universe.* Cambridge Published for the Royal Society of London at the University Press, 1903.

[20] Michael E. Plesha Robert J. Witt Robert D. Cook, David S. Malkus. *Concepts and Applications of Finite Element Analysis*, volume Fourth Edition. Wiley, 2002.

[21] R. Szymkiewicz and M. Mitosek. Analysis of unsteady pipe flow using the modified finite element method. *Communications in Numerical Methods in Engineering Volume 21 Issue 4, Pages 183 - 199 Communications in Numerical Methods in Engineering Volume 21 Issue 4, Pages 183 - 199 Communications in Numerical Methods in Engineering*, 2004.

[22] H K Versteeg and W Malalasekera. *Computational Fluid Dynamics*, volume 2nd edition. Pearson Education Limited, 1995.

[23] Frank M. White. *Viscous Fluid Flow.* McGraw-Hill, 2006.

[24] E.B. Wylie and V.L. Streeter. *Fluid Transients.* FBR Press, 1983.

[25] W. Yow. *Analysis and Control of Transient Flow in Natural Gas Piping Systems.* PhD thesis, Univ. of Michigan, Ann Arbor, 1971.

[26] J. Zhou and Michael A. Adewumi. Simulation of transients in natural gas pipelines using hybrid tvd schemes. *International Journal for Numerical Methods in Fluids, Volume 32, issue 4*, 2000.

