Transient Flow in Gas Transport

Joachim Dyrstad Gjerde

Master of Science in Product Design and Manufacturing
Submission date: March 2011
Supervisor: Tor Ytrehus, EPT

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

Problem Description

Large-scale gas transport I long pipelines is an important part of Norwegian and global petroleum industry. Models based on the equations of fluid mechanics are used to provide instant values of the properties in order to obtain maximum utilization of the pipelines. Much of the uncertainty of these models is related to the transients, with start-up and shutdown of flow.
The paper seeks a self-developed model for calculation of the parameters in natural gas transportation, by use of the method of characteristics. The aim of the model is to provide results as accurate as possible compared to measured results done by Gassco.
The paper is done, first having a brief introduction to computational methods used today, and developing-/expansion of existing model based on the method of characteristics. The paper is finished with a comparison of the results to measured data, and a discussion of the model.

Assignment given: 19. October 2010
Supervisor: Tor Ytrehus, EPT

Norges teknisk-
naturvitenskapelige universitet
NTNU

Institutt for energi- og prosessteknikk

EPT-M-2009-94

MASTEROPPGAVE

for

Stud.techn. Joachim Dyrstad Gjerde

Høsten 2010

Transient strømning i gasstransport

Transient Flow in Gas Transport

Bakgrunn
Stor-skala gasstransport i lange eksportledninger er en viktig del av norsk og internasjonal petroleumsindustri. Fluidmekanikk-baserte simuleringsmodeller i sann tid er viktige ingredienser både for instantan utnyttelse av rørledningenes kapasitet, og i "look ahead" planlegging av transport som skal møte krav til leveranser på mottakerstedet 500-600 km unna en dag eller to senere. Den største usikkerheten i beregningsmodellene synes å være knyttet til transiente perioder forbundet med avstengning/oppstart i deler av transportsystemet. Der er følgelig en viss fokus på det transiente strømningsregimet i håp om å forbedre modellene akkurat her. Denne masteroppgaven er et ledd i en større forskningsinnsats initiert av Gassco for å identifisere de svake punktene i dagens beregningsmodeller på området, og for å introdusere forbedringer.

Mål
Målet med oppgaven er å komme frem med en egenutviklet numerisk beregningsmodell som er i stand til å predikere transiente strømningsforløp i en eksportledning over en begrenset lengde; for eksempel Kårstø – Bokn som er en liten strekning av Europipe 2. For denne strekningen foreligger der driftsdata som kan brukes til validering av beregningene. Det er også av interesse å sammenligne resultater fra en slik modell med resultater fra de profesjonelle beregningsvertøyene som er i bruk hos Gassco, og som viser betydelige avvik fra målte verdier.

Oppgaven bearbeides ut fra følgende punkter

1. En kort gjennomgang av dagens status på området fluidmekanikk i gasstransport.
2. Utvikling av egen transient beregningsmodell, for eksempel basert på karakteristikkmetoden.
3. Om mulig bør energiligningen inkluderes i modellen for å få med et realistisk temperaturforløp langs Kårstø – Bokn ledningen.
4. Sammenligning med målte data for trykk og volumstrøm i transiente perioder.
5. Diskusjon av resultatene og oppsummering av modellens fordeler og begrensninger.

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 alle (andre) 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, 11. november 2010

Olav Bolland
Instituttleder

Tor Ytrehus
Faglig ansvarlig/veileder

Medveileder(e)

Preface

This report is the final project for receiving the Master of Science, or 'Sivilingeniør' at the Norwegian University of Science and Technology. The thesis has been done at the faculty of energy- and process engineering, fall and winter of 2010/11.

The report is in general a continuation of the work that was done in the project in spring 2010, on the topic of 'Transient gas transport'. The project left off stating that the developed model seemed provide somehow precise results at steady-state but collapsed as a consequence of wrong assumptions at the outlet for transient cases. The desire was therefore to find out how accurate the developed model actually was, and also to take it one step further.

The project has involved a lot of frustration, especially concerning the solution of the energy equation. For a long time the whole method seemed to collapse as a result of stability issues. I was unable to find any previous work that had similarities to mine, and some unintended assumptions had to be made in order to make the solution stable.

I would also like to direct a great thanks to Prof. Tor Ytrehus for his advice and knowledge over the period of this work, and for not giving up on me.

[illegible]

Joachim Dyrstad Gjerde,

Trondheim, 15th March 2011

Abstract

Transport of natural gas to continental Europe and UK is a large portion of Norwegian petroleum industry. The gas is mainly transported undersea in large-scale transport pipelines. Amount of transported gas is currently close to maximum capacity of the pipeline network, and as a consequence the gas transport must be careful planned so that the optimal capacity can be utilized.

An important tool in this planning is the use of computational method to predict the flow. Accurate computational tools is therefore of great value when predicting the pressures and flow rates in transient cases such as opening of a valve or shut down of a flow. This report is a part of a major research project initiated by Gassco, for better flow-predictions models in natural gas pipelines.

A computational model based on the method of characteristics has been developed. In this report the main focus is on the solution of the energy equation and introduction of this equation to an already existing code solving for pressure and mass flux. The method is verified using measured values of pressure at the inlet.

Since much of the uncertainty is related to the transients, this report focuses on transient cases. The old program solving the characteristic equations using an isothermal assumption actually proves surprisingly accurate, and the additional solution of temperature does not significantly improve the results. The method however does not provide satisfactory results at the larger transients.

If large temperature gradients are imposed on the solver we see instabilities in the flow and it affects the solution of the parameters. The Joule Thomson effect that we have in our solution also results in a much higher drop of temperature than what can be measured, in case of pressure drop at the inlet.

From the results we also see that the coefficient that is supposed to correct friction factor for additional drag effects, also should be a function of pressure and/or Reynold number. If such a correlation would provide more accurate results in the transient has not been debated, but more accurate correlation of friction depending on flow rate would probably give a more accurate result.

Also worth noticing is that the method does not have a clear convergence, or reduction of error as the number of calculation points increases. It gives smaller extreme values, but average error is not reduced significantly. This is probably a result of the reduced effect of missing convective-term as the grid has a finer resolution and time-step decreases and the effect of loss of velocity in the characteristic becomes small.

As a simple tool for calculation of gas transport in pipelines, the isothermal method of characteristics proves to give surprisingly accurate results. However, for more complex systems, i.e. including the temperature and variable properties such as compressibility and density, finite difference methods are more versatile. Finite difference methods can be done implicit, giving a more stable solver, and it's simpler to account for some of the effects such as temperature etc.

Samandrag

Transport av naturgass til det europeiske kontinentet og England er ein viktig del av norsk petroleumsverksemd. Gassen vert i all hovudsak transportert frå raffineri i store røyr som går under sjøen til ein mottakssentral på andre sida. Den mengda som per i dag vert transportert via slike røyr er per i dag nokså nære den maksimale kapasiteten røyrnettverket kan takle. Difor er planlegging av slik gasstransport særs viktig, slik at ein kan utnytte røyra makismalt.

Ettersom ein ikkje kan måle undervegs i røyret og berre ved målestasjonar, typisk ved start og ende av røyret, må ein nytte berekningsmodellar for å fastslå straumen. Nøyaktige berekningsverkty er difor særs viktige når ein slik transport skal planleggast i transiente tilfeller. Slike tilfeller er til dømes opning eller stenging av ein ventil. Denne rapporten er ein del av eit større forskingsprosjekt sett i gang av Gassco, som styrer den norske gasstransporten til utlandet, for utarbeiding av betre modellar for å kunne fastslå straumen i gassrøyr.

Eit berekningsverkty basert på karakteristikkmetoden har vore utvikla. Verktyet baserar seg på ein eksisterande kode, og hovudfokuset har her vore løysinga av energilikninga, å få denne tilpassa ein allereie fungerande kode. Verktyet vert verifisert ved å samanlikne trykk ved innløp med målte resultat.

Den største usikkerheita er knytt til transiente hendingar, og denne rapporten fokuserer på transiente tilfeller. Metoden syner særs god nøyaktigheit ved relativt små endringar i straumen, men i sjølve transienten vert ikkje resultata like gode. Den gamle koden tok utgangspunkt i konstant temperature, og tilhøyrande eigenskapar. Og med rette grensetilhøve syner metoden seg å gi betre resultat enn venta, og den nye koden med løysing av temperaturlikninga gjev ikkje merkbart betre resultat. Store temperaturegradientar inni sjølve straumen gjev derimot ustabilitetar og utslag på resultata. Joule Thomson effekten medfører også eit for høgt temperaturfall ved trykkfall på innløpet.

Ein kan og sjå at ein koeffisient som er satt for å korrigere for drag-effekter i straumen i høve til friksjon burde vore ein funksjon av trykk og/eller massestrøm. Dersom denne hadde vore korrigert i høve til dei nemnte parameterane ville ein kunne oppnå endå betre resultat. Om ei slik korrigering ville gitt betre resultat i sjølve transienten er ikkje vurdert.

Verdt å merke seg er at ein ikkje ser ein klar samanheng mellom auka antal nodar og feil. Ved å nytte ei betre oppløysing og fleire nodar, oppnår ein mindre ekstremverdiar, men snittfeil er omtrent uforandra. Dette er truleg eit resultat av at tapet av den konvektive termen i løysaren for trykk og massestrøm, ikkje vert like signifikant ved høg oppløysing ettersom tidssteget vert mindre og tapet av hastigheita i karakteristikken ikkje vert like stor.

Som eit enkelt verkty er den forenkla forma av karakteristikk metoden eit enkelt og godt verkty, men for meir kompliserte modellar, der ein og tek temperaturen med i utrekninga gjev ein differanse metode ein meir hendig metode, der ein kan gjere metoden implisitt.

Contents

List of Figures

List of Tables

Nomenclature

$Nu = \frac{h \cdot d}{k}$ Nusselt number

$Pr = \frac{c_p \mu}{k}$ Prandtl number

$Re_D = \frac{\rho U D}{\mu}$ Reynolds number

α	pipeline angle
$\bar{p}$	mechanical pressure
δ_{ij}	Diracs delta function
ϵ	surface roughness
γ	speed of sound [m/s](Chapter 3)
κ	bulk viscosity
κ_g	Iisothermal coefficient of compressibility
κ_r	presudo reduced compressibility
λ	viscosity multiplicator coefficient
μ	dynamic viscosity
$\mu\Phi$	dissipation function
μ_{JT}	Joule Thomson coefficient
ν	gas velocity [m/s] (Chapter 3)
ρ	density
ρ_r	pseudo-reduced density
σ_{ij}	stress tensor
τ_{ij}	shear stress tensor
C^+	positive characteristic

C^-	negative characteristic
c_p	specific heat at constant pressure
c_v	specific heat at constant volume
d_{outer}	outer diamter pipeline
h_o	outer heat transfer coefficient
k_{gas}	thermal conductivity gas
k_{souil}	thermal conductivity soil
p_c	critial pressure
p_r	reduced pressure
r_{inner}	inner radius pipeline
r_i	inner radius i'th layer
r_{outer}	outer diameter pipeline
S_p	stiffness matrix
T_c	critical temperature
T_{env}	ambient temperature
T_{gas}	temperature gas
T_{pr}	pseudo-reduced temperature
T_r	reduced temperature
$U_{W,tot}$	overall heat transfer coefficient
v	volume (Chapter 4)
f_i	force vector
A	pipeline area
a	speed of sound (Chapter 3)
E	total energy
e	internal energy
EFF	friction factor coefficient
g	acceleration of gravity

h	enthalpy
k	thermal conductivity [W/mK]
M	mass flow rate
M	molecular weight (Chapter 4)
MSm3	million standard cubic metres
n	molar density
n	normal vector
p	pressure
Q	heat
q	heat transfer rate
R	specific gas contant
s	entropy
T	temperature
U	1-D velocity
V	specific volume
W	work
Z	compressibility factor

Chapter 1

Introduction

Natural gas is becoming a larger portion of the petroleum sector, and is considered to play a more important role in the future of environmental friendly energy supply. The total energy delivered by Norway in form of gas is equal to eight parts of the total production of electricity, and covers approximately 15% of the total European gas consumption [23]. In the last year however the estimates of existing reservoirs and resources have been downscaled, making the future deliverance uncertain of Norwegian gas to the European mainland. One could however assume that new search areas will be opened for test drilling and that there still exists large undiscovered resources on Norwegian territory, making transportation of gas to Europe an important business for Norway for decades to come.

Gas is processed at refineries on shore and separated into natural gas and LNG (liquefied natural gas). LNG is mainly produced at the Snøhvit facility, and shipped to North America or South in Europe. A small portion of natural gas is sold and used in Norway, but the main part is transported in large pipelines to the European continent and the UK.

Norwegian gas production 2009		
Export through pipelines	96,564 MSm3	93.6%
Sale in Norway	1,388 MSm3	1.3%
Reinjection	1,840 MSm3	1.8%
LNG	3,358 MSm3	3.3%
Total	103,160 MSm3	100%

* MSm3 stands for million standard cubic metres

Table 1.1: Norwegian gas production[23]

As seen from the table above produced gas at Norwegian continental shelf already have passed 100 Billion Standard Cubic metres pr year and is expected to grow even further the next decade. Estimates vary from 105 to 130 billion Sm3. The current transport capacity in the existing pipeline system is approximately

120 Billion Sm3, which means that the current use is close to the maximum capacity and could pass the existing capacity in the years to come. This increases the importance of optimal utilization of the existing pipeline system, since investments in transport pipelines are capital intensive.

Parameters of gas transport, such as temperature, pressure and mass flux, can be measured at metering stations. These metering stations are typically placed at the inlet and the outlet of the pipeline, and the state of the gas in between must rely on computational models. With almost maximum capacity of the existing pipeline already used, the importance of planning of natural gas transport becomes more important. The planning is done using computational models based on basic equations of fluid mechanics. Accurate model that can reproduce the flow is therefore of great value in the planning of gas transport.

This report is a part of a larger research project initiated by Gassco, where the goal is to determine the weak sections of today's computational models for gas transport. If one better can determine the weak sections of the computational models, better simulators can be made, and better foresight and planning of the gas transport can be done.

This report uses an already existing code written in the project assignment 'Transient gas transport [13], based on the method of characteristics. The original code will be expanded and also include the temperature. This involves solution of the energy equation, and an extensively explanation of the energy equation in gas transport will be given, and a final solution of the equation that can be solved by the method of characteristics and included in the existing code. The previous code was based on the derivation done by Streeter and Wylie [4] and therefore made a few simplifying assumptions. The main difference from the old program is the assumption of isothermal flow. As a result of this assumption, wave speed, compressibility factor and density were previously kept constant through the entire calculation, whereas in the new program these properties become functions of both space and time. The simulations are done on real cases, using the test section from Kårstø to Vestre-Bokn, which is a 12,2 km long section with metering stations at upstream and downstream ends. Results from these metering stations will therefore be used in the validation of the new program.

Chapter 2

Governing equations

The governing equations of the flow are based on the basic conservation laws of fluid mechanics, and pipeline diameter is considered to be constant. The equations for momentum and continuity has been derived in detail in [13], and will not be given in detail. In general we have four equations governing the flow in a pipeline

- Equation of state
- Continuity
- Equation of motion
- Energy equation

2.1 Equation of state

In addition to the conservation equations, an equation of state is also introduced. Ideal gas law states that

$$pv = nRT \tag{2.1}$$

Where n is can be related to number of molecules and v represents the volume. By specific molar weight of the gas the term can be represented by its density, ρ. This equation describes the behavior of gas under ideal conditions. In case of extreme external factors, such as extreme temperatures and/or pressure, the gas no longer behaves ideal and we introduce the real gas law

$$p = \rho ZRT \tag{2.2}$$

where Z is a factor accounting for the change of the behavior of the gas under non-ideal conditions. This factor is called compressibilityfactor, is dimensionless and a function of pressure and temperature. By definition, the compressibility factor can be defined as the actual occupied volume by a gas compared to volume occupied under ideal conditions.

2.2 Continuity equation

The continuity or the preservation of mass states that as we follow a specific mass it may change shape and volume, but the mass will remain unchanged. Thus we write

$$\frac{D}{Dt}\int \rho dV = 0$$

We convert this into a volume integral by the Reynolds's transport theorem. Since the total mass cannot change the integrand must be equal to zero, and the equation can be written as

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial u_k}\left(\rho u_k\right) = 0 \tag{2.3}$$

Which again can be expanded to

$$\frac{\partial \rho}{\partial t} + u_k \frac{\partial}{\partial u_k}\left(\rho\right) + \rho \frac{\partial}{\partial u_k}\left(u_k\right) = \frac{D\rho}{Dt} + \rho \frac{\partial u_k}{\partial u_k} = 0 \tag{2.4}$$

2.3 Conservation of momentum

General momentum equation is for a flow in three dimensions is given as

$$\rho \frac{\partial u_i}{\partial t} + \rho u_j \frac{\partial u_i}{\partial u_j} = \frac{\partial \sigma_{ij}}{\partial x_i} + \rho f_i \tag{2.5}$$

The resulting momentum equation for flow in a one-dimensional pipeline can simplified to

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} = -\frac{1}{\rho}\frac{\partial p}{\partial x} - f\frac{U\left|U\right|}{2D} - g \cdot sin\left(\alpha\right) \tag{2.6}$$

where the term f, is a result of Darcy Weisbach's [27] friction formula.