Appendix A: Matlab Code

A1 main.m

```
%   ******************************
%   * MAIN METHOD FOR NATURAL GAS *
%   * PIPELINE CALCULATION        *
%   ******************************
%
clear all
clc

%-----------------------------------------------------
%   DETERMINE CASE
%-----------------------------------------------------
inletbc.case=input('steady-state[1],oscillating flow[2],closing valve[3] or
opening valve[4]: ');

%***PRE-PROCESSING***
%-----------------------------------------------------
%   GLOBAL CONSTANTS
%   R-Universal gas constant
%   g-gravity
%-----------------------------------------------------
g=9.81;
R=8314;
%-----------------------------------------------------
%   READING DATA
%-----------------------------------------------------
% reading data for pipeline
profile=xlsread('profile_europipe_2.xls');

% reading data for the different flows
if inletbc.case==1
    inletmassflow=70.6;
elseif inletbc.case==2
    testcase1=xlsread('test1_04feb.xls');
    inletmassflow=testcase1(:,1);
elseif inletbc.case==3
    inletmassflow=xlsread('shutdown_inletmass.xls');
elseif inletbc.case==4
    inletmassflow=xlsread('startup_case.xls');
else
    disp('a case must be given')
    break
end


%-----------------------------------------------------
%   PIPELINE PROPERTIES
%-----------------------------------------------------

%   pipeline.length-        length of pipeline from K?rst?-Bokn[m]
%   pipeline.nodes-         number of nodes [#]
%   pipeline.diameter-      diameter of the pipe[m]
%   pipeline.area-          area of the pipe[m2]
%   pipeline.height-        height of the pipeline at each initial node[m]
%   pipeline.initial_nodes- number of nodes from the original pipeline
%                           profile
%   pipeline.dx-            length between the nodes;[m]
%   pipeline.alfa-          agle of the pipeline piece[degrees]
%   nodes_pr_dx-            number of nodes for each pipeline segment from
our
%                           pipeline profile input.
```

```
%   pipeline.roughness-     Pipeline roughness
%-------------------------------------------------------

%---pipeline constants--------------------------------
pipeline.diameter=1.016;
pipeline.area=0.25*pi*pipeline.diameter^2;
pipeline.roughness=5e-6;

% pipeline properties
pipeline.height=profile(:,3);
pipeline.length=profile(length(profile(:,1)),2);
pipeline.initial_nodes=profile(length(profile(:,1)),1);
total_number_of_nodes= input('the total number of wanted nodes in
calculation: ');
nodes_pr_dx=round(total_number_of_nodes/pipeline.initial_nodes);
pipeline.dx_initial=pipeline.length/pipeline.initial_nodes;
pipeline.dx=pipeline.dx_initial/nodes_pr_dx;
pipeline.nodes=nodes_pr_dx*(pipeline.initial_nodes-1)+1;
disp(sprintf('total number of pipeline nodes: %3.4g', pipeline.nodes));

for i=1:(profile(length(profile(:,1)),1)-1)
    pipeline.alfa(i)=atan((pipeline.height(i+1)-
pipeline.height(i))/pipeline.dx_initial);
end

%-------------------------------------------------------
%   NATURAL GAS PROPERTIES
%-------------------------------------------------------

%   gas.p- gas pressure[Pa]
%   gas.Tc- critical temperature gas[K]
%   gas.rho-gas density[kg/m3]
%   gas.T- gas temperature[K]
%   R- universal gas constant[J/kmol*K]
%   gas.M- molecular mass gas[g/mol]
%   gas.R- gas constant
%   gas.Z- compressibility factor
%   gas.wavespeed- acustic wavespeed
%-------------------------------------------------------
if inletbc.case==1
    gas.p_init=180.8623657*1e5;
elseif inletbc.case==2
    gas.p_init=180.8623657*1e5;
elseif inletbc.case==3
    gas.p_init=185.431121826172*1e5;
elseif inletbc.case==4
    gas.p_init=147.213424682617*1e5;
end
gas.Tc=191;
gas.pc=46.4e5;
gas.T_init=273.15+33;
gas.Tr=gas.T_init/gas.Tc;
gas.pr=gas.p_init/gas.pc;
gas.MW=18.01;
gas.R=R/gas.MW;
gas.Z=compressibilityfactor(gas.pr,gas.Tr);
gas.wavespeed=sqrt(gas.Z*gas.T_init*gas.R);
gas.rho=gas.p_init/(gas.Z*gas.R*gas.T_init);
gas.my=viscosity(gas);
disp(sprintf('the acoustic wavespeed at ICs is set to: %3.4g',
gas.wavespeed));
disp(sprintf('the initial gas density is: %3.4g', gas.rho));
%-------------------------------------------------------
```

```
%   INITIAL CONDITIONS/STEADY-STATE
%   MSM3_D- Million standard cubic metres per day
%   steady_massflow- the flowrate in the
%-----------------------------------------------------

%***steady state calculations*******
MSM3_D=inletmassflow(1);
massflow_corr=8.831;
steady_massflow=MSM3_D*massflow_corr;
disp(sprintf('the initial gas velocity: %3.4g',
(steady_massflow/(gas.rho*pipeline.area))));
pipeline.s=2*g*min(pipeline.dx)*sin(pipeline.alfa)./gas.wavespeed^2;

initial.pressure(1)=gas.p_init;
friction=frictionfactor(steady_massflow,gas,pipeline);
C=friction*gas.wavespeed^2*steady_massflow^2*pipeline.dx/(pipeline.diameter
*pipeline.area^2);

%*****case 1*****
if nodes_pr_dx==1
disp('only one node each measuring point')
    for node=2:pipeline.nodes
            s=pipeline.s(node-1);
        if s==0
            initial.pressure(node)=sqrt(initial.pressure(node-1)^2-C);
        else
            initial.pressure(node)=sqrt((initial.pressure(node-1)^2-....
                C*((exp(s)-1)./s))/exp(s));
        end
    end
else
    for node=2:pipeline.nodes
          for n=0:pipeline.initial_nodes-2
                if node>=(2+nodes_pr_dx*n) && node<=(nodes_pr_dx*(1+n))
                    s=pipeline.s(n+1);
                end
          end
        if s==0
            initial.pressure(node)=sqrt(initial.pressure(node-1)^2-C);
        else
            initial.pressure(node)=sqrt((initial.pressure(node-1)^2-....
                C*((exp(s)-1)./s))/exp(s));
        end
    end
end

%__________INITIAL PLOTS_____________
    figure(1)
    subplot(2,1,2)
    hold on

    subplot(2,1,1)
    title('PIPE PROFILE')

plot(linspace(0,pipeline.length,length(pipeline.height)),pipeline.height);
    %clearaxis([0 12500 -100 150])
    xlabel('pipeline length[m]')
    ylabel('heightprofile[m]')

    subplot(2,1,2)
    title('PLOT OF INITIAL PRESSUREDISTRIBUTION')

plot(linspace(0,pipeline.length,pipeline.nodes),(initial.pressure)/10^5)
```

```
    xlabel('meter[m]')
    ylabel('Pressure[Bar]')

    hold off
%_____________________________________

%-----------------------------------------------------
% TIME EVALUATION
%-----------------------------------------------------
dt=pipeline.dx/gas.wavespeed;

%-----min simulation time--------------------------
t_min=pipeline.length/gas.wavespeed;

%-----time evaluation in seconds---------------------
 seconds=input('insert the number of secounds you want to do iterate over:
');
timesteps=ceil(seconds*1/dt);



%-----------------------------------------------------
%   BOUNDARY CONDITION
%-----------------------------------------------------


    %  *********************
    %  *VALVE              *
    %  *opening and closing*
    %  *********************

if inletbc.case==1
    inletbc.massflow=steady_massflow;
    disp('ateady state')

elseif inletbc.case==2
        inletbc.osc_case=input('case 1/2 or case 3/4 ?[1/2]: ');
        if inletbc.osc_case==1
            inletbc.massflow=inletmassflow;
        elseif inletbc.osc_case==2
            period=input('insert oscillation period[min]: ');
            inletbc.oscperiod=1/(period*60)*2*pi;
            inletbc.oscmagnitude=input('insert the magnitude of the
oscillations[kg/s]: ');
            inletbc.massflow=steady_massflow;
        end
elseif inletbc.case==3
    inletbc.close_case=input('case6/7 or case 8/9 [1/2]: ');
    if inletbc.close_case==1
        inletbc.closingtime=input('insert valve closingtime in sec: ');
        inletbc.massflow(1)=steady_massflow;
    elseif inletbc.close_case==2
        inletbc.massflow=inletmassflow;
    end


elseif inletbc.case==4
    inletbc.massflow=inletmassflow;
end

outletbc.pressure=initial.pressure(pipeline.nodes);
outletbc.massflow=0;
```

```
mass_init=1;
if sum(inletbc.massflow)~=0
    inletbc.type='massflow is given at inlet';
else
    inletbc.type='pressure is given at inlet';
end

if outletbc.massflow ~=0
    outletbc.type='massflow is given at outlet';
else
    outletbc.type='pressure is given at outlet';
end

%-----------------------------------------------------


%***MAIN PROCESSING***

fluid.massflow=ones(1,pipeline.nodes)*steady_massflow;
fluid.pressure=initial.pressure;

%-----------------------------------------------------
%   EVALUATION OF THE INERTIAL MULTIPLIER
%-----------------------------------------------------
inertia_m=input('using inertia multiplier? [Y/N]: ','s');


%inertia_m='Y';


if inertia_m=='Y'
    %fi=input('insert pressure acceptance error, fi: ');
    fi=0.05;
    alfa=inertial_multiplier(fi,inletbc,fluid,pipeline,gas,massflow_corr);
    alfa=ceil(alfa)-1;
    %alfa=round(alfa);
else
    alfa=1;
end

%-----------------------------------------------------

%testcase1_outletpressure(1)=fluid.pressure(1);

    node_save=input('Save result for each node number: ');
    sec_save=input('save for each second number: ');
    time_save=round(sec_save/dt);
    row=round(pipeline.nodes/node_save)+1;
    column=ceil(timesteps/time_save);
    massflow_save=zeros(column,row);
    pressure_save=zeros(column,row);



     fig2=figure(2);
     title('dynamic plot of pressure and massflow')
     hold on
%          numframes=timesteps;
%          winsize=get(fig2,'position');
%          winsize(1:2)=[0 0];
%          Movie=moviein(numframes,fig2,winsize);
%          set(fig2,'NextPlot','replacechildren')
```

```
it=2;
if inertia_m~='Y'
    alfa=1;
    tic
    contd=0;
    for time=1:timesteps

        sec=time*dt;
        if mod(time,100)==0
            disp(sprintf('time in seconds: %3.4g', round(sec)));
        end


fluid=solver(fluid,pipeline,gas,inletbc,outletbc,time,nodes_pr_dx,dt,alfa,m
assflow_corr);

          fig2;
          subplot(2,1,1)

p=plot(linspace(0,pipeline.length,pipeline.nodes),fluid.massflow);
          ylabel('massflow[kg/s]')
          xlabel('meter[m]')

          subplot(2,1,2)

l=plot(linspace(0,pipeline.length,pipeline.nodes),fluid.pressure/10^5);
          xlabel('meter[m]')
          ylabel('pressure[bar]')

        %--------------------
        %   movie object
        % Movie(:,time)=getframe(fig2);
        % aviobject=addframe(aviobject,Frames);
        %--------------------------------

        if time==100
            time_eval=toc;
            disp(sprintf('elapsed time first 100 timesteps:
%3.4g',time_eval));
            disp(sprintf('approximately total time for calculation:
%3.4g',time_eval*timesteps/(100*3600)));
            while contd==0
                answer=input('do you wish to continue?[Y/N]: ','s');
                if answer=='Y' || answer=='y'
                    contd=1;
                elseif answer=='N' || answer=='n'
                    contd=2;
                    break
                else
                    answer=input('do you wish to continue?[Y/N]: ','s');
                end
            end
        if contd==2
            break
        end
        end

            if time==1
                 for iter=node_save:node_save:pipeline.nodes

                    massflow_save(1,1)=fluid.massflow(1);
                    massflow_save(1,iter/node_save+1)=fluid.massflow(iter);
```

```
                    massflow_save(1,row)=fluid.massflow(pipeline.nodes);
                    pressure_save(1,1)=fluid.pressure(1);
                    pressure_save(1,iter/node_save+1)=fluid.pressure(iter);
                    pressure_save(1,row)=fluid.pressure(pipeline.nodes);
                end


            elseif mod(time,time_save)==0

                for iter=node_save:node_save:pipeline.nodes
                    massflow_save(it,1)=fluid.massflow(1);

massflow_save(it,iter/node_save+1)=fluid.massflow(iter);
                    massflow_save(it,row)=fluid.massflow(pipeline.nodes);
                    pressure_save(it,1)=fluid.pressure(1);

pressure_save(it,iter/node_save+1)=fluid.pressure(iter);
                    pressure_save(it,row)=fluid.pressure(pipeline.nodes);

                end

                it=it+1;
            end
         pause(0.01);
        if time~=timesteps
              delete(p)
              delete(l)
         end
    end

elseif inertia_m=='Y'
    iteration=1;
    time=alfa*dt;
    while time<seconds

fluid=solver(fluid,pipeline,gas,inletbc,outletbc,time,nodes_pr_dx,dt,alfa,m
assflow_corr);

        if mod(iteration,100)==0
            disp(sprintf('time in seconds: %3.4g', round(time*dt)));

alfa=inertial_multiplier(fi,inletbc,fluid,pipeline,gas,massflow_corr);
        end

       %------plotting----------------------------------
        fig2;
       subplot(2,1,1)
       p=plot(linspace(0,pipeline.length,pipeline.nodes),fluid.massflow);
       ylabel('massflow[kg/s]')
       xlabel('meter[m]')

       subplot(2,1,2)

l=plot(linspace(0,pipeline.length,pipeline.nodes),fluid.pressure/10^5);
       xlabel('meter[m]')
       ylabel('pressure[bar]')
       %-------------------------------

            if iteration==1
                for iter=node_save:node_save:pipeline.nodes
                    massflow_save(1,1)=fluid.massflow(1);
                    massflow_save(1,iter/node_save+1)=fluid.massflow(iter);
                    massflow_save(1,row)=fluid.massflow(pipeline.nodes);
```

```
                    pressure_save(1,1)=fluid.pressure(1);
                    pressure_save(1,iter/node_save+1)=fluid.pressure(iter);
                    pressure_save(1,row)=fluid.pressure(pipeline.nodes);
                end
            elseif mod(iteration,time_save)==0
                for iter=node_save:node_save:pipeline.nodes
                    massflow_save(it,1)=fluid.massflow(1);

massflow_save(it,iter/node_save+1)=fluid.massflow(iter);
                    massflow_save(it,row)=fluid.massflow(pipeline.nodes);
                    pressure_save(it,1)=fluid.pressure(1);

pressure_save(it,iter/node_save+1)=fluid.pressure(iter);
                    pressure_save(it,row)=fluid.pressure(pipeline.nodes);
                end
            end
            pause(0.01);
            %delete(p)
            %delete(l)
              time=time+alfa*dt;
            iteration=iteration+1;

    end
end



%***POST PROCESSING***
%     massflowsaving=input('insert name of saved massflow file: ','s');
     %save testcase6_massflow_new.dat massflow_save -ascii
%     pressuresaving=input('insert name of saved pressure file: ','s')
     %save testcase6_pressure_new.dat pressure_save -ascii
%
```