2.4 Conservation of energy

During transportation in a pipeline the energy and temperature of the gas changes. This is due to several effects that will be discussed later in this paper, and a final equation will be presented and solved. The general expression for the internal energy of a fluid can be derived from the first law of thermodynamics, and the result is

$$\rho \frac{De}{Dt} + p\frac{\partial u_j}{\partial x_j} = \mu \Phi + \frac{\partial}{\partial x_j}\left(k\frac{\partial T}{\partial x_j}\right) \tag{2.7}$$

The terms of the energy equation is a result of

- Total change of internal energy
- Change of energy due to pressure effects

- Dissipation of momentum energy
- Heat convection

The energy equation will be extensively treated in chapter 4.

Chapter 3

Numerical methods for solving gas transport problems

The problem of gas transport in large pipeline cannot be resolved exactly. Flow is three-dimensional, solution is depending on the Navier-Stokes equations and the conservation of energy. The problem must be simplified and a numerical procedure to solve the problem must be introduced. The equations are only solved in one dimension, and the terms must be rewritten in order to fit our one-dimensional approach. A number of methods to solve the problem of gas transport can be utilized, such as finite element method, finite volume method and finite difference methods. In addition we will also introduce the method of characteristics, which will be the selected solver in this paper.

3.1 Finite difference methods

The most frequently used method for larger commercial simulators is the method of finite differences. Langelandsvik's thesis [17], describes such a finite difference approach by the software called TGNet. A brief introduction of the simulator will be given to highlight the finite difference approach.

To solve the conservation of mass and momentum, the equations can be written as

$$\frac{\partial u}{\partial t} + \mathbf{A}\frac{\partial u}{\partial x} = \mathbf{F} \tag{3.1}$$

where

$$\mathbf{A} = \frac{1}{L}\begin{bmatrix} 0 & 1 \\ \gamma^2 - \nu^2 & 2\nu \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 0 \\ \frac{-f \cdot m|m|}{2d\rho} - \frac{1}{L}\rho g sin\alpha \end{bmatrix}$$

and u is given as $\begin{bmatrix} \rho \\ m \end{bmatrix}$. γ and ν refers to speed of sound and gas velocity respectively. In addition TGNet uses a linearization process for the previous time step in order to obtain the values at the new. This is done so that the solver avoids solving non-linear equations. The procedure will not be given in detail her, but has been more extensively described by Langelandsvik. The simulator then evaluates the variables at the new and previous time step, at the midpoint, $\left(x_{i+1/2}, t_{j+1/2}\right)$.

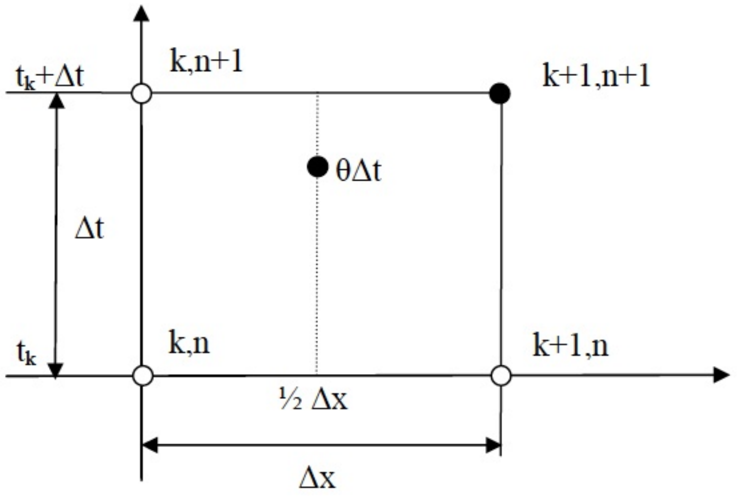


Figure 3.1: Numerical box scheme, evaluation at midpoints [17]

Using the following relations

$$U\left(x_{i+1/2}, t_j\right) = \frac{1}{2}\left(U\left(x_{i+1}, t_j\right) + U\left(x_i, t_{,j}\right)\right)$$

$$U\left(x_i, t_{j+1/2}\right) = \frac{1}{2}\left(U\left(x_i, t_j\right) + U\left(x_i, t_{,j+1}\right)\right)$$

$$\frac{\partial}{\partial t}U\left(x_{i+1/2}, t_{j+1/2}\right) = \frac{1}{\triangle t}\left(U\left(x_{i+1/2}, t_{j+1}\right) + U\left(x_{i+1/2}, t_{,j}\right)\right)$$

$$\frac{\partial}{\partial x}U\left(x_{i+1/2}, t_{j+1/2}\right) = \frac{1}{\triangle x}\left(U\left(x_{i+1}, t_{j+1/2}\right) + U\left(x_i, t_{,j+1/2}\right)\right)$$

Together this forms an implicit method where each of the values at the next timestep depends on the solution of the previous value of the next time step. This means that the new values must be calculated at the exact same time by means of some numerical method, i.e. LU-decomposition and the TDMA algorithm. More detailed information on the mathematical procedures can be found in Luskin[19].

3.2 Method of characteristics

A much used method to solve hyperbolic partial differential equations(PDE's), such as the Navier-Stokes equations, is the method of characteristics. The method consists of reducing a partial differential equation to an ordinary differential equation. This method has been quite popular for simple calculations, both for gas flow problems and in the hydropower industry.

3.2.1 Explicit Method of Characterisitcs(MOC)

The method consists by combining the conservation equation and the momentum equation. As a result, the partial differential equations (PDE's) 'collapse' and form ordinary differential equations (ODE's) along its characteristics. The one-dimensional Navier-Stokes equations yield

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho U)}{\partial x} = 0 \tag{3.2}$$

$$\frac{\partial(\rho U)}{\partial t} + u\frac{\partial\{\rho U)}{\partial x} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left[\mu\frac{\partial U}{\partial y}\right] - \rho g \sin\alpha \tag{3.3}$$

Using a Darcy-Weissbach friction formula [4], for the loss due to friction inside the pipe

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} = -\frac{1}{\rho}\frac{\partial p}{\partial x} - f\frac{U\left|U\right|}{2D} - g\sin\alpha \tag{3.4}$$

Moving all terms over to the left hand side to get an equation in the form of $f(x) = 0$

$$\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + f\frac{U\left|U\right|}{2D} + g\sin\alpha = 0 \tag{3.5}$$

The equation of conservation can be rewritten using the following relation between density and pressure

$$p = \rho ZRT,\ \rho = \frac{p}{ZRT} \tag{3.6}$$

And introduction of the speed of sound

$$B^2 = \left(\frac{\partial p}{\partial \rho}\right)_T \tag{3.7}$$

As a result we can write the continuity equation in terms of dependent variables p and U

$$\frac{\partial p}{\partial t} + U\frac{\partial p}{\partial x} + \rho B^2\frac{\partial U}{\partial x} = 0 \tag{3.8}$$

The characteristics equation can be found by combining conservation equation and momentum equation, using a multiplication factor λ.

$$\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + f\frac{U\left|U\right|}{2D} + g\sin\alpha + \lambda\left[\frac{\partial p}{\partial t} + U\frac{\partial p}{\partial x} + \rho B^2\frac{\partial U}{\partial x}\right] = 0 \quad (3.9)$$

Rearrangement or the terms in equation 3.9, shows that the same equation can be written as

$$\lambda\left[\frac{\partial p}{\partial t} + \left(U + \frac{1}{\rho\lambda}\right)\frac{\partial p}{\partial x}\right] + \left[\frac{\partial U}{\partial t} + \left(U + \rho\lambda B^2\right)\frac{\partial U}{\partial x}\right] + f\frac{U\left|U\right|}{2D} + g\sin\alpha = 0 \quad (3.10)$$

A closer look at the equation above reveals its similarity to the substantial derivative, $\frac{DA}{Dt} = \frac{\partial A}{\partial t} + \frac{dx}{dt}\frac{\partial A}{\partial x}$, where

$$\frac{Dx}{Dt} = U + \frac{1}{\rho\lambda} = U + \rho\lambda B^2 \quad (3.11)$$

Solving this with respect to λ, gives

$$\lambda = \pm\frac{1}{\rho B}. \quad (3.12)$$

Inserting this back into equation 3.10gives

$$\frac{DU}{Dt} \pm \frac{1}{\rho B}\frac{Dp}{Dt} + f\frac{U\left|U\right|}{2D} + g\sin\alpha = 0 \quad (3.13)$$

Where

$$\frac{dx}{dt} = U \pm B \quad (3.14)$$

Using a positive sign refers to the positive characteristic called C^+, and the negative characteristic, C^-. These will be represented as curved lines in the xt-plane.

A major restriction on the explicit method of characteristics, is the strict conditions on the time step. The MOC suffers under the Courant-Friedrich-Levy, or CFL condition which states that

$$u + |B| \leq \frac{\triangle x}{\triangle t} \quad (3.15)$$

Meaning that the largest time step available is restricted by ${}^{\triangle x}/_{a+C}$, which in some cases means that the time step becomes very small. This can be shown graphically in figure 3.2

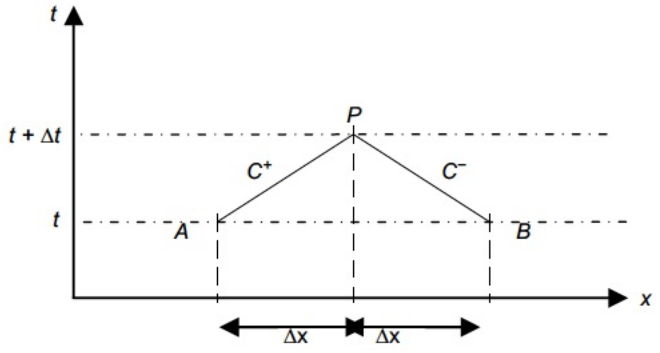


Figure 3.2: Characteristic curves in xt-plane

3.2.2 Implicit Method of Characteristics(IMOC)

As a remedy for the shortcomings concerning time steps for the explicit MOC, implicit methods have been introduced. These methods are used in particular to simulations and analysis of water hammers in hydropower and hydraulics. As i result of an implicit approach, the CFL-condition no longer limits the time step, and longer time steps can be made.

Using central differences

Afshar and Rohani[3] presented a solution of the implicit method of characteristics for solution of waterhammers in hydropower systems, using central differences in a "*box*" scheme. This method was used for the simplified equations, not containing the convective term, where $\frac{dt}{dx} = \pm a$ becomes the characteristics (a represents the speed of sound). The box scheme used is shown in figure 3.3.

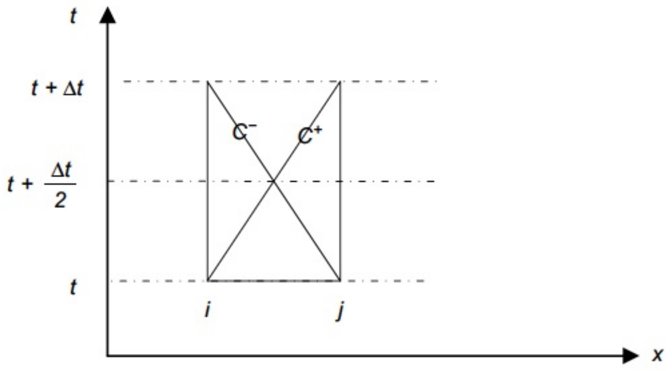


Figure 3.3: IMOC using central differences[3]

Using that n represents a discrete time step we evaluate the characteristic equations at $n + 1/2$. For the C^+ characteristic the equation becomes

$$\frac{dU}{dt} + \frac{1}{\rho a}\frac{dp}{dt} + \frac{f}{2D}U\,|U| + g\sin\alpha = 0\ |^{n+1/2}$$

Using central differences

$$\frac{dU}{dt} = \frac{U_{i+1}^{n+1} - U_i{}^n}{\triangle t}$$

$$\frac{dp}{dt} = \frac{p_{i+1}^{n+1} - p_i^n}{\triangle t}$$

Weighting of the term representing friction loss

$$\frac{f}{2D}U\,|U| = \frac{f}{2D}\left[\frac{1}{4}(U_{i+1}^{n+1})^2 sign(U_{i+1}^{n+1} + U_i^n) + \frac{1}{2}U_{i+1}^{n+1}U_i^n sign(U_{i+1}^{n+1} + U_i^n) + \frac{1}{4}(U_i^n)^2 sign(U_{i+1}^{n+1} + U_i^n)\right]$$

Results in the equation

$$\frac{U_{i+1}^{n+1} - U_i{}^n}{\triangle t} + \frac{1}{\rho a}\frac{p_{i+1}^{n+1} - p_i^n}{\triangle t} + \frac{f}{2D}[\frac{1}{4}(U_{i+1}^{n+1})^2 sign(U_{i+1}^{n+1} + U_i^n)$$

$$+\frac{1}{2}U_{i+1}^{n+1}U_i^n sign(U_{i+1}^{n+1} + U_i^n) + \frac{1}{4}(U_i^n)^2 sign(U_{i+1}^{n+1} + U_i^n)] + g\sin\alpha = 0 \quad (3.16)$$

And similar for the C^- characteristic. The sign function returns a positive sign if the result is positive and a negative if negative and zero if result is zero

$$sign(A) = \begin{cases} 1 & A > 0 \\ 0 & A = 0 \\ -1 & A < 0 \end{cases}$$

By moving all the terms from the previous timestep to the right hand side of the equation we get

$$\left[1 + \frac{1}{4}\frac{f}{2D}U_{i+1}^{n+1} sign(U_{i+1}^{n+1} + U_i^n) + \frac{1}{2}\frac{f}{2D}U_i^n sign(U_{i+1}^{n+1} + U_i^n)\right]U_{i+1}^{n+1}$$

$$+\frac{1}{\rho a}p_{i+1}^{n+1} = U_i^n + \frac{1}{\rho a}p_i^n - \frac{1}{4}\frac{f}{2D}(U_i^n)^2 sign(U_{i+1}^{n+1} + U_i^n) + g\sin\alpha \qquad (3.17)$$

Which can be written in a matrix form as

$$\mathbf{S_p x_p} = \mathbf{b_p} \qquad (3.18)$$

Where $\mathbf{x_p} = [\mathbf{U_i}, \mathbf{U_{i+1}}, \mathbf{p_i}, \mathbf{p_{i+1}}]^{n+1}$ and $\mathbf{S_p}$ is the matrix of unknowns, also referred to as stiffness matrix, and $\mathbf{b_p}$ is a vector of known values. $\mathbf{S_p}$ is then a

$2x4$ matrix and $\mathbf{b_p}$ is a $2x1$ vector. Since the equations are non linear, due to the term $U_{i+1}^{n+1} \cdot U_{i+1}^{n+1}$ a linearization must be done to obtain a solution to the system of equations. Using a Newton-Rhapson approach will result in the same set of equations, replacing the $\mathbf{x}$ vector by $\triangle\mathbf{x}$, where $\triangle\mathbf{x} = [\triangle U_i, \triangle U_{i+1}, \triangle p_i, \triangle p_{i+1}]$ for the next timestep. The $\triangle$ meaning the difference between old and new iteration in our Newton-Rhapson scheme.

Even though this method was derived for waterhammer analysis, one could imagine a somewhat similar approach to the problem of gas transport, introducing a central difference scheme, and obtaining a similar stiffness matrix, but different coefficients.

Solving the set of matrices

This method lead to either a set of $2x4$ matrices that needs to be solved, since each of the characteristics requires one boundary condition. The matrices can be written as

$$\mathbf{S_p} = \begin{bmatrix} 0 & C_1 & 0 & D \\ C_2 & 0 & -D & 0 \end{bmatrix}$$

Or we get a space matrix following the same form as the equation presented above. The term D refers to $1/\rho a$ and the C_1 and C_2 refers to

$$C_1 = [1 + \frac{1}{2}\frac{f\triangle t}{2D}U_{i+1}^{m} sign(U_{i+1}^{m} + U_{i}^{n}) + \frac{1}{2}\frac{f\triangle t}{2D}Ui^{n} sign(U_{i+1}^{m} + U_{i}^{n})]$$

$$C_2 = [1 + \frac{1}{2}\frac{f\triangle t}{2D}U_{i}^{m} sign(U_{i+1}^{m} + U_{i}^{n}) + \frac{1}{2}\frac{f\triangle t}{2D}U_{i+1}^{n} sign(U_{i+1}^{m} + U_{i}^{n})]$$

This can be solved by putting the $2x4$ matrix into a system where the matrices must be solved using the result of the previous matrix as the new boundary condition. This requires a numerical procedure in order to solve. The other solution creates a $Nx2N$ and solve the single matrix by a numerical procedure. A solution for this was presented Edenhofer and Schmitz [9].

Chapter 4

The Energy Equation

In the old program the energy equation was not solved, and we considered the flow to be constant This will off course not be the case in real life since the temperature and pressure changes with transients and as a other effects such as hydrostatic pressure.