A2 solver.m

```
%   *********************************
%   *The solver used to solve        *
%   *the transient problem using the *
%   *method of characteristics       *
%   *********************************

function
fluid=solver(fluid,pipeline,gas,inletbc,outletbc,time,nodes_pr_dx,dt,alfa,m
assflow_corr)

k=pipeline.nodes;

tolerance=1e-6;
maxiter=100;

A=pipeline.area;
D=pipeline.diameter;
s=pipeline.s;

B=gas.wavespeed;

newvalue=[zeros(1,k);zeros(1,k)];

iterlist=zeros(1,k);
node_count=0;
slope_count=1;
for node=1:k
    %---inlet boundary---
    if node==1

        fr=frictionfactor(fluid.massflow(1),gas,pipeline);

        C=fr*B^2*pipeline.dx/D/A^2;
        if s(1)==0
            s_tempval=1;
        else
            s_tempval=(exp(s(1))-1)/s(1);
        end

        switch inletbc.type

            case 'pressure is given at inlet'

                newvalue(2,1)=inletbc.pressure;
                M_1=fluid.massflow(1);
                M_2=fluid.massflow(2);

                P_1=newvalue(2,1);
                P_2=fluid.pressure(2);

                err=inf;
                iter1=0;


                while err>tolerance
                    newvalue(1,1)=M_1-((B/A)*(M_1-M_2)-P_1+P_2+...
                              (C/(P_1+P_2)*s_tempval*...

((M_1+M_2)/2)*abs((M_1+M_2)/2)+(P_2^2/(P_2+P_1))*...
                             (exp(s(1))-1))/(C/(P_1+P_2))*...
                             s_tempval*abs((M_1+M_2)/2));
```

```
                    err=norm(newvalue(1,1)-M_1);
                    M_1=newvalue(1,1);
                    iter1=iter1+1;
                    if iter1>maxiter
                        disp('we do not have convergence at inlet
boundary');
                        break
                    end
                            iterlist(node)=iter1;
                end


            case 'massflow is given at inlet'
                    if inletbc.case==1
                        % steady state
                        newvalue(1,1)=inletbc.massflow;

                    elseif inletbc.case==2
                        % oscillating flow
                        if inletbc.osc_case==1

inletmass=interp1(linspace(1,length(inletbc.massflow),length(inletbc.massfl
ow)),inletbc.massflow,1+time*dt/60);
                            newvalue(1,1)=inletmass*massflow_corr;
                        elseif inletbc.osc_case==2

newvalue(1,1)=inletbc.massflow+inletbc.oscmagnitude*sin(inletbc.oscperiod*t
ime*dt);
                        end
                    elseif inletbc.case==3
                        % closing valve
                        if inletbc.close_case==1
%                             for i=1:timesteps
                              if time<ceil(inletbc.closingtime/dt)
                                  newvalue(1,1)=inletbc.massflow(1)*(1-
time/ceil(inletbc.closingtime/dt));
                                  if newvalue(1,1)<0
                                      newvalue(1,1)=0;
                                  end
                              else
                                  inletbc.massflow(1)=0;
                              end

                              %newvalue(1,1)=inletbc.massflow(time);
                        elseif inletbc.close_case==2

newvalue(1,1)=massflow_corr*interp1(linspace(1,length(inletbc.massflow),len
gth(inletbc.massflow)),inletbc.massflow,1+time*dt/60);
                        end
                    elseif inletbc.case==4
                        % opening valve

newvalue(1,1)=massflow_corr*interp1(linspace(1,length(inletbc.massflow),len
gth(inletbc.massflow)),inletbc.massflow,1+time*dt/60);
                    end
                M_1=newvalue(1,1);
                M_2=fluid.massflow(2);

                P_1=fluid.pressure(1);
                P_2=fluid.pressure(2);

                err=inf;
```

```
                iter2=0;

               while err>tolerance
                    newvalue(2,1)=P_1-((B/A)*(M_1-M_2)-
P_1+P_2+C/(P_1+P_2)*...

s_tempval*(((M_1+M_2)/2)*abs((M_1+M_2)/2))+(P_2^2*(exp(s(1))-1))...
                              /(P_1+P_2))/(-1-
(C/(P_1+P_2)^2)*s_tempval*((M_1+M_2)/2)*abs((M_1+M_2)/2)-...
                              (P_2^2)/(P_1+P_2)^2*(exp(s(1))-1));

                    err=norm(newvalue(2,1)-P_1);
                    P_1=newvalue(2,1);
                    iter2=iter2+1;
                    if iter2>maxiter
                        disp('we do not have convergence at inlet
boundary');
                        break
                    end
                              iterlist(node)=iter2;
                end


        end

    %---Outlet boundary
    elseif node==k

        if s(pipeline.initial_nodes-1)==0
                s_tempval2=0;
        else
                s_tempval2=(exp(s(pipeline.initial_nodes-1))-
1)/s(pipeline.initial_nodes-1);
        end

        fr=frictionfactor(fluid.massflow(k),gas,pipeline);
        C=(fr*B^2*pipeline.dx)/(D*A^2);
        switch outletbc.type

            case 'pressure is given at outlet'
                newvalue(2,k)=outletbc.pressure;
                M_k=fluid.massflow(k);
                M_k_1=fluid.massflow(k-1);

                P_k=newvalue(2,k);
                P_k_1=fluid.pressure(k-1);
                err=inf;
                iter1=0;
                while err>tolerance
                    newvalue(1,k)=M_k-((B/A)*(M_k-M_k_1)+P_k-
P_k_1+C/(P_k+P_k_1)*...

s_tempval2*((M_k+M_k_1)/2)*abs((M_k+M_k_1)/2)+(P_k^2/....
                              (P_k+P_k_1))*(exp(s(pipeline.initial_nodes-
1))-1))/...
                              (B/A+(C/(P_k+P_k_1))*...
                              s_tempval2*abs((M_k+M_k_1)/2));
                    err=norm(newvalue(1,k)-M_k);
                    M_k=newvalue(1,k);
                    iter1=iter1+1;
                    if iter1>maxiter
                        disp('we do not have convergence at outlet
boundary');
```

```
                    break
                end
            end

        case 'massflow is given at outlet'
            newvalue(1,k)=outletbc.massflow;
            M_k=newvalue(1,k);
            M_k_1=fluid.massflow(k-1);

            P_k=fluid.pressure(k);
            P_k_1=fluid.pressure(k-1);
            err=inf;
            iter2=0;
            while err>tolerance
                newvalue(2,k)=P_k-(B/A*(M_k-M_k_1)+P_k-
P_k_1+(C/(P_k+P_k_1))*...

s_tempval2*((M_k+M_k_1)/2*abs((M_k+M_k_1)/2)+(P_k^2/....
                        (P_k_1+P_k))*(exp(s(pipeline.initial_nodes-1))-
1)))/(1-(C/(P_k+P_k_1))*...

s_tempval2*((M_k+M_k_1)/2)*abs((M_k+M_k_1)/2)+...

((2*P_k*P_k_1+P_k^2)/(P_k+P_k_1)^2)*(exp(s(pipeline.initial_nodes-1))-1));
                err=norm(newvalue(2,k)-P_k);
                P_k=newvalue(2,k);
                iter2=iter2+1;
                if iter2>maxiter
                    disp('we do not have convergence at outlet
boundary');
                    break
                end
                        iterlist(node)=iter2;
            end

    end

%---Nodes in the middle---
else
    node_count=node_count+1;

    if node_count==nodes_pr_dx
        s1=pipeline.s(slope_count);
        s2=pipeline.s(slope_count+1);
        slope_count=slope_count+1;
        node_count=0;
    else
        s1=pipeline.s(slope_count);
        s2=s1;
    end

    iter3=0;
    err=inf;
    M_p=fluid.massflow(node);
    M_a=fluid.massflow(node-1);
    M_b=fluid.massflow(node+1);
    P_p=fluid.pressure(node);
    P_a=fluid.pressure(node-1);
    P_b=fluid.pressure(node+1);
    fr=frictionfactor(M_p,gas,pipeline);
    %fb=fa;
    B=gas.wavespeed;
    C=fr*B^2*pipeline.dx/D/A^2;
```

```
        if s1==0
            slope1=1;                           %slope-1 is horizontal
        else
            slope1=(exp(s1)-1)/s1;%slope-1 is non-horizontal
        end

        if s2==0
            slope2=1;                           %slope is horizontal
        else
            slope2=(exp(s2)-1)/s2;              %slope is non-horizontal
        end
         while err>tolerance
            f_xn(1,1)=(B/A)*(M_p-M_a)+P_p-P_a+C/(P_p+P_a)*slope1*...
                    ((M_p+M_a)/2)*abs((M_p+M_a)/2)+...
                    (P_p^2/(P_p+P_a))*(exp(s1)-1);

            f_xn(2,1)=(B/A)*(M_p-M_b)-P_p+P_b+C/(P_p+P_b)*slope2*...
                    ((M_p+M_b)/2*abs((M_p+M_b)/2))+...
                    (P_b^2/(P_p+P_b)*(exp(s2)-1));

            J(1,1)=(B/A)+C/(P_a+P_p)*slope1*abs((M_p+M_a)/2);
            J(1,2)=1-C/(P_a+P_p)*slope1*((M_p-M_a)/2)*abs((M_p+M_a)/2)...
                    +((exp(s1)-1)*(2*P_p*P_a+P_p^2))/(P_p+P_a)^2;
            J(2,1)=(B/A)+C/(P_p+P_b)*slope2*abs((M_p+M_b)/2);
            J(2,2)=-1-C/(P_p+P_b)*slope2*((M_p-M_a)/2)*abs((M_p+M_b)/2)...
                    -(P_b^2/(P_p+P_b)^2)*(exp(s2)-1);
             iter3=iter3+1;

            newvalue(:,node)=[M_p;P_p]-J\f_xn;
            err=norm(newvalue(:,node)-[M_p;P_p]);

            M_p=newvalue(1,node);
            P_p=newvalue(2,node);

            if iter3>maxiter
                disp(sprintf('this did not converge in node: %d',node));
                break
            end
            iterlist(node)=iter3;

         end
    end
end


%-------------------------------------------------------------------------
%   CORRECTING THE VALUES ACCORING TO THE INTERTIAL MULTIPLIER
%-------------------------------------------------------------------------
if alfa~=1
%***Correction of values***

newvalue=inertia_solver(newvalue,fluid,pipeline,gas,nodes_pr_dx,alfa);

fluid.massflow=newvalue(1,:);
fluid.pressure=newvalue(2,:);
else
%-------------------------------------------------------------------------
%  Asssigning the new value to fluid
%-------------------------------------------------------------------------

fluid.massflow=newvalue(1,:);
fluid.pressure=newvalue(2,:);
```

```
end
%-------------------------------------------------------------------------
%   function end
%-------------------------------------------------------------------------
end
```

A3 inertial_solver.m

```
%   ************************************
%   *SOLVER USING THE INERTIA MULTIPLIER*
%   ************************************


function
new_newvalue=inertia_solver(newvalue,fluid,pipeline,gas,nodes_pr_dx,alfa)

k=pipeline.nodes;

tolerance=1e-6;
maxiter=100;

node_count=0;
slope_count=1;

C1=alfa*(gas.wavespeed/pipeline.area);
C2=(gas.wavespeed/2/pipeline.area)*(1/alfa-alfa);

%---Solving at the inner nodes---
for node=2:k-1
         node_count=node_count+1;
         if node_count==nodes_pr_dx
            s1=pipeline.s(slope_count);
            s2=pipeline.s(slope_count+1);
            slope_count=slope_count+1;
            node_count=0;
         else
            s1=pipeline.s(slope_count);
            s2=s1;
         end

        err=inf;
        M_c=fluid.massflow(node);
        M_p=newvalue(1,node);
        M_a=fluid.massflow(node-1);
        M_r=newvalue(1,node-1);
        M_b=fluid.massflow(node+1);
        M_s=newvalue(1,node+1);
        P_p=newvalue(2,node);
        P_a=fluid.pressure(node-1);
        P_b=fluid.pressure(node+1);

        fric=frictionfactor(M_p,gas,pipeline);
        B=gas.wavespeed;

Const=(fric*gas.wavespeed^2*pipeline.dx/(pipeline.diameter*pipeline.area^2)
);

        if s1==0
            slope1=1;                           %slope-1 is horizontal
        else
            slope1=(exp(s1)-1)/s1;%slope-1 is non-horizontal
        end

        if s2==0
            slope2=1;                           %slope is horizontal
        else
            slope2=(exp(s2)-1)/s2;
        end
        iter3=0;
```

```
        while err>tolerance

            f_xn(1,1)=C1*(M_p-M_a)+P_p-P_a+(Const/(P_p+P_a))*slope1*...
                    ((M_p+M_a)/2)*abs((M_p+M_a)/2)+...
                    (P_p^2/(P_p+P_a))*(exp(s1)-1)+C2*(M_r-M_a+M_p-M_c);

            f_xn(2,1)=C1*(M_p-M_b)-P_p+P_b+(Const/(P_p+P_b))*slope2*...
                    ((M_p+M_b)/2*abs((M_p+M_b)/2))+...
                    (P_b^2/(P_p+P_b)*(exp(s2)-1))+C2*(M_p-M_c+M_s-M_b);

            J(1,1)=C1+(Const/(P_a+P_p))*slope1*abs((M_p+M_a)/2)+C2;
            J(1,2)=1-(Const/(P_a+P_p))*slope1*((M_p-
M_a)/2)*abs((M_p+M_a)/2)...
                    +((exp(s1)-1)*(2*P_p*P_a+P_p^2))/(P_p+P_a)^2;

            J(2,1)=C1+(Const/(P_p+P_b))*slope2*abs((M_p+M_b)/2)+C2;
            J(2,2)=-1-(Const/(P_p+P_b))*slope2*((M_p-
M_a)/2)*abs((M_p+M_b)/2)...
                    -(P_b^2/(P_p+P_b)^2)*(exp(s2)-1);

             iter3=iter3+1;

            newvalue(:,node)=[M_p;P_p]-J\f_xn;
            err=norm(newvalue(:,node)-[M_p;P_p]);

            M_p=newvalue(1,node);
            P_p=newvalue(2,node);

            if iter3>maxiter
                disp(sprintf('this did not converge in node: %d',node));
                break
            end

        end
%     end
end
new_newvalue=newvalue;
new_newvalue(:,1)=newvalue(:,1);
new_newvalue(:,k)=newvalue(:,k);
end
```