4.1 Derivation of the energy equation

The energy equation can be derived from the 1st law of thermodynamics. The change of energy is equal to the sum of work done on the system and the heat added.

$$dE = dW + dQ \tag{4.1}$$

The term on the left hand side refers to the total change of energy of the system, which consists of two parts; change of internal energy and change of kinetic energy. In some cases potential energy is also considered, but in this section the potential energy part is left out of the evaluation. Hence the energy can be written as

$$E = e + \frac{1}{2}u^2 \tag{4.2}$$

Where e refers to the internal- or intrinsic energy, and u is denoting the velocity. That means that we in a control volume V, have a total amount of energy equal to

$$\int_V \rho(e + \frac{1}{2}u^2)dV \tag{4.3}$$

Then consider the work that the fluid does during an defined event, dW. Such an event can be forces acting on the surface of an element. Forces can be pressure and viscous forces. These forces are denoted as σ_{ij}, and for the entire control volume under consideration we write

$$\int_S u_j \sigma_{ij} n_i dS \tag{4.4}$$

By use of the Gauss theorem [6], this can be written in terms of volume integral

$$\int_V \frac{\partial}{\partial x_i}(u_j \sigma_{ij}) dV \tag{4.5}$$

We then consider the latter term on the right hand side, dQ, that refers to the quantity of heat leaving the fluid. The heat leaves through the boundaries and can be written as $\int_S \mathbf{q} \cdot \mathbf{n} dS$. Further investigation of this term and using the Gauss law as we did prior, the added heat term becomes

$$\int_S \mathbf{q} \cdot \mathbf{n} dS = \int_V \frac{\partial \mathbf{q_j}}{\partial \mathbf{x}_j} dV \tag{4.6}$$

In addition we also have a term representing body forces, such as gravity and/or magnetic force $\int_V \mathbf{u} \cdot \rho \mathbf{f} dV$. The entire energy equation in terms of volume integrals can be written as

$$\frac{D}{Dt}\int_V (\rho e + \frac{1}{2}\rho \mathbf{u} \cdot \mathbf{u}) dV = \int_V \frac{\partial}{\partial x_i}(\mathbf{u_j}\sigma_{ij}) dV + \int_V \mathbf{u_j} \cdot \rho \mathbf{f} dV - \int_V \frac{\partial q_j}{\partial x_j} dV \tag{4.7}$$

We can also assume that the volume integral will be equal to zero, hence all the terms inside the integral can also be assumed to be equal to zero.

4.1.1 Convective transport of energy

The total change of energy on our left hand side is

$$\frac{D}{Dt}\left[\rho e + \frac{1}{2}\rho u_j u_j\right] = \frac{\partial}{\partial t}(\rho e + \frac{1}{2}\rho u_j u_j) + \frac{\partial}{\partial x_j}\left[(\rho e + \frac{1}{2}\rho u_j u_j) u_k\right] \tag{4.8}$$

Further evaluation of the time derivative gives

$$\frac{\partial}{\partial t}(\rho e + \frac{1}{2}\rho u_j u_j) = \rho\frac{\partial e}{\partial t} + e\frac{\partial \rho}{\partial t} + \rho\frac{\partial}{\partial t}(\frac{1}{2}u_j u_j) + (\frac{1}{2}u_j u_j)\frac{\partial \rho}{\partial t} \tag{4.9}$$

Similarly for the latter term of equation 4.8

$$\frac{\partial}{\partial x_k}\left[(\rho e + \frac{1}{2}\rho u_j u_j) u_k\right] = \rho u_k \frac{\partial e}{\partial x_k} + e\frac{\partial}{\partial x_k}(\rho u_k) + (\frac{1}{2}u_j u_j)\frac{\partial}{\partial x_k}(u_k \rho) + \rho u_k \frac{\partial}{\partial x_k}(\frac{1}{2}u_j u_j) \tag{4.10}$$

For the term $\partial/\partial x_k(\rho u_k)$ this can be replaced by $-\partial\rho/\partial t$ in terms of continuity giving

$$\frac{\partial}{\partial x_k}\left[(\rho e + \frac{1}{2}\rho u_j u_j) u_k\right] = -e\frac{\partial \rho}{\partial t} - (\frac{1}{2}u_j u_j)\frac{\partial \rho}{\partial t} + \rho u_k \frac{\partial e}{\partial x_k} + \rho u_k \frac{\partial}{\partial x_k}(\frac{1}{2}u_j u_j) \tag{4.11}$$

Summation of the results in equation 4.11 and 4.9 yields

$$\frac{D}{Dt}\left[\rho e + \frac{1}{2}\rho u_j u_j\right] = \rho\frac{\partial e}{\partial t} + \rho u_k \frac{\partial e}{\partial t} + \rho u_j \frac{\partial u_j}{\partial x_j} + \rho u_j u_k \frac{\partial u_j}{\partial x_k} \tag{4.12}$$

And the total change of energy then becomes

$$\frac{DE}{Dt} = \rho\frac{\partial e}{\partial t} + \rho u_k\frac{\partial e}{\partial t} + \rho u_j\frac{\partial u_j}{\partial x_j} + \rho u_j u_k\frac{\partial u_j}{\partial x_k} = \frac{\partial}{\partial x_i}(\mathbf{u_j}\sigma_{ij}) + \mathbf{u_j}\cdot\rho\mathbf{f_i} - \frac{\partial q_j}{\partial x_j} \quad (4.13)$$

4.1.2 Viscous term

For the viscous term, represented by forces acting on the surface, $\frac{\partial}{\partial x_i}(\mathbf{u_j}\sigma_{ij})$, it can be noted that

$$\frac{\partial}{\partial x_i}(\mathbf{u_j}\sigma_{ij}) = \mathbf{u_j}\frac{\partial\sigma_{\mathbf{ij}}}{\partial x_i} + \sigma_{ij}\frac{\partial\mathbf{u_j}}{\partial x_i} \quad (4.14)$$

If we combine the first term in the viscous representation, and the body force term we get

$$\mathbf{u_j}\frac{\partial\sigma_{\mathbf{ij}}}{\partial\mathbf{x_i}} + \mathbf{u_j}\cdot\rho\mathbf{f_i} \quad (4.15)$$

A closer look at the equation above reveals that it is equal to the right hand side in the momentum equation multiplied by $\mathbf{u_j}$.

$$\rho\frac{\partial\mathbf{u_j}}{\partial t} + \rho\mathbf{u_k}\frac{\partial\mathbf{u_j}}{\partial x_k} = \frac{\partial\sigma_{ij}}{\partial x_i} + \rho\mathbf{f_i} \quad (4.16)$$

The terms therefore cancel out, and we are left with

$$\rho\frac{\partial e}{\partial t} + \rho u_k\frac{\partial e}{\partial t} = \sigma_{ij}\frac{\partial u_j}{\partial x_i} - \frac{\partial q_j}{\partial x_j} \quad (4.17)$$

The term σ_{ij} which is the stress tensor, that we used in the derivation of the general energy equation represents

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij} \quad (4.18)$$

Where p is the pressure, τ_{ij} is the shear stress tensor and δ_{ij} represents the Kronecker delta. The relation for stress in a Newtonian fluid can be shown to be equal to

$$\sigma_{ij} = -p\delta_{ij} + \lambda\delta_{ij}\frac{\partial u_k}{\partial x_k} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \quad (4.19)$$

Making the first term on the right hand side of the energy equation equal to

$$\sigma_{ij}\frac{\partial u_k}{\partial x_k} = -p\frac{\partial u_k}{\partial x_k} + \lambda\left(\frac{\partial u_k}{\partial x_k}\right)^2 + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_k}{\partial x_k} \quad (4.20)$$

In terms of energy, the first term is the reversible work that the pressure gradient does to the fluid, or in other words the energy due to compression. The two latter terms together form the dissipation function, $\mu\Phi$. The dissipation function is a term representing the rate at which forces act on the fluid transforming mechanical or kinetic energy into thermal energy. Hence we write

$$\sigma_{ij}\frac{\partial u_k}{\partial x_k} = -p\frac{\partial u_k}{\partial x_k} + \mu\Phi \tag{4.21}$$

Where

$$\Phi = \frac{\lambda}{\mu}\left(\frac{\partial u_k}{\partial x_k}\right)^2 + \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_k}{\partial x_k}$$

Which gives us the energy equation with the heat flux term represented by Fourier's law, $-q_j = k\frac{\partial T}{\partial x_j}$

$$\rho\frac{De}{Dt} + p\frac{\partial u_j}{\partial x_j} = \mu\Phi - \frac{\partial q_j}{\partial x_j} \tag{4.22}$$

4.2 The complete equation

Equation 4.22 refers to the general equation of energy for a fluid. This equation must be modified to fit our requirements and simplifications. The equation that we are interested in is an equation for a one-dimensional compressible flow in a pipeline.

4.2.1 Classic form

Working with a term such as internal energy e, is rather confusing. Using a more tangible parameter such as temperature can sometimes be more desirable. Evaluation of the internal energy, since $e = e(T, p)$ we expand the expression

$$\frac{De}{Dt} = \left(\frac{\partial e}{\partial T}\right)_\rho \frac{DT}{Dt} + \left(\frac{\partial e}{\partial \rho}\right)_T \frac{D\rho}{Dt}$$

Using the definition of specific heat capacity at constant volume $c_v = \left({}^{\partial e}/_{\partial T}\right)_\rho$, we get

$$\frac{De}{Dt} = c_v\frac{DT}{Dt} + \left(\frac{\partial e}{\partial \rho}\right)_T \frac{D\rho}{Dt} \tag{4.23}$$

The thermodynamic relation

$$\left(\frac{\partial e}{\partial \rho}\right)_T = -\frac{T}{\rho^2}\left(\frac{\partial p}{\partial T}\right)_\rho + \frac{p}{\rho^2}$$

Is used to obtain the following

$$\frac{De}{Dt} = c_v\frac{DT}{Dt} + \left[-\frac{T}{\rho^2}\left(\frac{\partial p}{\partial T}\right)_\rho + \frac{p}{\rho^2}\right]\frac{D\rho}{Dt} \tag{4.24}$$

Using the continuity equation enables us to insert the term $-{}^{\partial u_j}/_{\partial x_j}$ instead of the substantial derivative of ρ. This relation inserted back into the energy equation that we have derived in the previous section leaves us with the following expression

$$\rho\left(c_v\frac{DT}{Dt}+\left[\frac{T}{\rho}\left(\frac{\partial p}{\partial T}\right)_\rho-\frac{p}{\rho}\right]\frac{\partial u_j}{\partial x_j}\right)+p\frac{\partial u_j}{\partial x_j}=\Phi+\frac{\partial}{\partial x_j}\left[k\frac{\partial T}{\partial x_j}\right]$$

Which is the energy equation solved by TGNet

$$\rho c_v\frac{DT}{Dt}=-T\left(\frac{\partial p}{\partial T}\right)_\rho\frac{\partial u_j}{\partial x_j}+\mu\Phi+\frac{\partial}{\partial x_j}\left[k\frac{\partial T}{\partial x_j}\right] \tag{4.25}$$

4.2.2 Enthalpy form of energy equation

The enthalpy is given by $h=e+p/\rho$. Starting with the continuity equation

$$\frac{D\rho}{Dt}+\rho\frac{\partial u_k}{\partial x_k}=0$$

Rewriting this equation, and adding the pressure gives

$$p\frac{\partial u_k}{\partial x_k}=-\frac{p}{\rho}\frac{D\rho}{Dt}$$

which is equal to the latter term on the left hand side on equation 4.22. Further modifications of the term yields

$$-\frac{p}{\rho}\frac{D\rho}{Dt}=\rho p\frac{D}{Dt}\frac{1}{\rho}=\rho\left[\frac{D}{Dt}\left(\frac{p}{\rho}\right)-\frac{1}{\rho}\frac{Dp}{Dt}\right] \tag{4.26}$$

Inserting the result back into the energy equation gives,

$$\rho\frac{De}{Dt}+\rho\left[\frac{D}{Dt}\left(\frac{p}{\rho}\right)-\frac{1}{\rho}\frac{Dp}{Dt}\right]=\rho\frac{D}{Dt}\left(e+\frac{p}{\rho}\right)-\frac{Dp}{Dt}=\mu\Phi-\frac{\partial q_j}{\partial x_j} \tag{4.27}$$

Where we recognize the term $(e+p/\rho)$ as enthalpy and we can write the expression the following way

$$\rho\frac{Dh}{Dt}-\frac{Dp}{Dt}=\mu\Phi-\frac{\partial q_j}{\partial x_j} \tag{4.28}$$

4.2.3 A closer look on the enthalpy term

The definition of enthalpy states that the enthalpy is equal to the internal energy and the work done on the fluid, $h=e+pv$. Since $e=e(T,p)$, we can write that $h=h(T,p)$. For the substantial derivative of the enthalpy, we can write

$$\frac{Dh}{Dt}=\left(\frac{\partial h}{\partial T}\right)_p\frac{DT}{Dt}+\left(\frac{\partial h}{\partial p}\right)_T\frac{Dp}{Dt} \tag{4.29}$$

We have from the thermodynamic definitions that $\left(\partial h/\partial T\right)_p = c_p$, which is the specific heat at constant pressure. Inserting this back into the left hand side of the enthalpy equation gives the equation presented in [29]

$$\rho\frac{Dh}{Dt} - \frac{Dp}{Dt} = \rho c_p \frac{DT}{Dt} - \left(1 - \rho\left(\frac{\partial h}{\partial p}\right)_T\right)\frac{Dp}{Dt}$$

In section 4.2.1 we introduced a term $-T\left(\frac{\partial p}{\partial T}\right)_\rho \frac{\partial u_j}{\partial x_j}$, which is a term accounting for an effect known as the Joule-Thomson effect. This effect is also accounted for in the term above as

$$\left(1 - \rho\left(\frac{\partial h}{\partial p}\right)_T\right)\frac{Dp}{Dt}$$

This effect will be discussed in section 4.6

4.3 Dissipation term

The dissipation term, $\mu\Phi$, accounts for the rate of which velocity is transformed into thermal energy, resulting in an increase of heat. The dissipation is relatively small, and can in some cases be neglected.

$$\mu\Phi = \lambda\left(\frac{\partial u_k}{\partial x_k}\right)^2 + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_j}{\partial x_i} \tag{4.30}$$

Where λ is associated with the volume expansion. And can be defined by a quantity referred to as bulk viscosity, K. The bulk viscosity is defined by the difference of pressure, p, versus mechanical pressure, $\overline{p} = \frac{1}{3}\left[\sigma_{11} + \sigma_{22} + \sigma_{33}\right]$ in the following relation

$$p - \overline{p} = K\frac{\partial u_k}{\partial x_k} \tag{4.31}$$

The dissipation term however must be simplified in order to fit our one-dimensional solution. And by use of a turbulence model the dissipation term can be approximated as

$$\approx \rho\frac{f}{2 \cdot D}U^3 \tag{4.32}$$

The derivation of this term is tedious, and has not been done in this report.

4.4 Overall heat transfer coefficient

The heat conducted through the pipe wall to the surrounding environment cannot, unlike the dissipation be neglected. This term may become significant in particular if difference between gas and surrounding temperature is large. The amount of heat transferred is dependent on the materials in the pipeline, and the coating. It also depends on the exposure of the pipeline. If the pipeline is exposed to water or soil, will significantly change the total heat transfer coefficient.

An expression for the overall heat transfer coefficient [10] is given as

$$U_i = \frac{1}{\frac{1}{h_i} + \sum \frac{r_i}{k_i} ln\left(\frac{r_i}{r_o}\right) + \frac{r_i}{r_o h_o}} \tag{4.33}$$

This coefficient is defined by the following relation

$$q = UA\triangle T \tag{4.34}$$

The heat transferred from it's surrounding environments is defined by the following three steps

- Surrounding heat to outer wall of pipe
- Heat conduction through the pipe walls
- Heat transfer from the pipe wall to the gas

Pipe wall to gas

In TGNet [17] it is referred to the formula of Dittus and Boelter [7] for computation of Nusselt number for the flow inside the pipeline

$$Nu = 0.023 \cdot Re^{0.8} \cdot Pr^{n} \tag{4.35}$$

where n have the following values

$$n = \begin{cases} 0.4 & \text{heating} \\ 0.3 & \text{cooling} \end{cases} \tag{4.36}$$

This equation is valid for fully turbulent flow in a smooth tube with $0.6 \leq Pr \leq 100$, and $2500 \leq Re_d \leq 1.25 \cdot 10^6$. Since by definition

$$Nu = \frac{h \cdot d}{k},\ Pr = \frac{c_p \mu}{k},\ Re = \frac{\rho u d}{\mu}$$

Where

k	refers to thermal conductivity
d	refers to diameter
h	refers to heat transfer coefficient
c_p	specific heat
μ	kinematic viscosity

Surroundings to wall

Depending on the surroundings of the pipeline the wall of the pipeline will experience different exposure, and consequently different heat transfer coefficients. The assumption is that we have three types of burial of the pipeline.