A4 viscosity.m

```
%   ***************************************
%   *CALCULATION OF THE VISCOSITYCOEFFICIENT*                    *
%   ***************************************

function my=viscosity(gas)
% first doing the calculation using Sutherlads power law

% data given for methan, CH$, which is approx 90% of the content of the
% natural gas
%     To=272.8;
%     myo=1.2018e-5;
%     S=197.78;
%     T=gas.T_init;

 % my=myo*(T/To)^(3/2)*(To+S)/(T+S)


%3rd attempt
T=gas.T_init*1.8;       % temperature in Rankine
rho=gas.rho/1000;       % gas density in g/cc
MW=gas.MW;              % molecular weight of gas
k1=9.379;
k2=0.016007;
k3=1.5;
k4=209.2;
k5=19.26;
x1=3.448;
x2=986.4;
x3=0.010009;
y1=2.447;
y2=0.2224;
X=x1+x2/T+x3*MW;
K=(k1+k2*MW)*T^k3/(k4+k5*MW+T);
Y=y1-y2*X;
my=10^(-4)*K*exp(X*rho^Y)*0.001;
```

A5 frictionfactor.m

```
%   ******************************
%   * FRICTION FACTOR            *
%   ******************************


%***Colebrook***

function f=frictionfactor(M,gas,pipeline)


Re=gas.rho*(abs(M)/(gas.rho*pipeline.area))*pipeline.diameter/gas.my;

 if Re<2000 && Re>500
          f=64/Re;
     elseif Re<=500
          f=0.1;

     else

    %---Colebrook----------
    % 1/f
        err=inf;
        tol_limit=1e-6;
        old_sqrtf=30;
        while err>tol_limit
            new_sqrtf=-
2.0*log10((pipeline.roughness/pipeline.diameter)/3.7+2.51/Re*old_sqrtf);
            err=abs(new_sqrtf-old_sqrtf);
            old_sqrtf=new_sqrtf;
        end
        f=(1/new_sqrtf)^(2);
        %----------------------

end

%***Haalands equation***
%one_sqrtf=-
1.8*log10(6.9/steady.re+((pipeline.roughness/pipeline.diameter)/3.7)^1.11);
%steady.f_haaland=(1/one_sqrtf)^(2);
%----------------------
```

A6 compressibilityfactor.m

```
%   *********************************
%   * Obtaining the compressibility  *
%   * factor for the gas, using redu-*
%   * sed temperature and pressure   *
%   *********************************

%-------------------------------------
%   STANDING-KATZ COMPRESSIBILITY
%   FUNCTION
%-------------------------------------

function Z=compressibilityfactor(pr,Tr)

coeff= [1.6643 -2.2114 -0.3647 1.4385;...
    0.5222 -0.8511 -0.0364 1.0490;...
    0.1291 -0.2988  0.0007 0.9969;...
    0.0295 -0.0825  0.0009 0.9967;...
    -1.357 1.4942  4.6315 -4.7009;...
    0.1717 -0.3232 0.5869 0.1229;...
    0.0984 -0.2053 0.0621 0.8580;...
    0.0211 -0.0527 0.0127 0.9549;...
    -0.3278 0.4752 1.8223 -1.9036;...
    -0.2521 0.3871 1.6087 -1.6635;...
    -0.0284 0.0625 0.4714 -0.0011;...
    0.0041 0.0039 0.0607 0.7927];

if pr > 0.2 && pr <=1.2
    if Tr >1.05 && Tr<=1.2
        A=coeff(1,1);
        B=coeff(1,2);
        C=coeff(1,3);
        D=coeff(1,4);
    elseif Tr >1.2 && Tr<=1.4
        A=coeff(2,1);
        B=coeff(2,2);
        C=coeff(2,3);
        D=coeff(2,4);
    elseif Tr >1.4 && Tr<=2.0
        A=coeff(3,1);
        B=coeff(3,2);
        C=coeff(3,3);
        D=coeff(3,4);
    elseif Tr >2.0 && Tr<=3.0
        A=coeff(4,1);
        B=coeff(4,2);
        C=coeff(4,3);
        D=coeff(4,4);
    else return
    end
elseif pr > 1.2 && pr <=2.8
    if Tr >1.05 && Tr<=1.2
        A=coeff(5,1);
        B=coeff(5,2);
        C=coeff(5,3);
        D=coeff(5,4);
    elseif Tr >1.2 && Tr<=1.4
        A=coeff(6,1);
        B=coeff(6,2);
        C=coeff(6,3);
```

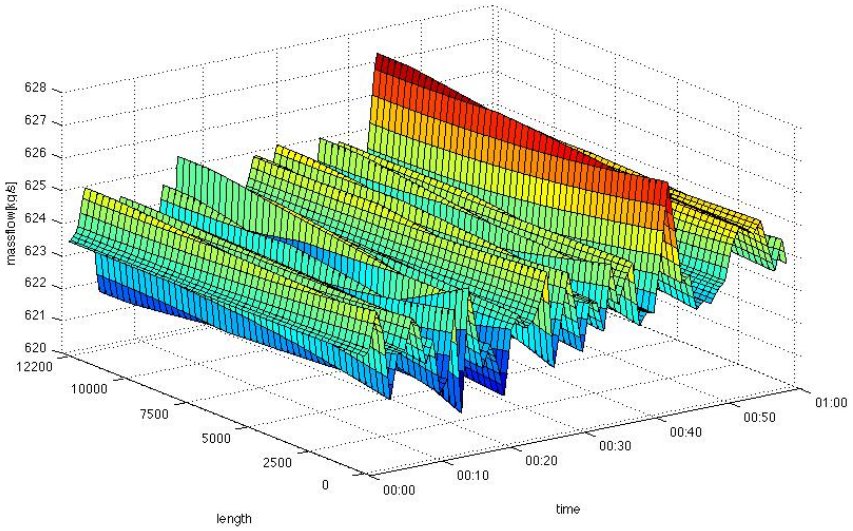
```
        D=coeff(6,4);
    elseif Tr >1.4 && Tr<=2.0
        A=coeff(7,1);
        B=coeff(7,2);
        C=coeff(7,3);
        D=coeff(7,4);
    elseif Tr >2.0 && Tr<=3.0
        A=coeff(8,1);
        B=coeff(8,2);
        C=coeff(8,3);
        D=coeff(8,4);
    else return
    end
elseif pr > 2.8 && pr <=5.4
    if Tr >1.05 && Tr<=1.2
        A=coeff(9,1);
        B=coeff(9,2);
        C=coeff(9,3);
        D=coeff(9,4);
    elseif Tr >1.2 && Tr<=1.4
        A=coeff(10,1);
        B=coeff(10,2);
        C=coeff(10,3);
        D=coeff(10,4);
    elseif Tr >1.4 && Tr<=2.0
        A=coeff(11,1);
        B=coeff(11,2);
        C=coeff(11,3);
        D=coeff(11,4);
    elseif Tr >2.0 && Tr<=3.0
        A=coeff(12,1);
        B=coeff(12,2);
        C=coeff(12,3);
        D=coeff(12,4);
    else return
    end
else return
end
   Z=pr*(A*Tr+B)+C*Tr+D;

   return
end
```

A7 inertial_multiplier.m

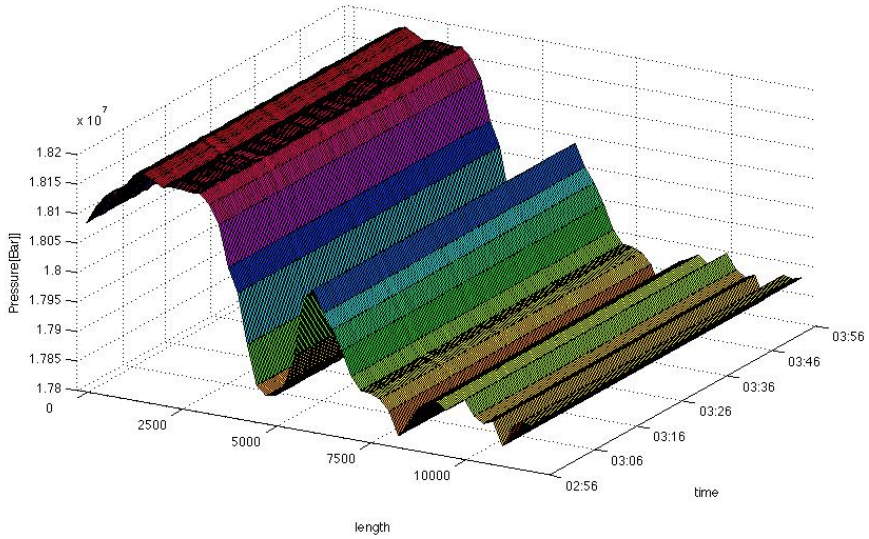
```
%   ******************************
%   * EVALUATION OF THE INERTIAL *
%   * MULTIPLIER                 *
%   ******************************

function
alfa=inertial_multiplier(fi,inletbc,fluid,pipeline,gas,massflow_corr)

if inletbc.case==2
    delta_M=massflow_corr*0.5*((mean(inletbc.massflow)-
min(inletbc.massflow))+(max(inletbc.massflow)-mean(inletbc.massflow)));
    oscill_period=input('oscillation period of massflow[min]: ');
    omega=1/(oscill_period*60)*2*pi;
    delta_q=delta_M/(massflow_corr*fluid.massflow(1));

elseif inletbc.case==3
    dM=input('Time derivative of massflow: ');
    delta_M=max(inletbc.massflow);
    omega=(pi/2)*dM/delta_M;
    delta_q=delta_M/fluid.massflow(1);
elseif inletbc.case==4
    dM=0.013;
    delta_M=(max(inletbc.massflow)-min(inletbc.massflow));
    omega=(pi/2)*dM/delta_M;
    delta_q=(pipeline.area/gas.wavespeed)*((max(fluid.pressure)-
min(fluid.pressure))/fluid.massflow(1));
else
    disp('not a valid case for the inertial multiplier')
    quit;
end
h=pipeline.dx/pipeline.length;
m=fluid.massflow(1)/(gas.rho*pipeline.area)/gas.wavespeed;
friction=frictionfactor(fluid.massflow(1),gas,pipeline);
sigma=friction*pipeline.length/(2*pipeline.diameter);
p_d=sqrt(1-2*sigma*m^2);
omega_star=omega*pipeline.length/gas.wavespeed;

alfa=sqrt(1-
(sigma*m*h)^2/(3*p_d)+fi*sqrt(p_d*(1+(sigma*m*omega_star)^2))/(m*omega_star
*delta_q));
end
```