1. Exposure to seawater: The pipeline if completely exposed to seawater.
2. Shallow burial: The pipeline is buried in soil with the centre of the pipeline deeper than the radius of the pipeline
3. Deep burial: The pipeline is buried deep in the soil.

Depending on the exposure, a different set of formulas must be used to obtain a value for the heat transfer coefficient.

Exposed to seawater

In Langelandsvik's thesis[17], when describing the heat transfer coefficient heat transfer for a pipe exposd to water calculates the Nusselt number using the following formula

$$Nu = 0.26 \cdot Re^{0.6} \cdot Pr^{0.3} \tag{4.37}$$

Where the Nusselt, Prandtl and Reynolds number is defined as above using values for seawater. These values are to be used with respect to the film temperature which is defined as the average temperature between pipewall and seawater.

In a former masterthesis by Klock[15] a somewhat different approach to the heat transfer coefficient of the outside pipe wall, where the total Nusselt number is a combination of Nusselt number due to free convection such as buoyancy effects and forced convection due to the velocity of the seawater.

Shallow burial

When the pipe is considered to be shallow buried, meaning that the depth or the ceterline of the pipe does not go much deeper than the half diameter of the pipeline. For this case we use the following correlation for the heat transfer coefficient

$$h_o = \frac{\frac{2k_{zoil}}{d_{outer}}}{ln\left(x + \sqrt{x^2 - 1}\right)} \tag{4.38}$$

Where the definitions apply

h_o	outer heat transfer coefficient
k_{soil}	thermal conductivity of soil
d_{outer}	Outer diameter of pipeline
x	$\frac{2D_{center}}{d_{outer}}$
D_{center}	Depth to center of pipe

Deep burial

If the pipeline is buried deeper than $\frac{3 \cdot d_{outer}}{2}$ we consider the pipeline to be deep buried and we introduce a different correlation for the heat transfer coefficient. This coefficient is given by

$$h_{outer} = \frac{\frac{2 \cdot k_{soil}}{d_{outer}}}{ln\left({}^{4 \cdot D_{center}}/_{d_{outer}}\right)} \tag{4.39}$$

Using the same definitions as above.

Wall thermal resistance

When a pipe consists of several layers, it is convenient to use a general expression for the thermal resistance through the entire wall. We start by writing the Fourier's law in cylindrical form

$$q_r = -kA\frac{dT}{dr} = -k(2\pi rL)\frac{dT}{dr} \tag{4.40}$$

Integrating this from inner to outer gives

$$\frac{q_r}{2\pi L}ln\left(\frac{r_{outer}}{r_{inner}}\right) = k\left(T_{inner} - T_{outer}\right) \tag{4.41}$$

And the thermal resistance becomes

$$R = \frac{ln\left({}^{r_{outer}}/_{r_{inner}}\right)}{2\pi kL} \tag{4.42}$$

Since the terms that make up the area is cancelled in the total overall heat transfer coefficient that we have presented in section 4.4, the equation if we have several layers of pipeline coating is

$$R_{wall} = \sum_i \frac{r_i}{k_i}ln\left(\frac{r_{outer}}{r_{inner}}\right) \tag{4.43}$$

Discussions of the heat transfer term

In order to obtain values for the overall heat transfer coefficient we need values for conductivity of seawater, soil, pipewalls and gas. Values for these parameters, except thermal conductivity of gas, were found in Langelandsvik [17] and Klock [15]. Thermal conductivity for gas is a rather complex property, and no simple method was found. Values for pure gas is available to a larger extent[18] and since the gas composed mainly of methane, values for pure methane has been approximated to temperature and pressure.

In the analysis Langelandsvik used the following values

Inner diameter	1.00	m
Wall thickness	0.001	m
Ambient temperature	5	^oC
Ground conductivity	2.0	$\frac{W}{mK}$
Sea water velocity	0.1	$\frac{m}{s}$

Table 4.1: Properties in Langelandsvik[17]

Performing a test of the total heat transfer coefficient with the following properties for gas:

- Gas flow rate: 615.92 $\left[\frac{kg}{s}\right]$
- Pressure: 185.4776 $[bar]$
- Heat capacity: 1683.9 $\left[\frac{J}{kgK}\right]$
- Viscosity: $1.9686 \cdot 10^{-5}$ $\left[\frac{N}{m^2s}\right]$
- Seawater thermal conductivity [2] 0.580 $\left[\frac{W}{mK}\right]$
- Gas conductivity [18] 0.0519 $\left[\frac{W}{mK}\right]$

And the results of this test yields

Dept	Ambient temperature	$U_{W,tot}$ thin pipe	$U_{W,tot}$ coated pipe	$U_{W,tot}$from [17]
3.0	5^oC	1.6101	1.6313	1.61
2.0	5^oC	1.9242	1.9545	1.94
1.5	5^oC	2.2332	2.2741	2.27
1.0	5^oC	2.8867	2.9554	3.03
0.75	5^oC	4.1604	4.6316	4.15
0.501	5^oC	62.9420	22.9521*	59.35
Exposed to water	5^oC	217.4477	30.2882	79.05

* depth of centreline is $1mm$ more that radius of the coated pipe

Table 4.2: Overall heat transfer coefficient values

From table 4.2 it can be seen that values for deep burial fits well to values from [17]. When pipeline is exposed to seawater on the other hand the results deviate significantly from values calculated by TGNet. A test using the formula for forced convection proposed by Klock [15] , first given by Churchill-Bernstein

[5] , neglecting the free convection

$$Nu_{forced} = 0.3 + \frac{0.62 Re_{d_0}^{1/2} Pr^{1/3}}{\left[1 + \left(0.4/Pr\right)^{2/3}\right]^{1/4}} \left[1 + \left(\frac{Re_{d_o}}{282000}\right)^{5/8}\right]^{4/5} \tag{4.44}$$

The values of used in calculation of the different values should be calculated based on the film temperature, given as the average of sea temperature and wall temperature. The resulting Nusselt number based on this formula is $Nu_{CB} = 429.96$ and the simpler equation used by TGNet gives $Nu_{TGNet} = 394.33$. Prandtl number for seawater became 13.13, which is as expected. For calculations later the results from [17]have been used.

4.5 Compressibility

The compressibility is related to the volume expansion of the gas, dv. For a single phase situation , the volume can be considered a function of pressure and temperature [22], $v = v(p, T)$, and the differential becomes

$$dv = \left(\frac{\partial v}{\partial T}\right)_p dT + \left(\frac{\partial v}{\partial p}\right)_T dp \tag{4.45}$$

The two differentials can be related to two thermo dynamical properties, $\frac{1}{v}\left(\frac{\partial v}{\partial T}\right)_p$ and $-\frac{1}{v}\left(\frac{\partial v}{\partial p}\right)_T$which are called coefficient of volume expansion and isothermal compressibility, respectively. Volume expansion term expresses the rate at which a volume expands with temperature given a constant pressure. The isothermal compressibility refers to the rate of change of volume doe to change of pressure. This value will always be positive, meaning that the increase of pressure will always decrease the substances volume.

Considering the isothermal compressibility

$$\kappa = -\frac{1}{v}\left(\frac{\partial v}{\partial p}\right)_T \tag{4.46}$$

And introduction of the real gas law in equation 2.2 to the term above gives

$$\kappa_g = -\frac{p}{ZRT}\left(\frac{\partial}{\partial p}\left(\frac{ZRT}{p}\right)\right)_T = \frac{1}{p} - \frac{1}{Z}\left(\frac{\partial Z}{\partial p}\right)_T \tag{4.47}$$

Her we have introduced the subscript g to the isothermal compressibility. Trube [26] introduced the concept of pseudo reduced compressibility, κ_r as a function of reduced pressure

$$\kappa_r = \kappa_g p_c = \frac{1}{p_r} - \frac{1}{Z}\left(\frac{\partial Z}{\partial p_r}\right)_{T_r} \tag{4.48}$$

This relationship gave a direct connection to Standing Katz compressibility factor [25] as a function of reduced pressure and pressure.

4.5.1 Standing Katz compressibility factor

The compressibility factor accounts for the rate at which the gas volume is different from the ideal gas state. The compressibility factor is a function of reduced pressure, p_r and reduced temperature, T_r. Standing and Katz [25] showed that the compressibility curve of several different gasses coincide closely when plotted along the same axes. This is also known as *the principle of corresponding states*. The resulting plot is shown below.

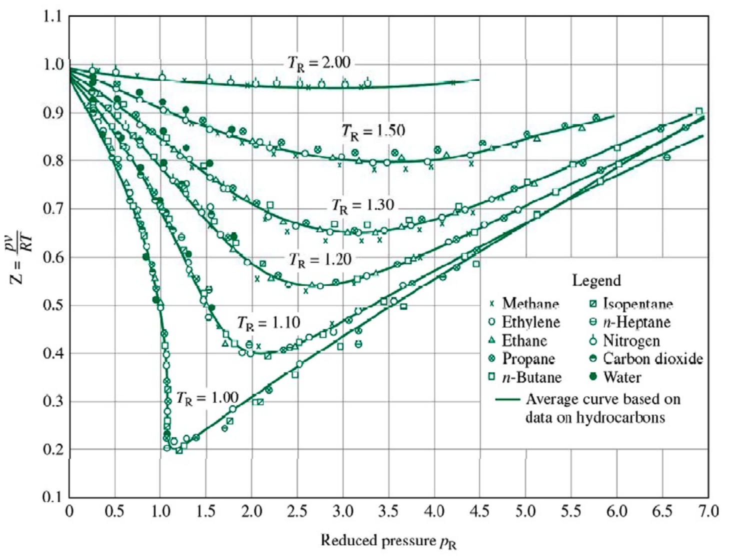


Figure 4.1: Generalized compressibility chart for different gases[22]

4.5.2 Compressibility factor by Gopal

Gopal [14] found a popular straight line fit for the Standing-Katz chart in the form

$$Z = p_r\left(AT_r + B\right) + CT_r + D \tag{4.49}$$

Where the values of the factors A, B, C and D where different depending on the reduced pressure and temperature. This gave a set of 13 equations presented below

Reduced pressure range,$[p_r]$	Reduced temperature range, $[T_r]$	Resulting equation
0.2 to 1.2	1.05 to 1.2	$p_r\left(1.6643T_r - 2.2114\right) - 0.3647T_r + 1.4385$
	1.2 to 1.4	$p_r\left(0.5222T_r - 0.8511\right) - 0.0364T_r + 1.0490$
	1.4 to 2.0	$p_r\left(0.1291T_r - 0.2988\right) + 0.0007T_r + 0.9969$
	2.0 to 3.0	$p_r\left(0.0295T_r - 0.0825\right) + 0.0009T_r + 0.9967$
1.2 to 2.8	1.05 to 1.2	$p_r\left(-1.357T_r + 1.4942\right) + 4.6315T_r - 4.7009$
	1.2 to 1.4	$p_r\left(0.1717T_r - 0.3232\right) + 0.5869T_r + 0.1229$
	1.4 to 2.0	$p_r\left(0.0984T_r - 0.2053\right) + 0.0621T_r + 0.8580$
	2.0 to 3.0	$p_r\left(0.0211T_r - 0.0527\right) + 0.0127T_r + 0.9549$
2.8 to 5.4	1.05 to 1.2	$p_r\left(-0.3278T_r + 0.4752\right) + 1.8223T_r - 1.9036$
	1.2 to 1.4	$p_r\left(-0.2521T_r + 0.3871\right) + 1.6087T_r - 1.6635$
	1.4 to 2.0	$p_r\left(-0.0284T_r + 0.0625\right) + 0.4714T_r - 0.0011$
	2.0 to 3.0	$p_r\left(0.0041T_r + 0.0039\right) + 0.0607T_r + 0.7927$
5.4 to 15.0	1.05 to 3.0	$p_r\left(0.711 + 3.66T_r\right)^{-1.4667} - \frac{1.637}{0.319T_r + 0.522} + 2.071$

Table 4.3: Standing Katz by Gopal[14]

The resulting plot for Gopals best fit for Standing Katz compressibility factor is shown in figure 4.5.2

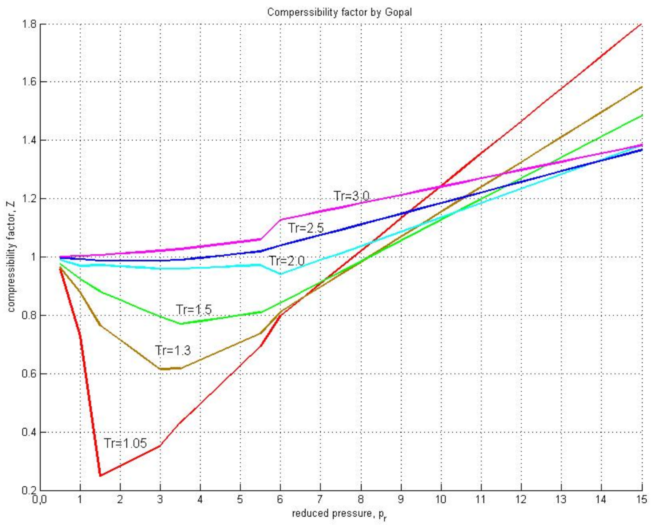


Figure 4.2: Standing Katz by Gopal

4.5.3 Dranchuck, Purvis and Robinson

A different method presented by Mattar & Brar et al. [21] originally developed by Dranchuk, Purvis and Robinson [24], was based on a Benedict-Webb-Rubi type of equation. This method is based on a best-fit approach to the factors of Standing Katz. The Z-correlation is given by

$$Z = 1 + \left[a_1 + \frac{a_2}{T_{pr}} + \frac{a_3}{T_{pr}^3}\right]\rho_r + \left[a_4 + \frac{a_5}{T_{pr}}\right]\rho_r^2 + \left[\frac{a_5 a_6}{T_{pr}}\right]\rho_r^5 + \left[\frac{a_7}{T_{pr}^3}\rho_r^2\left(1 + a_8\rho_r^2\right) EXP\left(-a_8\rho_r^2\right)\right] \tag{4.50}$$

A1=0.31506237	A2=-1.0467099	A3 = -0.57832729
A4 = 0.53530771	A5 = -0.61232032	A6 = -0.10488813
A7 = 0.68157001	A8 = 0.68446549	

And the value of the reduced gas density, ρ_r is given by the relation

$$\rho_r = \frac{0.27 p_{pr}}{Z \cdot T_{pr}} \tag{4.51}$$

This method is applicable under some restrictions for temperature and pressure. The restrictions are

$1.05 \leq T_{pr} \leq 3.0$

$0.2 \leq p_{pr} \leq 3.0$

Differentiating this term with respect to temperature, which is necessary for the calculation of the Joule Thomson term in the next section, we get

$$\left(\frac{\partial Z}{\partial T}\right)_p = \left(\frac{\partial Z}{\partial T_{pr}}\right)_{p_{pr}} \left(\frac{\partial T_{pr}}{\partial T}\right) \tag{4.52}$$

The derivative of reduced value is simple and gives $({}^{\partial T_{pr}}/{}_{\partial T}) = {}^{1}/{}_{T_{critical}}$, and since $Z = Z\left(T_{pr,}\rho_r\left(T_{pr}, p_{pr}\right)\right)$ we get

$$\begin{aligned}\left(\frac{\partial Z}{\partial T_{pr}}\right)_{p_{pr}} &= a_1\rho_r' + a_2\left(\rho_r' - \frac{\rho_r}{T_{pr}}\right)\frac{1}{T_{pr}} + a_3\left(\rho_r' - 3\frac{\rho_r}{T_{pr}}\right)\frac{1}{T_{pr}^3} \\ &+2a_4\rho_r\rho_r' + a_5\left(2\rho_r\rho_r' - \frac{\rho_r^2}{T_{pr}}\right)\frac{1}{T_{pr}} + a_5a_6\left(5\rho_r^4\rho_r' - \frac{\rho_r^2}{T_{pr}}\right)\frac{1}{T_{pr}} + \\ a_7\rho_r&\left[2\rho_r' - \frac{\rho_r}{T_{pr}} - 2a_8\rho_r'\rho_r^2\right]\frac{e^{-a_8\rho_r^2}}{T_{pr}} + a_7a_8\rho_r^3\left[4\rho_r' - \frac{\rho_r}{T_{pr}} - 2a_8\rho_r'\rho_r^2\right]\frac{e^{-a_8\rho_r^2}}{T_{pr}}\end{aligned} \tag{4.53}$$

The term $\rho_r' = \left({}^{\partial\rho_r}/{}_{\partial T_{pr}}\right)_{p_{pr}}$ which becomes

$$\rho_r' \equiv \left(\frac{\partial\rho_r}{\partial T_{pr}}\right)_{p_{pr}} = \left(\frac{\partial}{\partial T_{pr}}\right)_{p_{pr}}\left(\frac{0.27p_{pr}}{ZT_{pr}}\right)$$

$$= -\frac{0.27\cdot p_{pr}}{ZT_{pr}^2} - \frac{0.27\cdot p_{pr}}{Z^2T_{pr}}\left(\frac{\partial Z}{\partial T_{pr}}\right)_{p_{pr}} = -\rho_r\left(\frac{1}{T_{pr}} + \frac{1}{Z}\left(\frac{\partial Z}{\partial T_{pr}}\right)_{p_{pr}}\right) \tag{4.54}$$

4.5.4 Dranchuk and Abou-Kassem

This 8-factor model also gave the background for an 11-factor model developed by Dranchuk and Abu-Kassem [8] of the form

$$Z = 1 + \left[a_1 + \frac{a_2}{T_{pr}} + \frac{a_3}{T_{pr}^3} + \frac{a_4}{T_{pr}^4} + \frac{a_5}{T_{pr}^5}\right]\rho_r + \left[a_6 + \frac{a_7}{T_{pr}} + \frac{a_8}{T_{pr}^2}\right]\rho_r^2$$

$$-a_9\left[\frac{a_7}{T_{pr}} + \frac{a_8}{T_{pr}^2}\right]\rho_r^5 + a_{10}\left(1 + a_{11}\rho_r^2\right)\frac{\rho_r^2}{T_{pr}^3}e^{-a_{11}\rho_r^2} \tag{4.55}$$

where $\rho_r$is given the same way as above.

$a_1 = 0.3262$	$a_2 = -1.0700$	$a_3 = -0.5339$	$a_4 = 0.01569$
$a_5 = -0.05165$	$a_6 = 0.5475$	$a_7 = -0.7361$	$a_8 = 0.1844$
$a_9 = 0.1056$	$a_{10} = 0.6134$	$a_{11} = 0.7210$	

This method is applicable for a larger pressure range

$0.2 \leq p_{pr} \leq 30$

$1.0 \leq T_{pr} \leq 3.0$

Its derivative with respect to T_{pr} becomes

$$\left(\frac{\partial Z}{\partial T_{pr}}\right)_{p_{pr}} = a_1\rho_r' + a_2\left(\rho_r' - \frac{\rho_r}{T_{pr}}\right)\frac{1}{T_{pr}} + a_3\left(\rho_r' - 3\frac{\rho_r}{T_{pr}}\right)\frac{1}{T_{pr}^3} +$$

$$a_4\left(\rho_r' - 4\frac{\rho_r}{T_{pr}}\right)\frac{1}{T_{pr}^4} + a_5\left(\rho_r' - 5\frac{\rho_r}{T_{pr}}\right)\frac{1}{T_{pr}^5} + 2a_6\rho_r\rho_r' + a_7\rho_r\left(2\rho_r' - \frac{\rho_r}{T_{pr}}\right)\frac{1}{T_{pr}} +$$

$$+2a_8\rho_r\left(\rho_r' - \frac{\rho_r}{T_{pr}}\right)\frac{1}{T_{pr}^2} - a_7a_9\rho_r^4\left(5\rho_r' - \frac{\rho_r}{T_{pr}}\right)\frac{1}{T_{pr}} - a_8a_9\rho_r^4\left(5\rho_r' - \frac{\rho_r}{T_{pr}}\right)\frac{1}{T_{pr}^2} +$$

$$a_{10}\rho_r\left[2\rho_r' - 3\frac{\rho_r}{T_{pr}} - 2a_{11}\rho_r^2\rho_r'\right]\frac{e^{-a_{11}\rho_r^2}}{T_{pr}^3} + a_{10}a_{11}\rho_r^3\left[4\rho_r' - 3\frac{\rho_r}{T_{pr}} - 2a_{11}\rho_r^2\rho_r'\right]\frac{e^{-a_{11}\rho_r^2}}{T_{pr}^3} \tag{4.56}$$

Where the definition of reduced density,ρ_r' applies as before. The resulting plot for the compressibilityfactor by Dranchuk and Abou-Kassem is presented in figure 4.5.4

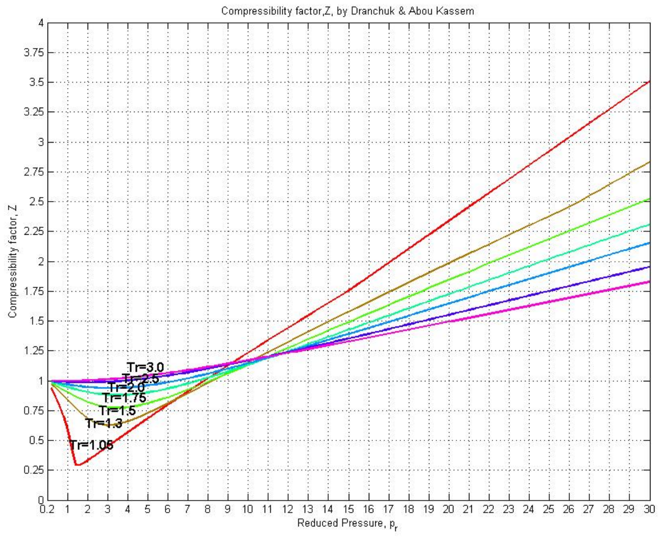


Figure 4.3: Dranchuk & Abou Kasseem

The numerics of the solution of Dranchuk and Abou Kassem's correlation, is solved in a similar matter as done by [12], where the solution is solved with respect to $\rho_r Z T_r$ and a Newton type of linearization. The compressibility factor is then found from the relation of reduced density.

$$Z = \frac{0.27 p_r}{\rho_r T_r} \tag{4.57}$$

4.6 Joule Thomson effect

The definition of the Joule-Thomson effect states that

$$\mu_{JT} = \left(\frac{\partial T}{\partial p}\right)_h \tag{4.58}$$

As presented the Joule Thomson coefficient is defined in terms of dependent variables and can therefore itself be considered a property[22]. This coefficient can also be related to the form

$$\mu_{JT} = -\frac{1}{c_p}\left(\frac{\partial h}{\partial p}\right)_T \tag{4.59}$$

This can be shown by following

$$\left(\frac{\partial h}{\partial p}\right)_T \left(\frac{\partial p}{\partial T}\right)_h \left(\frac{\partial T}{\partial h}\right)_p = -1$$

$$\left(\frac{\partial h}{\partial p}\right)_T = -\frac{1}{\left(\frac{\partial p}{\partial T}\right)_h \left(\frac{\partial T}{\partial h}\right)_p} = -\left(\frac{\partial T}{\partial p}\right)_h \left(\frac{\partial h}{\partial T}\right)_p \tag{4.60}$$

Since we define the specific heat capacity with constant pressure as $c_p = \left({}^{\partial h}/_{\partial T}\right)_p$ we can write

$$\mu_{JT} = \left(\frac{\partial T}{\partial p}\right)_h = -\frac{1}{c_p}\left(\frac{\partial h}{\partial p}\right)_T \tag{4.61}$$

In order to obtain a good relation for the Joule-Thomson coefficient we need to introduce some basic fundamental equations of thermodynamics. The Tds equation [22] gives

$$Tds = du + pdv \tag{4.62}$$

and since enthalpy is defined as

$$h = u + pv \tag{4.63}$$

It's derivative can be written as

$$dh = du + pdv + vdp \tag{4.64}$$

or $dh - vdp = du + pdv$, and inserted back into equation 4.62 gives

$$dh = Tds + vdp \tag{4.65}$$

Our result can then be divided by dp at constant temperature T

$$\left(\frac{\partial h}{\partial p}\right)_T = T\left(\frac{\partial s}{\partial p}\right)_T + v \tag{4.66}$$

Using one of Maxwells relations $\left(\frac{\partial s}{\partial p}\right)_T = -\left(\frac{\partial v}{\partial T}\right)_p$ we get

$$\left(\frac{\partial h}{\partial p}\right)_T = -T\left(\frac{\partial v}{\partial T}\right)_p + v \tag{4.67}$$

Introducing the equation for derivative of enthalpy 4.29, multiplied by dt

$$dh = \left(\frac{\partial h}{\partial p}\right)_T dp + \left(\frac{\partial h}{\partial T}\right)_p dT = \left(\frac{\partial h}{\partial p}\right)_T dp + c_p dT \tag{4.68}$$

And from the equation we recognize the term $\left({}^{\partial h}/_{\partial p}\right)_T$ so that we can write the equation as

$$dh = \left[-T\left(\frac{\partial v}{\partial T}\right)_p + v\right] dp + c_p dT \tag{4.69}$$

Dividing the equation above with dp and assuming constant enthalpy we obtain

$$\left(\frac{\partial T}{\partial p}\right)_h = \frac{1}{c_p}\left[T\left(\frac{\partial v}{\partial T}\right)_p - v\right] \tag{4.70}$$

Which we recognize from equation 4.58. v in this expression represents the specific volume of the fluid, denoted by $\left[{}^{m^3}/{}_{kg}\right]$, and is the equivalent of ρ^{-1}. We have the equation for a real gas given by

$$p = \rho ZRT \tag{4.71}$$

Hence the expression for ρ^{-1} becomes

$$v = \frac{1}{\rho} = \frac{ZRT}{p} \tag{4.72}$$

Inserting this into equation4.70

$$\left(\frac{\partial T}{\partial p}\right)_h = \frac{1}{c_p}\left[T\left(\frac{\partial}{\partial T}\left(\frac{ZRT}{p}\right)\right)_p - \frac{ZRT}{p}\right]$$

And since $Z = Z(T, p)$ and the derivative in the first term cancels the second term we are left with [20]

$$\mu_{JT} = \left(\frac{\partial T}{\partial p}\right)_h = \frac{RT^2}{pc_p}\left(\frac{\partial Z}{\partial T}\right)_p \tag{4.73}$$

Results of the Joule Thomson coefficient

The 11-factor model by Dranchuk-AbouKassem has been used to calculate the Joule Thomson coefficient.

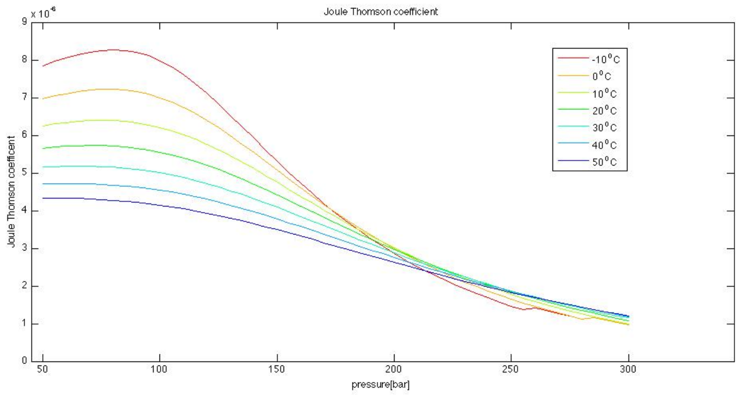


Figure 4.4: Joule Thomson Coefficient μ_{JT} for natural gas

The results shows similarities to results obtained by [20], but no verification of the term is been done.

4.7 Dissipation

in order to obtain a value for the loss of energy as a result of dissipation, we assumed the dissipation function to be

$$\mu\Phi \simeq \rho\frac{f}{2D}U^3 \tag{4.74}$$

In this term, f represents the friction factor found by the famous equation of Colebrook and White [27]

$$\frac{1}{\sqrt{f}} = -2.0 \cdot log\left(\frac{\epsilon}{3.7 \cdot D} + \frac{2.51}{Re\sqrt{f}}\right) \tag{4.75}$$

Which must be solved by an iterative approach since the term $1/\sqrt{f}$ is represented on both sides of the equation. However, the results of the frictional loss under steady state conditions of the pipeline shows that the numeric's have a tendency of over-predicting the loss of pressure due to the wall friction, and for TGNet [17] the friction factor is adjusted with a factor that is dependent on the pipeline, and other parameters.

$$\frac{1}{\sqrt{f}} = -2.0 \cdot log\left(\frac{\epsilon}{3.7 \cdot D} + \frac{2.51}{Re\sqrt{f}}\right) \cdot EFF \tag{4.76}$$

The term EFF is supposed to account for additional drag effects. As the term increases the friction factor is reduced. And dependent on the pipeline and different cases, the term varies between

$$0.95 \leq EFF \leq 1.05 \tag{4.77}$$

4.8 Specific heat capacity

In order to solve the equations one needs a correlation for specific heat capacity of gas. The correlations used is the same as the one presented in TGNet[17].For isobaric heat capacity one uses,

$$c_p = 1.432 \cdot 10^4 - 1.045 \cdot 10^4 \cdot SG + 3.255 \cdot T + 10.01 \cdot SG \cdot T + EXP \tag{4.78}$$

Where EXP is defined as

$$EXP = \frac{15.69 \cdot 10^{-2} \cdot p^{1.106} \cdot e^{-6.203 \cdot 10^{-3}}}{SG} \tag{4.79}$$

A correlation for ratios of specific heat is also given

$$\frac{c_v}{c_p} = 1.03836 - 0.000115 + \frac{5.61 - 0.002 \cdot T}{M} \tag{4.80}$$

However, these values are given in unknown units and according to [1] , a general,multiplication factor of ~ 0.16 gives an accurate results in terms of SI-units.

Chapter 5

Solution of the energy equation

5.1 Introduction

The energy equation must be solved along with the corresponding momentum and pressure equations. These equations however, do not have the same characteristic, and therefore cannot be solved at the same place and time simultaneously. The characteristics of the characteristic equations derived previously are equal to the wavespeed. In the gas-transport problem we have subsonic flow, and $B \gg U$. Therefore in a graphic representation, we se as follows.

Therefore an interpolation must be done in order to obtain the result at the correct time or distance. We also see that $\triangle x - B\triangle t \ll \triangle x - U\triangle t$. A closer look at the temperature reveals that $\partial p/\partial t$ is rather small, and a remedy for the interpolation problem, we can use the fact that the energy equation does not indeed need to be resolved at each time step.

5.2 Characteristic representation of the energy equation

The enthalpy equation stated

$$\rho c_p \frac{dT}{dt} - \left(1 - \rho \left(\frac{\partial h}{\partial p}\right)_T\right) \frac{dp}{dt} = \phi - \frac{4 \cdot U_{W,tot}}{D}\left(T - T_{env}\right) \tag{5.1}$$

Rewriting the equation using $\phi = \rho \frac{f}{2 \cdot D} U^3$, $-\rho \left(\frac{\partial h}{\partial p}\right)_T = \rho c_p \mu_{JT}$, results in

$$\rho c_p \frac{dT}{dt} - \left(1 + \rho c_p \mu_{JT}\right) \frac{dp}{dt} = \rho \frac{f}{2 \cdot D} U^3 - \frac{4 \cdot U_{W,tot}}{D}\left(T - T_{env}\right) \tag{5.2}$$

In order to obtain a stable solution we assume constant density and specific heat over the integration. And we can therefore multiply both sides of the equation by $\frac{1}{\rho c_p}$ which gives us

$$\frac{dT}{dt} - \left(\frac{1}{\rho c_p} + \mu_{JT}\right)\frac{dp}{dt} = \frac{1}{2 \cdot c_p}\frac{f}{D}U^3 - \frac{1}{\rho c_p}\frac{4 \cdot U_{W,tot}}{D}\left(T - T_{env}\right) \tag{5.3}$$

Then, since this is represented in a characteristics manner, the equation holds along $\frac{dx}{dt} = U$. Therefore, by multiplying the equation by $dt = \frac{dx}{U}$ we get

$$dT - \left(\frac{1}{\rho c_p} + \mu_{JT}\right)dp = \frac{1}{2 \cdot c_p}\frac{f}{D}U^3\frac{dx}{U} - \frac{1}{\rho c_p}\frac{4 \cdot U_{W,tot}}{D}\left(T - T_{env}\right)\frac{dx}{U} \tag{5.4}$$

If we then consider the right hand side of our equation

$$\frac{1}{2 \cdot c_p}\frac{f}{D}U^3\frac{dx}{U} - \frac{1}{\rho c_p}\frac{4 \cdot U_{W,tot}}{D}\left(T - T_{env}\right)\frac{dx}{U} = \frac{1}{2 \cdot c_p}\frac{f}{D}U^2 dx - \frac{1}{\rho c_p}\frac{4 \cdot U_{W,tot}}{D}\frac{1}{U}\left(T - T_{env}\right)dx \tag{5.5}$$

Since U and ρ can be written in terms of dependent variables T, M and p, using the definitions of velocity and equation of state of real gas

$$U = \frac{M}{\rho A}$$

$$\rho = \frac{p}{ZRT}$$

The dissipation term on the right hand side then becomes

$$\frac{1}{2 \cdot c_p}\frac{f}{D}U^2 dx = \frac{1}{2 \cdot c_p}\frac{f}{D}\left(\frac{M}{\rho A}\right)^2 dx = \frac{(ZRT)^2}{2 \cdot c_p}\frac{f}{DA^2} \cdot \frac{M^2}{p^2}dx \tag{5.6}$$

And the heat transfer term can be written as

$$\begin{aligned}\frac{1}{\rho c_p}\frac{4 \cdot U_{W,tot}}{D}\frac{1}{U}\left(T - T_{env}\right)dx &= \frac{4}{\rho c_p}\frac{U_{W,tot}}{D}\frac{\rho A}{M}\left(T - T_{env}\right)dx \\ &= \frac{4}{c_p}\frac{A}{D}\frac{U_{W,tot}}{M}\left(T - T_{env}\right)dx = \frac{1}{c_p}\frac{U_{W,tot}}{M}\left(T - T_{env}\right)\pi D dx\end{aligned} \tag{5.7}$$

On our left hand side we have the term

$$dT - \left(\frac{1}{\rho c_p} + \mu_{JT}\right)dp$$

which by insertion of dependent variables p and T, becomes

$$dT - \left(\frac{ZRT}{pc_p} + \mu_{JT}\right)dp = dT - \frac{ZRT}{c_p}\frac{dp}{p} - \mu_{JT}dp$$

And we then end up wit ha final equation

$$dT - \left(\frac{ZRT}{pc_p} + \mu_{JT}\right)dp = \frac{1}{2}\frac{(ZR)^2}{c_P} \cdot \frac{f}{DA^2} \cdot \frac{T^2}{p^2}M^2 dx - \frac{1}{c_p}\frac{U_{W,tot}}{M}\left(T_{gas} - T_{env}\right)\pi D dx \tag{5.8}$$

Where we have the Joule Thomson coefficient represented with dp.