Appendix B: 3D Plots

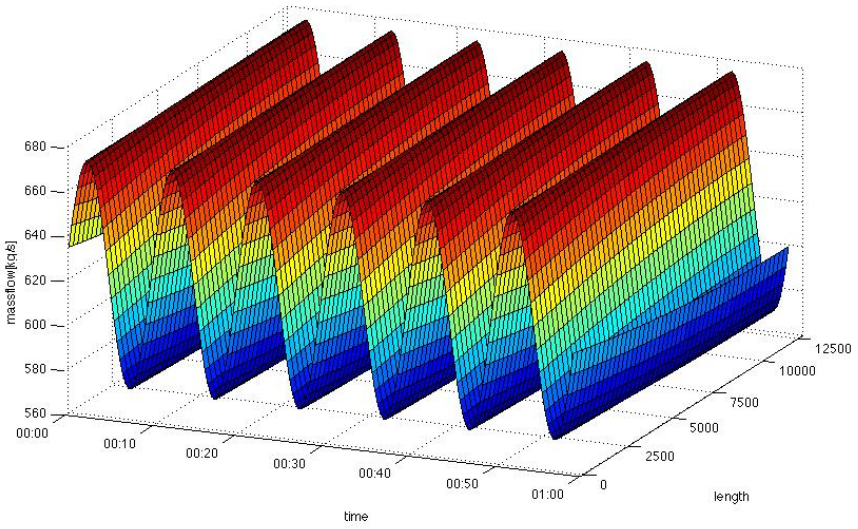
Case 2: Massflow



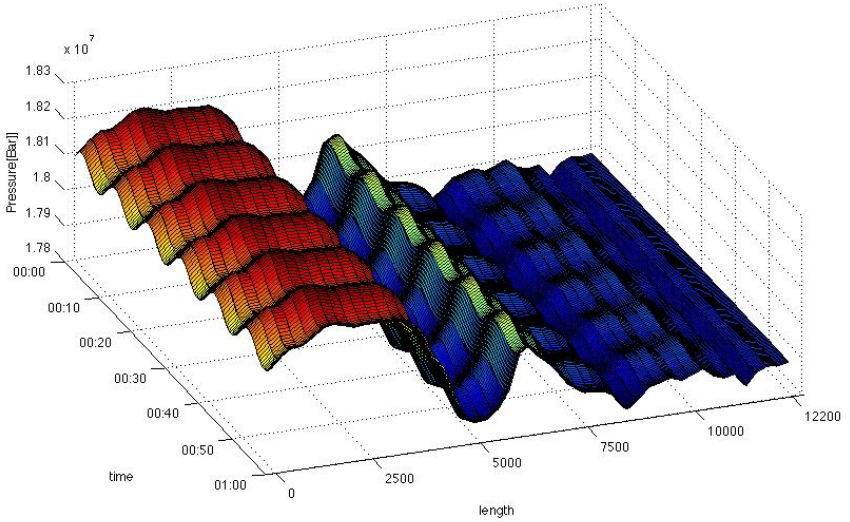
Case 2: Pressure



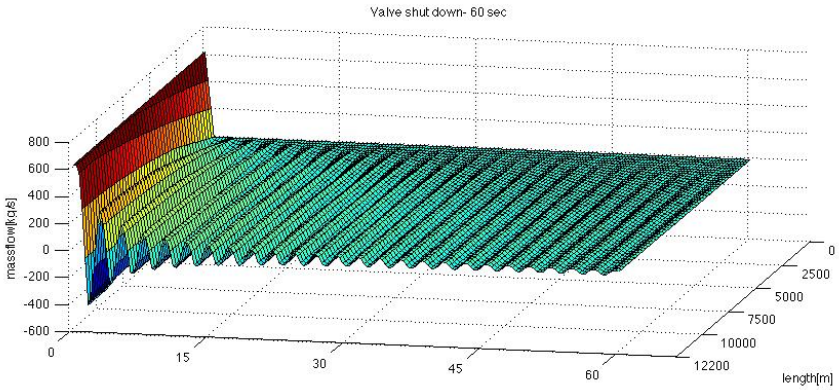
Case 4: Massflow



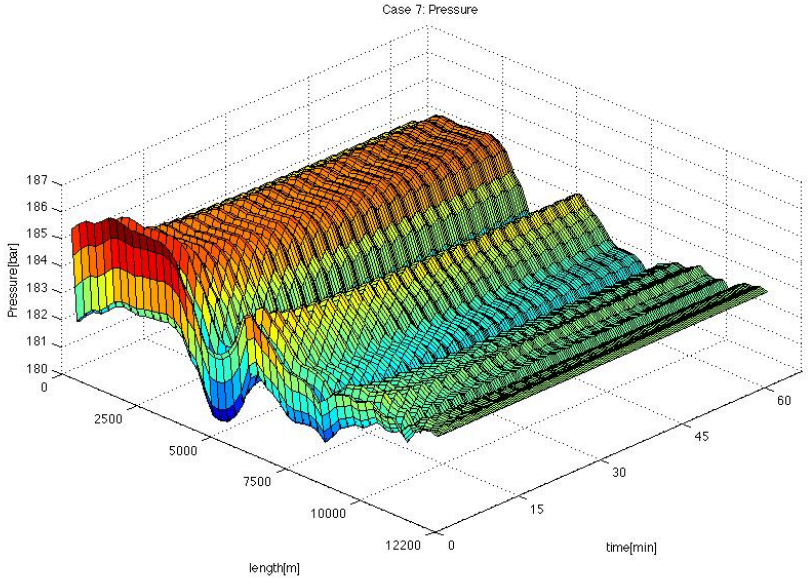
Case 4: Pressure



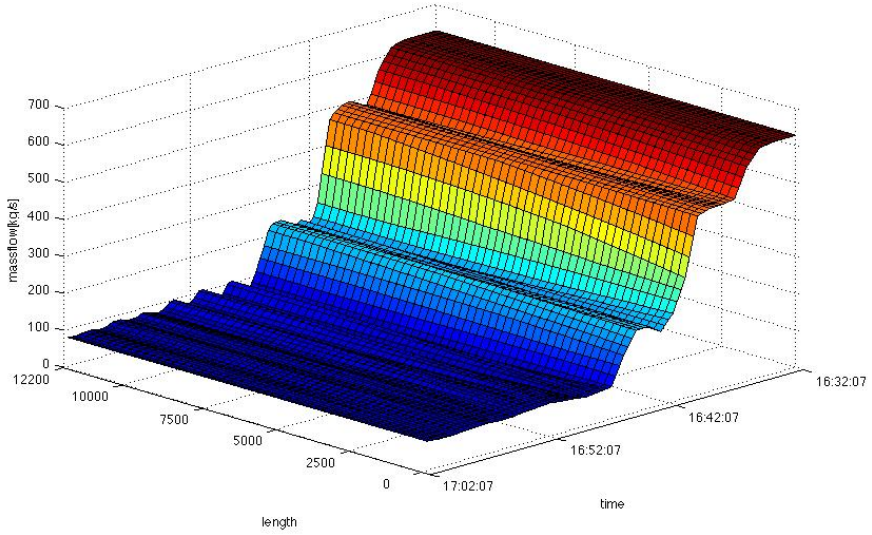
Case 7: Massflow



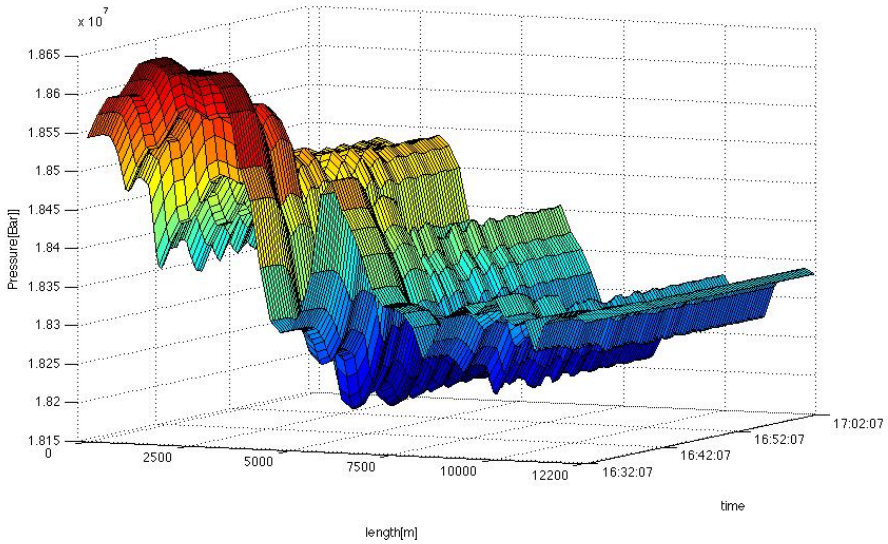
Case 7: Pressure



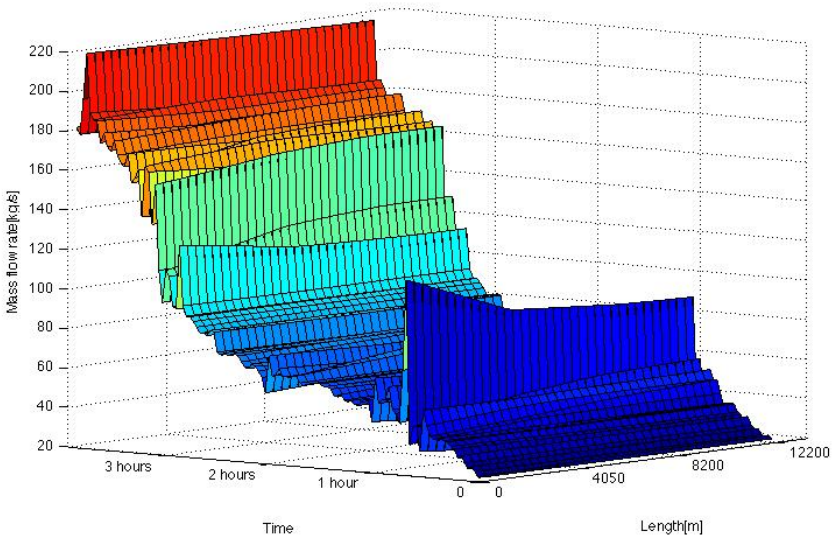
Case 9: Massflow



Case 9: Pressure



Case 11: Massflow



Case 11: Pressure