5.2.1 Integrating the characteristic representation

Taking all terms over to the left hand side setting the equation equal to zero gives

$$dT - \left(\frac{ZRT}{pc_p} + \mu_{JT}\right)dp - \frac{1}{2}\frac{(ZR)^2}{c_P}\cdot\frac{f}{DA^2}\cdot\frac{T^2}{p^2}M^2dx + \frac{1}{c_p}\frac{U_{W,tot}}{M}\left(T_{gas} - T_{env}\right)\pi Ddx = 0 \tag{5.9}$$

This equation is then integrated along it's characteristic $\frac{dx}{dt} = U$, from point A to point P.

$$\int_A^P dT - \int_A^P \frac{ZRT}{c_p}\frac{dp}{p} - \int_A^P \mu_{JT}dp - \int_A^P \left[\frac{1}{2}\frac{(ZR)^2}{c_P}\cdot\frac{f}{DA^2}\cdot\frac{T^2}{p^2}M^2\right]dx + \int_A^P \left[\frac{1}{c_p}\frac{U_{W,tot}}{M}\left(T_{gas} - T_{env}\right)\pi D\right]dx = 0 \tag{5.10}$$

The first term is straight forward to integrate. For the other terms simplifications mut be done. We have the following situation, as

$Z = Z(T, p)$ which means that Z is a function of both temperature and pressure and will therefore change along the pipe.

$c_p = c_p\left(T, p, Mw\right)$ so heat capacity is also a function of the

$\mu_{JT} = \mu_{JT}\left(T, c_p, p, \left(\partial Z/\partial T\right)_p\right)$ and will as a consequence change along the slope

$U_{W,tot} = U_{W,tot}\left(U, burial, .\rho, \mu, c_p\right)$ The overall heat transfer coefficient depends on velocity, gas state and pipeline situation.

The average value of T and p is used, and the two latter terms is written as

$$\int_A^P \left[\frac{1}{2}\frac{(ZR)^2}{c_P}\cdot\frac{f}{DA^2}\cdot\frac{T^2}{p^2}M^2dx\right]dx = \frac{1}{2}\frac{(Z_{avg}R)^2}{c_{P,avg}}\cdot\frac{f}{DA^2}\cdot\left(\frac{T_P + T_A}{p_P + p_A}\right)^2\cdot\left(\frac{M_P + M_A}{2}\cdot\left|\frac{M_P + M_A}{2}\right|\right)\triangle x$$

and

$$\int_A^P \left[\frac{1}{c_p}\frac{U_{W,tot}}{M}\left(T_{gas} - T_{env}\right)\pi Ddx\right]dx = \frac{2}{c_{p,avg}}\frac{U_{W,tot}}{M_P + M_A}\left(T_{gas} - T_{env}\right)\pi D\triangle x$$

On our left han side we have

$$dT - \frac{ZRT}{c_p}\frac{dp}{p} - \mu_{JT}dp$$

Which shall be integrated from A to P.

$$\int_A^P dT - \int_A^P \frac{ZRT}{c_p}\frac{dp}{p} - \int_A^P \mu_{JT}dp = (T_P - T_A) - \frac{ZRT_{avg}}{c_p}ln\left(\frac{p_P}{p_A}\right) - \mu_{JT}\left(p_P - p_A\right) \tag{5.11}$$

Here we have used the assumption that T is taken as the average value in the second term, and that the Joule Thomson coefficient is the is the coefficient based on average values

$$\mu_{JT} \equiv \frac{RT_{avg}^2}{p_{avg}c_p}\left(\frac{\partial Z}{\partial T}\right)_p \tag{5.12}$$

And the total expression for the energy equation becomes

$$(T_P - T_A) - \frac{Z_{avg}R\left(T_P + T_A\right)}{2\cdot c_p}ln\left(\frac{p_p}{p_A}\right) - \mu_{JT}\left(p_p - p_A\right)$$

$$-\frac{1}{2}\frac{(Z_{avg}R)^2}{c_{p,avg}}\cdot\frac{f}{DA^2}\cdot\left(\frac{T_P+T_A}{p_p+p_A}\right)^2\cdot\left(\frac{M_P+M_A}{2}\right)^2\triangle x+\frac{2}{c_{p,avg}}\frac{U_{W,tot}}{M_P+M_A}\left(\frac{T_A+T_P}{2}-T_{env}\right)\pi D\triangle x=0 \tag{5.13}$$

Her we have used $T_{gas} = T_{avg}$.

5.2.2 Newton Rhapson linearization

Newton Rhapson linearization is a method used to solve equations of the form

$$f(x) = 0$$

The method is described in the project, and can be found in the litterature [16]. For a single equation system, the solution can be written as

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

Which means that we have to obtian a derivative of equation 5.13 to obtain the Newton-linearized form the equation. The derivative is found with respect to T_P. The first two terms in the expression is rather straight forward to take it's derivative. The latter two terms becomes

$$\frac{d}{dT_P}\left[\frac{1}{2}\frac{(Z_{avg}R)^2}{c_{p,avg}}\cdot\frac{f}{DA^2}\cdot\left(\frac{T_P+T_A}{p_p+p_A}\right)^2\cdot\left(\frac{M_P+M_A}{2}\cdot\left|\frac{M_P+M_A}{2}\right|\right)\triangle x\right]$$

$$=\frac{(Z_{avg}R)^2}{c_{p,avg}}\cdot\frac{f}{DA^2}\cdot\left(\frac{T_P+T_A}{p_p+p_A}\right)\cdot\left(\frac{M_P+M_A}{2}\cdot\left|\frac{M_P+M_A}{2}\right|\right)\triangle x\cdot\frac{1}{p_p+p_A}$$

$$=\frac{(Z_{avg}R)^2}{c_{p,avg}}\cdot\frac{f}{DA^2}\cdot\frac{T_P+T_A}{(p_p+p_A)^2}\cdot\left(\frac{M_P+M_A}{2}\cdot\left|\frac{M_P+M_A}{2}\right|\right)\triangle x$$

The value of $Z_{avg}$is then said to be constant, and the term $\frac{\partial Z}{\partial T}$ is considered to be negliglible. The heat transfer term becomes

$$\frac{2}{c_{p,avg}}\frac{U_{W,tot}}{M_P+M_A}\left(\frac{T_A+T_P}{2}-T_{env}\right)\pi D\triangle x=\frac{1}{c_{p,avg}}\frac{U_{W,tot}}{M_P+M_A}\pi D\triangle x$$

Derivation of the Joule-Thomson term is not so straight-forward as the other terms

$$\frac{\partial\mu_{JT}}{\partial T_p}=\frac{\partial\mu_{JT}}{\partial T_{avg}}\frac{\partial T_{avg}}{\partial T_p}=\frac{RT_{svg}}{pc_p}\left(\frac{\partial Z}{\partial T}\right)+\frac{RT_{avg}^2}{2pc_p}\left(\frac{\partial^2 Z}{\partial T^2}\right) \tag{5.14}$$

but if we consider the last term to be negligible due to the term $\left(\frac{\partial^2 Z}{\partial T^2}\right)$, the derivative of the Joule Thomson coefficient becomes

$$\frac{\partial\mu_{JT}}{\partial T_p}=\frac{RT_{svg}}{pc_p}\left(\frac{\partial Z}{\partial T}\right)_p \tag{5.15}$$

In the equation above Z, c_p, T and μ_{JT} are kept as constant. These values can be calculated by using average values, i.e.

$T = T_{avg} = \frac{1}{2}\left(T_P + T_A\right)$

$p = p_{avg} = \frac{1}{2}\left(p_P + p_A\right)$

$c_p = c_{p,avg} = c_p\left(T_{avg}, p_{avg}\right)$

$Z = Z\left(T_{avg}, p_{avg}\right)$

$\mu_{JT} = \mu_{JT}\left(T_{avg}, p_{avg}, c_{p,avg}\right)$

The functional derivative with respect to T_p then becomes

$$f'(T_p) \equiv T_p - \frac{Z_{avg}R}{2 \cdot c_p} ln\left(\frac{p_p}{p_A}\right) - \frac{\partial \mu_{JT}}{\partial T_p}\left(p_p - p_A\right)$$

$$-\frac{\left(Z_{avg}R\right)^2}{c_{p,avg}} \cdot \frac{f}{DA^2} \cdot \frac{T_P + T_A}{\left(p_p + p_A\right)^2} \cdot \left(\frac{M_P + M_A}{2}\right)^2 \triangle x + \frac{1}{c_{p,avg}} \frac{U_{W,tot}}{M_P + M_A} \pi D \triangle x = 0 \quad (5.16)$$

Hence the final form of the equation we want to solve becomes

$$T_p^{n+1} = T_p^n - \frac{\left(T_P - T_A\right) - \frac{Z_{avg}R\left(T_P + T_A\right)}{2 \cdot c_p} ln\left(\frac{p_p}{p_A}\right) - \mu_{JT}\left(p_p - p_A\right)}{T_p - \frac{Z_{avg}R}{2 \cdot c_p} ln\left(\frac{p_p}{p_A}\right) - \frac{\partial \mu_{JT}}{\partial T_p}\left(p_p - p_A\right)} \cdots$$

$$\cdots \frac{-\frac{1}{2}\frac{\left(Z_{avg}R\right)^2}{c_{p,avg}} \cdot \frac{f}{DA^2} \cdot \left(\frac{T_P + T_A}{p_p + p_A}\right)^2 \cdot \left(\frac{M_P + M_A}{2} \cdot \left|\frac{M_P + M_A}{2}\right|\right) \triangle x + \frac{2}{c_{p,avg}} \frac{U_{W,tot}}{M_P + M_A}\left(\frac{T_A + T_P}{2} - T_{env}\right) \pi D \triangle x}{-\frac{\left(Z_{avg}R\right)^2}{c_{p,avg}} \cdot \frac{f}{DA^2} \cdot \frac{T_P + T_A}{\left(p_p + p_A\right)^2} \cdot \left(\frac{M_P + M_A}{2} \cdot \left|\frac{M_P + M_A}{2}\right|\right) \triangle x + \frac{1}{c_{p,avg}} \frac{U_{W,tot}}{M_P + M_A} \pi D \triangle x} \quad (5.17)$$

Chapter 6

New program

In the previous program only pressure and flow rate were solved, based on isothermal assumption. The isothermal assumption and simplification of the equations resulted in that variables such as density, compressibility factor, speed of sound and other were kept constant. The equations also neglected the convective terms, of pressure and flow rate. The new program also solves the energy equation together with the characteristic equations for pressure and flow rate. Therefore the flow can no longer be assumed isothermal.

When the energy equation was solved and temperature no longer could be assumed constant, variables such as density and speed of sound also changes. This would of course have to be accounted for in the new code. However the equations are solved using the method of characteristics and time steps are set as a result of the CFL condition. The characteristic equations calculating pressure and flow rate, have a characteristic equal to plus/minus the speed of sound, however characteristic of the energy equation is equal to the velocity. Since $B \gg U$, the equations could not necessarily be solved at the same time, without major interpolations, which is likely to be a significant source of error. And since the temperature gradients are relatively small one does not need to solve the energy equation at each time step. Therefore as a remedy for this, the energy equation is solved when $\triangle x - \sum_i U_i \triangle t_i \leq 0$, where i represents a number of time steps. When the energy equation is solved, the value of i is reset, and the temperature is kept constant until the next time the sum of characteristic and time step of the energy equation is 'big enough' to almost reach the specified $\triangle x$.

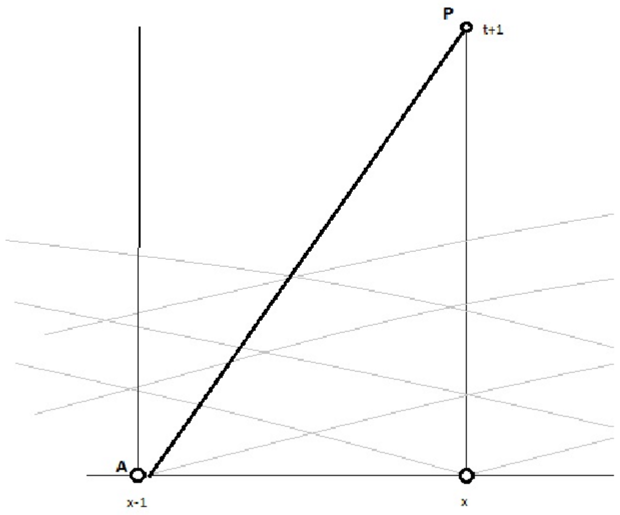


Figure 6.1: Characteristic of energy equation

Figure 6 represents the development of the characteristic of the energy equation compared to the characteristics of the equations for pressure and flow rate, represented in gray lines.

6.1 Modification of the old code

The program initiates a class called fluid. This class previously contained only pressure and flow rate. This was done since all other properties were kept constant. This is no longer the case and the new class fluid contains the following entities

fluid.	variable	size
temperature	Tempertaure	$1x306$
flowrate	Mass flow rate	$1x306$
pressure	Pressure	$1x306$
Z	compressibility factor	$1x306$
density	Density of fluid	$1x306$
velocity	Velocity of fluid	$1x306$
enthalpy	Enthalpy	$1x306$
time	Time new time step	$1x1$

Table 6.1: Entities fluid class in code

Modifications of the old *solver* involves calculating frictionfactor based on the node's previous value, and the wavespeed is used as the wavespeed at the old time step. In addition an 'oldFluid' class is initiated in the main-function. This is done since wavespeed no longer is considered constant, and the characteristics no longer intersects the same point in the xt-plane, and we interpolate at the nodes to find the exact value in space. The timestep is determined based on the highest wavespeed, and other characteristics therefore hits the previous time step somewhere in between $[x_{i-1}, x_i]$ or $[x_i, x_{i+1}]$. The values at the old time step are also interpolated in space, since the characteristics are unequal. At the end of the solver-function new properties for the class fluid is calculated. All properties except flow rate, pressure and temperature are calculated for the new fluid as a result of the recent calculated pressure and flow rate.

Chapter 7

Simulations of the flow in a gas transport pipeline

7.1 Simulations done by Gassco

The main purpose of this work is to be a part of the road to provide more accurate numerical models for computation of natural gas transport. Previous computational model have proved inaccurate in certain cases and the question is why these results deviates as much from measured values as they do. A plot was provided by Gassco to show some of the resulting errors commercial software models provided

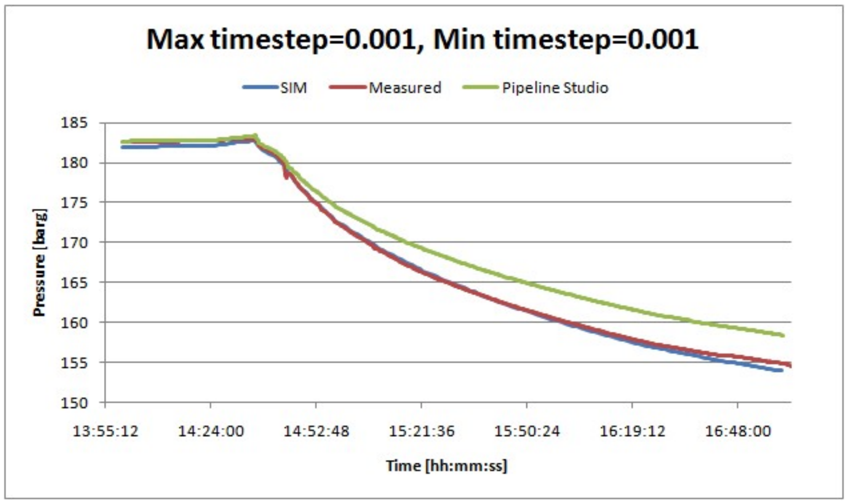


Figure 7.1: Simulation by Gassco

The line referring to 'Pipeline studio' is the same as TGNet, which has been described earlier. From the figure above, one sees that the old TGNet simulator gives a too high pressure at the inlet. The new simulator however proves more accurate, but tends to give too low pressure at the end of the transient.

7.2 Model of pipeline

Simulations has been done considering the section from Kårstø to Vestre-Bokn. This part is amongst others crossing a fjord and goes above an island, making the elevation of the pipeline rather significant. The model of this pipeline is the same as used in the previous project and was approximated from a figure of the section found in Langelandsvik's thesis[17]. The profile of the pipeline is shown below

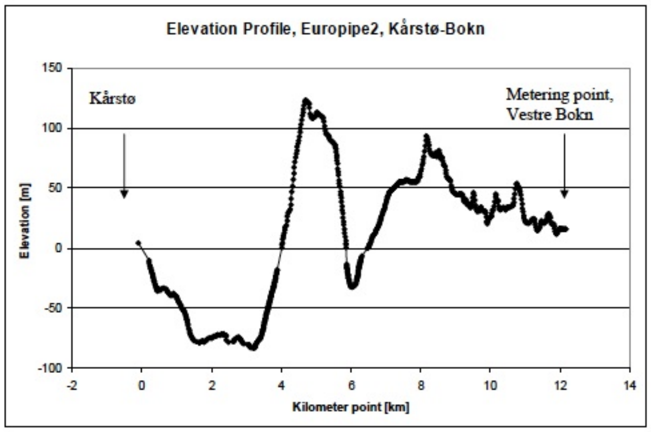


Figure 7.2: Elevation of Kåstø Bokn section of Europipe II[17]

The pipeline was thereby divided into 61 sections of 200 metres each, and the elevation was approximated at 62 points. The sections contain a certain number of nodes. All nodes in a section is given the same angle of attack with respect to gravity force.

Since no data is given on the situation with respect to the surrounding material and burial of the pipeline along the slope, a set of situations for the pipe burial had to be guessed. The pipeline was guessed to have 4 possible states.

1. Exposed to water
2. Shallow burial at half diameter + 1 mm.
3. Deep burial 1 metres.
4. A small section exposed to air of coated pipe

For the pipeline itself, 3 situations has been used in the simulations

Situation	**Description**
Exposed pipeline	A thin pipeline with only 1mm thick walls is completely exposed to water in when below sea level. The first section above sea the heat transfer has been approximated by a shallow burial coefficient and the last section is approximated by deep burial
Buried pipeline	The same thin pipeline as above, but now it's partially buried much of the time when below seawater.
Coated pipeline	A pipeline coated with an inner layer of steel, then a layer of asphalt and finally concrete. When in seawater the pipe is exposed to water and close to surface shallow buried. In the end a small section exposed to air has been tested (this resulted in very low heat transfer and could be considered negligible).

Table 7.1: Description of three pipeline situations

7.3 Boundary- and initial conditions

7.3.1 Steady state boundary conditions

For the steady state calculations, we keep the mass flux constant at the inlet. The value used at the inlet is the first value of flow rate given in the specific problem at the starting time. At the outlet the pressure is kept constant as the pressure at the outlet calculated by the initial equations. This could of course have been done the other way around, with flowrate at the outlet and constant pressure at the inlet. Temperature must also be given at the inlet.

One implication with the assumption of constant flow rate at the inlet, is that it uses a constant value througout the pipeline according to the isothermal assumption, with constant compressibility factor and wavespeed. When these results are imposed on the equations using variable temperature, and the other properties vary correspondingly, we don't have stable conditions inside the pipeline immediately. This stabilizes however rather quickly as velocity and density are being adjusted.

7.3.2 Transient boundary conditions

For the transient cases we use the mass flow rate measured by Gassco[11] at the inlet for the boudary condition at the inlet. For the outlet we use the measured pressure at the outlet. This however is not all straight forward. The sampling of the data in [11] is not done systematically. The flowrates have mainly been sampled each minute pluss minus a few seconds, but in most cases some of the sampling datas have been left out. The pressure and temperatures on the other hand have not been sampled systematically, with equal time intervals. This represents a problem, since the date- and time stamp on the sample data is in an Excel-format that MATLAB was unable to read. Therefore an algorithm was written to fit the data to consistent predetermined intervals.

This meant that for flow rate, the sampling rates were set to be exact at each minute. Therefore the method fills inn the holes of the data by choosing a sequence of random number and inserting the average of the values in front and behind, making the sample data consistent. For the pressure, the data were tested with respect to number of samplings in the given time interval. From there one took the closest value the number of samplings each minute to fit the sampling rate was found, i.e. 4 samplings a minute gives a pressure reading each 15 seconds and a total of 240 values each hour. The method adapts the data sampling to the required number of data to reconstruct a sampling using a consistent time interval. This is mainly done with deleting a number of random values in the vector so that the final input and output vectors becomes consistent.

7.3.3 Initial conditions

We need initial values for all dependent variables, temperature, pressure and mass flow rate. The initial values used for pressure is the one that was used in the previous project and by Streeter and Wylie [4]. This uses the steady state equations for the simplified model using constant temperature, speed of sound and compressibility. This is of course not correct but its relatively good, and after some time the initial equations will no longer affect the solution. For temperature the initial values of temperature distribution may affect the solution if the initial temperature distribution is very inaccurate from computed. This will result in high temperature gradients, and will create instabilities in the flow and oscillating waves in the solution that will affect the result until the characteristic of the temperature has reached the outlet. Therefore steady-state simulations to obtain good initial conditions for the temperature were done.

The flow rate is set equal to the flow rate at the inlet for the entire pipeline. However, the value of the flow rate is then not corrected according to the actual temperature reduction, with it's variables such as compressibility and wave speed. This affects the solution of flow rate and it will immediately oscillate, with maximum amplitude of approximately $1\,kg/s$. After short time this oscillation stabilize.

7.4 Steady state simulations

In order to verify the model developed, simulations in a steady state situation were made. As boundary conditions for this part we kept the flow rate at the inlet constant and equal to the first value and the pressure were kept constant, with the value found at the calculation of initial conditions. The initial flow rate inside the pipe is a constant value, as the previous program. For the new simulation model, the value of flow rate must stabilize by adjusting density and velocity accordingly.

7.4.1 Initial Temperature Calculations

In order to start our simulations, we need an initial distribution of the temperature inside the pipeline. Using steady state conditions such as fixed flow rate at the inlet and a continuous pressure at the outlet, we calculated the temperature distribution along the pipeline.

7.4.1.1 Contribution due to Joule Thomson effect

The Joule Thomson effect is the effect of temperature changes due to pressure changes without proper heat transfer. We have an overall pressure drop due to friction in our 12,5 km long pipeline, and the temperature due to the Joule Thomson effect should decrease to a certain degree. Final results of temperature calculations setting both overall heat transfer coefficient and friction factor zero show that the total temperature drop due to the Joule Thomson effect, $\triangle T_{JT}$, is approximately $1.5^{o}C$

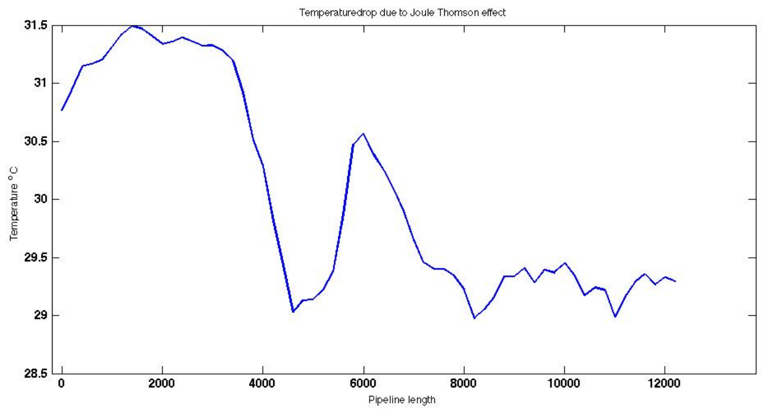


Figure 7.3: Temperature due to Joule Thomson

The temperature follows the hydrostatic pressure, and the inverse elevation of the pipeline.

7.4.1.2 Dissipation

Due to friction in the pipeline and turbulent flow some of the fluids energy will be transformed from momentum energy to thermal energy, also known as dissipation. This will increase the temperature of the gas by a relative small amount.

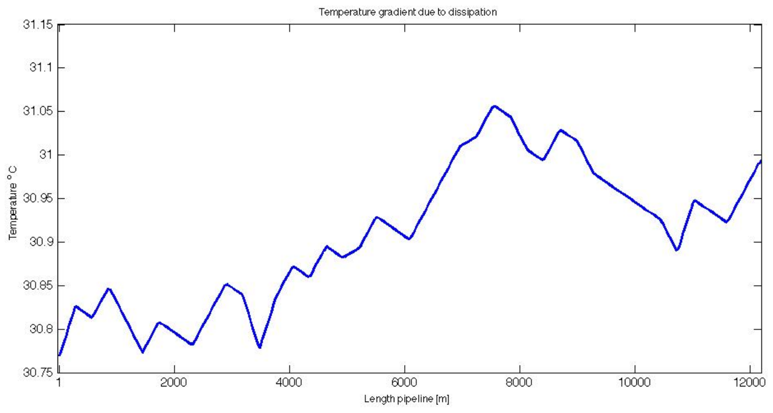


Figure 7.4: Dissipation effects on temperature

As we clearly see from the figure the increase of temperature as a result of dissipation very small and could in many cases be neglected. The total increase of temperature due to dissipation is approximately $\triangle T_{\phi} = 0.25^{o}C$.

7.4.2 Temperature calculations at steady state conditions

A steady state simulation to calculate temperature with the three different cases for pipeline described in table 7.2. The exposed pipeline gives a too high drop of temperature, and for buried pipe we see that since most of the section below water is buried we do not get the steep temperature gradient immediately. The case of the coated pipeline seems to have a more reasonable temperaturedistribution, also compared to pig-measurements of temperature in [17].

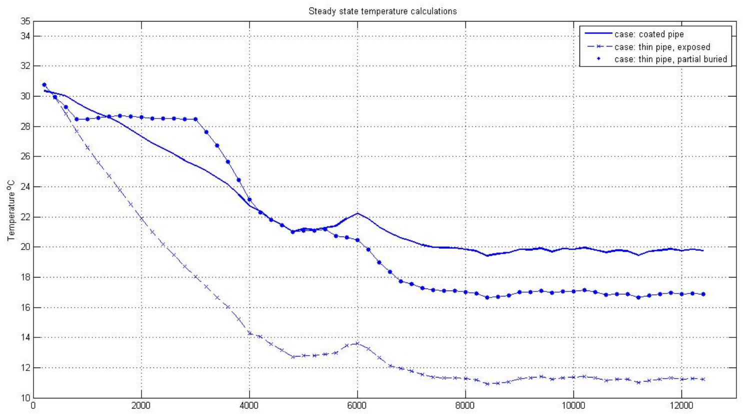


Figure 7.5: Steady state temperature

The temperature distribution of the coated pipeline seems to give a relatively good correspondance with 'pig-measured' temperatures found in [17], and it has therefore been the preffered situation of the pipeline.

7.4.3 Variables at steady state

For the case of steady state the distribution of the different paramters were also evaluated.

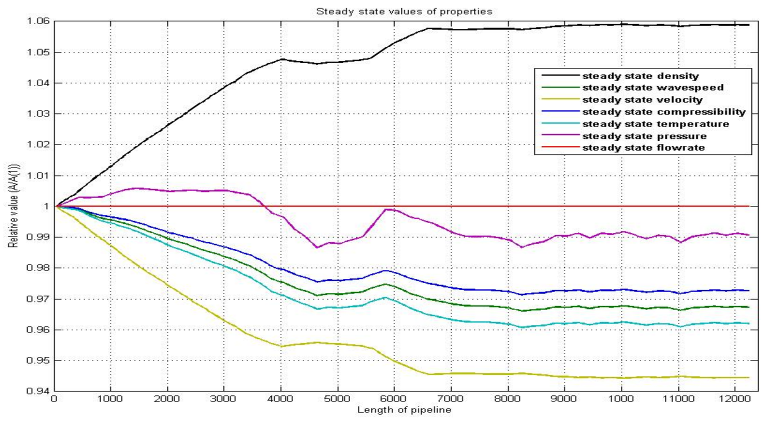


Figure 7.6: Variables at steady state

From this chart we see that the pressure is the most significant parameter, which is also noticeable when looking at the definition of the properties.

- Velocity is defined $U = \frac{M}{pA} ZRT$
- Density is defined $\rho = \frac{p}{ZRT}$
- Comperssibility is a function of $Z = Z\left(T_r . p_r\right)$
- Wavespeed is defined as $B = \sqrt{ZRT}$

7.5 Transients

7.5.1 The old Matlab program

The old MATLAB program was written in the project of spring 2010. The program used the method of characteristics as presented by [4], without the inertial multiplier. The derivation of the method was done under a set of assumptions

- Isothermal flow
- Expasion of pipe wall negliglible
- Pipeline divided into segments, all having the same inclination.
- Equation of state,$p = \rho ZRT$, where the value of Z is considered to be constant.
- Flow is one-dimensional
- Friction factor assumed a function of Reynolds number and wall roughness, and calculated as steady-state
- Change of kinetic energy neglected.

The equations are derived by means of continuity and momentum, using the state equation, the resulting equations are defined by dependent variables pressure and flow rate. The wavespeed, or speed of sound, is keept constant through the entire problem due to the assumptions made above, $B = \sqrt{\left(\frac{\partial p}{\partial \rho}\right)} = \sqrt{ZRT}$.

The continuity equation becomes

$$p_t + \frac{B^2}{A} M_x = 0 \tag{7.1}$$

The method neglects changes in velocity head, and for the total derivative of velocity we get

$$\frac{dU}{dt} = \frac{\partial U}{\partial t} + U \frac{\partial U}{\partial x} \simeq \frac{\partial U}{\partial t} \equiv U_t \tag{7.2}$$

This result in terms of dependent variables M and p therefore becomes

$$\frac{dU}{dt} \simeq U_t = \frac{1}{A} \left(\frac{M}{\rho}\right)_t = \frac{B^2}{Ap} \left(M_t - \frac{M}{p} p_t\right) \tag{7.3}$$

Using this term with the result in 7.1 and forming a characteristics type of equation results in

$$C^{+}\begin{cases}\frac{1}{A}\frac{dM}{dt}+\frac{1}{B}\frac{dp}{dt}+\frac{pgsin\theta}{B^2}+\frac{fB^2M^2}{2DA^2p}=0\\ \frac{dx}{dt}=B\end{cases}\tag{7.4}$$

and

$$C^{-}\begin{cases}\frac{1}{A}\frac{dM}{dt}-\frac{1}{B}\frac{dp}{dt}+\frac{pgsin\theta}{B^2}+\frac{fB^2M^2}{2DA^2p}=0\\ \frac{dx}{dt}=-B\end{cases}\tag{7.5}$$

These equations are the integrated in a characteristics manner and has been derived by[4]. The final equations that the previous MATLAB program solves are

$$\frac{B}{A}(M_p-M_A)+p_p-p_A+\frac{fB^2\triangle x}{(p_p+p_A)DA^2}\frac{e^s-1}{s}\left(\frac{M_p+M_A}{2}\left|\frac{M_p+M_A}{2}\right|\right)+\frac{p_p^2}{p_p+p_A}(e^s-1)=0\tag{7.6}$$

and

$$\frac{B}{A}(M_p-M_B)-p_pp_B+\frac{fB^2\triangle x}{(p_p+p_B)DA^2}\frac{e^s-1}{s}\left(\frac{M_p+M_B}{2}\left|\frac{M_p+M_B}{2}\right|\right)+\frac{p_B^2}{p_p+p_B}(e^s-1)=0\tag{7.7}$$

Where we have used the following

s	$2g\triangle xsin\theta/B^2$
M	flow rate
B	speed of sound
θ	inclination of particular pipe reach

These equations are then solved using a Newton-Raphson approach.

7.5.2 Simulations using the old program

The old simulations in the previous project were done using constant pressure, calculated by the steady state equations, at the outlet. The results were compared to measured pressure at the inlet. As the valve closes, the flow rate decreases and the corresponding pressure also decrease. As a consequence the pressure at what we refer to as the outlet, will drop as the characteristics reach that point. In fact, what we refer to as the outlet is in reality only a metering station. Therefore, holding the pressure constant at the outlet gives a too high pressure at the inlet.

Case I: shut-down at date 04.02.2009

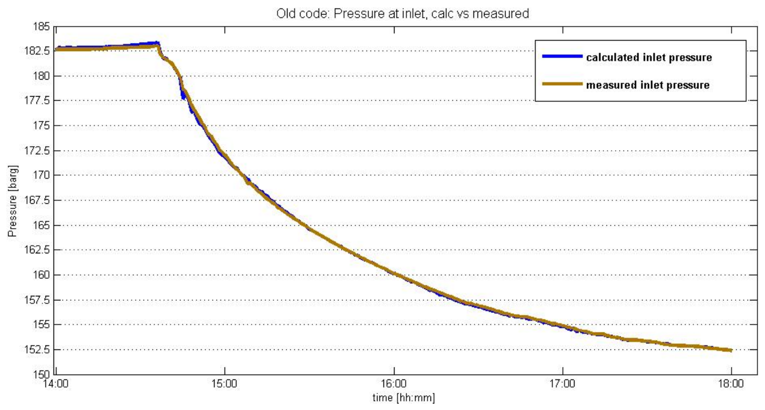


Figure 7.7: Pressure at inlet case I, using the old program

Case II: shut-down at date 04.04.2008

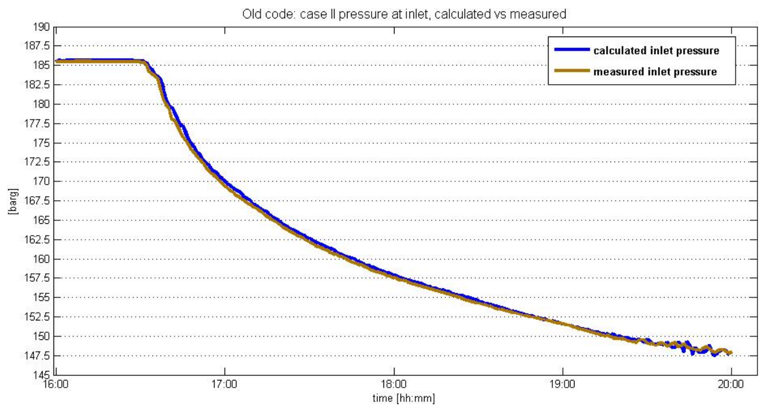
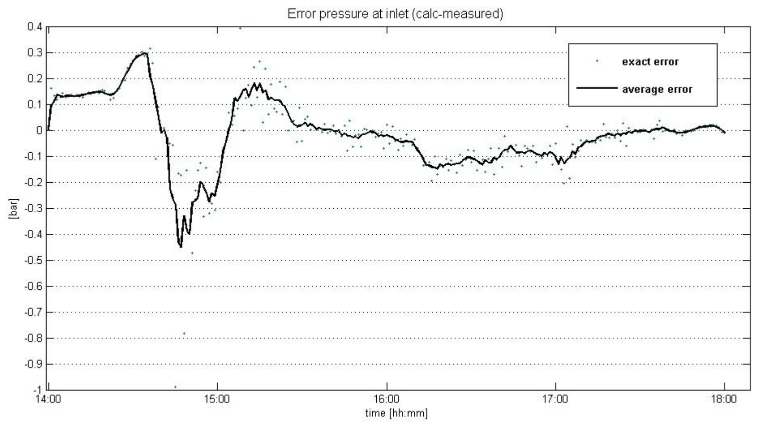


Figure 7.8: Pressure at inlet case II, using the old program

Discussions of the result

The results using the old code give surprisingly accurate results, and more specific evaluation is necessary to obtain a more precise description of the error. The resulting pressure at the inlet were compared to the measured pressure and the result is plotted in figure 7.5.2



* avg error is value at node $i \pm 1, 2$

Figure 7.9: Error case I at the inlet using old program

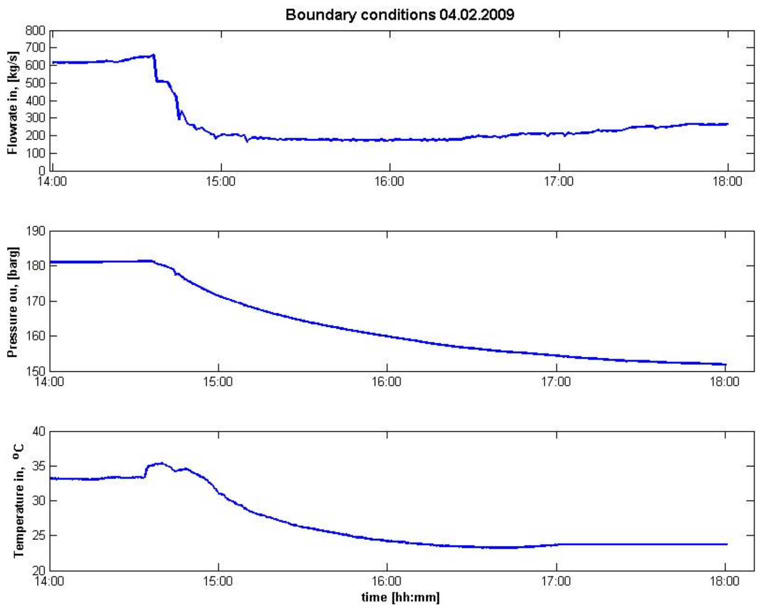
In the old code the friction factor multiplier, EFF, were equal to zero, which results in a too high pressure at the inlet as a result of no drag-forces on the frictionfactor. As a result the inletpressure in the start of the calculation is too high, where as it becomes more accurate as the flowrate and pressure decreases.

7.5.3 Calculations using new program

The derived energy equation and the resulting temperature was then embedded in the existing code. The energy equation is solved as described in chapter 5, and solved when the $\sum U_i \triangle t_i$ were almost equal to $\triangle x$, and held constant until the energy equation was solved again.

Case I, 04.02.2009:

Case I is a case of valve closure. Flowrate decreases from about $620kg/s$ to about $100kg/s$. The flow rate at inlet and pressure at outlet is give below together with the incoming temperature.



* The straight line temperature at the last hour of temperature in is due to an error in the data sheet[11], and the temperature is therefore kept constant

Figure 7.10: Boundary conditions case I, 04.02.2009

The flow rate slowly increases before it drops instantly with about 125 kg/s. The pressure at the outlet starts slowly to decrease as a result of the drop of mass flux. Interestingly we see that the temperature increases close to the point of valve closure before starts to decrease. The simulations are done in a 4 hour perspective, staring at 14:00 and ending at 18:00. The complete results of the simulation is given in the appendices in terms of 3-dimensional plots, but the computed inlet pressure compared to the measured pressure at the inlet is presented below.

Results of simulation:

EFF	Found based on a best fit to be equal to 1.015

The resulting computed pressure compared to the measured pressure at the inlet becomes

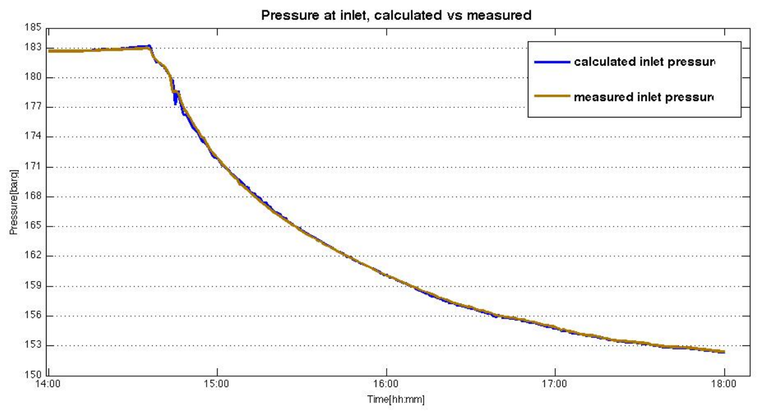
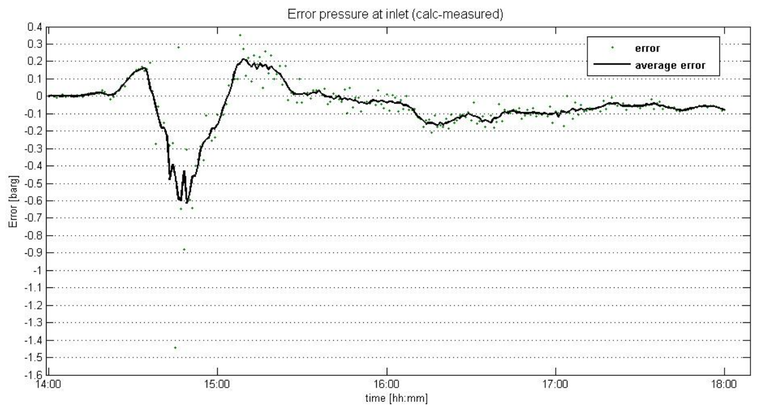


Figure 7.11: Result calculated vs measured at the inlet

The figure above shows the computed results corresponds well to the actual values at the outlet. Unlike the results of Gasscos simulator,TGNet, where the pressure at the inlet becomes to high or too low as the flowrate decreases. It has been debated that this might be due to the friction factor, . However further investigation of the accuracy of the results reveals that the computed pressure using this method is slightly smaller than the actual pressure.

The difference of calculated and measured pressure is presented in figure 7.12



* avg error is value at node $i \pm 1, 2$

Figure 7.12: Calculated pressures and measured, case I

Mean error is based on the value at the node and $i \pm 1, 2$. This is done to give a more precise result of the error since the difficulties with respect to timestamp of the results. The resulting mean absolute error of each node in this case was $e\bar{r}r = 0.1010$.

The result of the temperature calculation can also be tested and compared to measured values, and the values at the inlet.

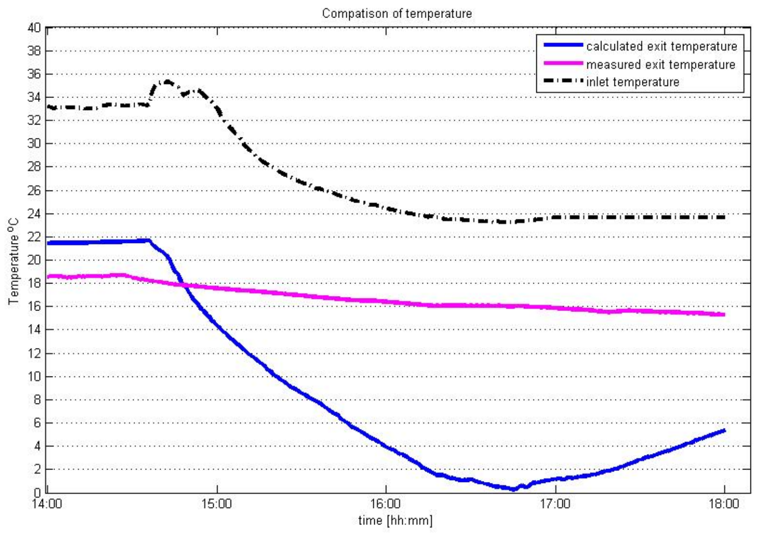
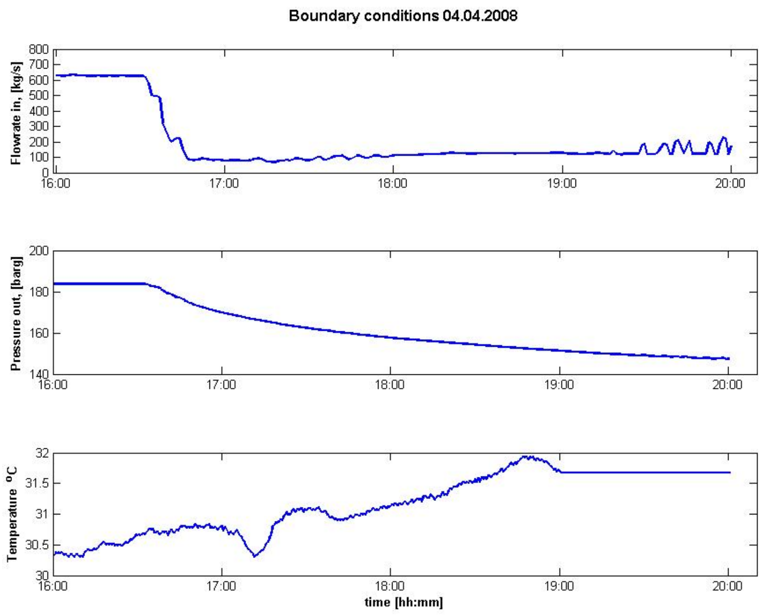


Figure 7.13: Temperatures case I

As clearly can be seen from these figures, the temperature drop in the calculated cases is too large. Where as the differentce $T_{in} - T_{out}$ increases during the transient for the computational case, it actually decreases in the real case.

Case II, 04.04.2008

This case, as before describes a closure of the valve. With a stable flow rate for 3/4 of an hour, and then the flow rate drops significant. As before, the flow does not completely stopped, but it decreases by approximately 80%. The boundary conditions used for this case are given as



* The straight line temperature at the last hour of temperature in is due to an error in the data sheet[11], and the temperature is therefore kept constant

Figure 7.14: Boundary conditions case II, 19.10.2007

Unlike the case before, this case actually have an increasing temperature as the valve closes. This however, can be discussed on how real this actually is. This will be debated in the discussion on errors to come in a later section.

Results of simulation:

EFF	Found based on a best fit to be equal to 1.010

The resulting plot of calculated pressure at the inlet compared to measured values is presented in figure 7.15.

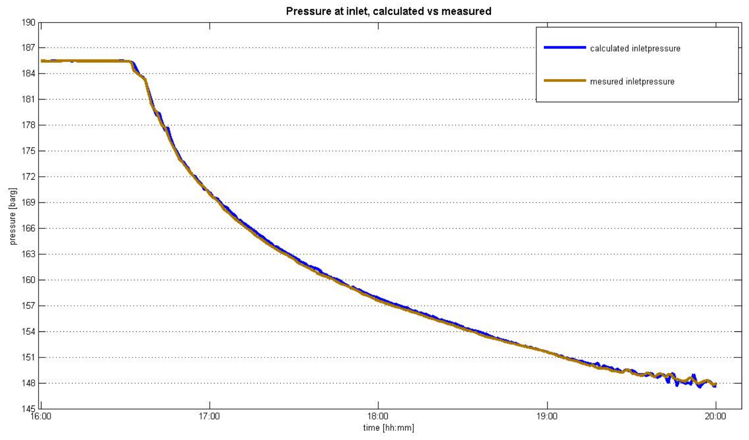
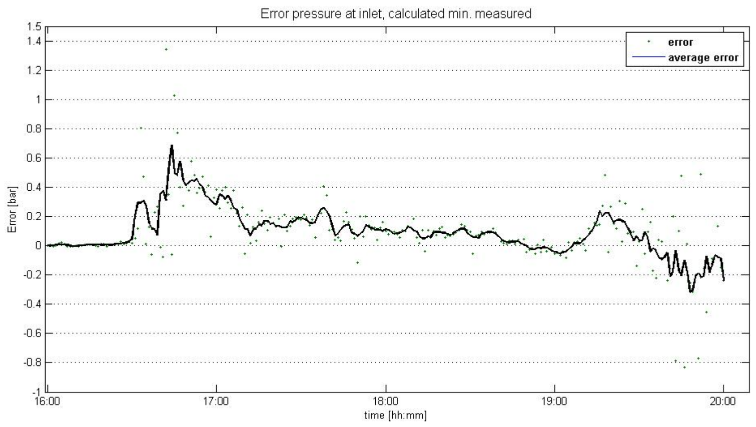


Figure 7.15: Result calculated pressure vs measured pressure at inlet

The pressure, as before, corresponds well, and we don't see any clear deviations on calculated and measured values. Therefore a more detailed analysis of the difference of measured and computed results were done. The error between computed and measured were plotted, and the result was



* avg error is value at node $i \pm 1, 2$

Figure 7.16: Calculated pressures and measured, case II

The mean value of the absolute error becomes for this case $e\bar{r}r = 0.1995$. Which is double the case previous case. Looking at the figure above it becomes obvious that this increase of total error is due to the last hour of the simulations where we get oscillations of both pressures and flowrate. Since this is in the final stage of the computation, the results are likely to become more skewed to the exact results at the given time due to the adaption procedure of the boundary conditions.

As for the temperature, the initial temperature is rather accurate. The temperature drops significantly as a result of the pressure drop an Joule Thomson, but measured temperature drops only slightly.

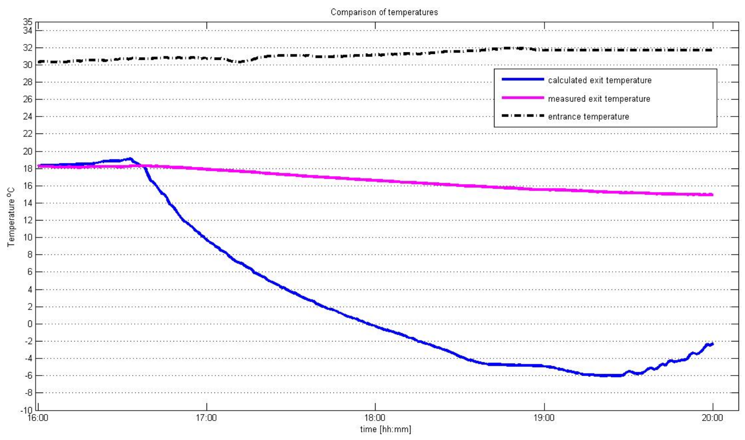


Figure 7.17: Temperatures case II

7.5.4 Startup of flow

Case III, 19.10.2007

The case 19.10.2007 describes an opening of a valve and re-start of the flow. The flow has been shut down for about an hour, and the flow slowly starts up again. The simulation is done for 5 hours, but the flow only less than half of normal flow rate. The boundary conditions are given as

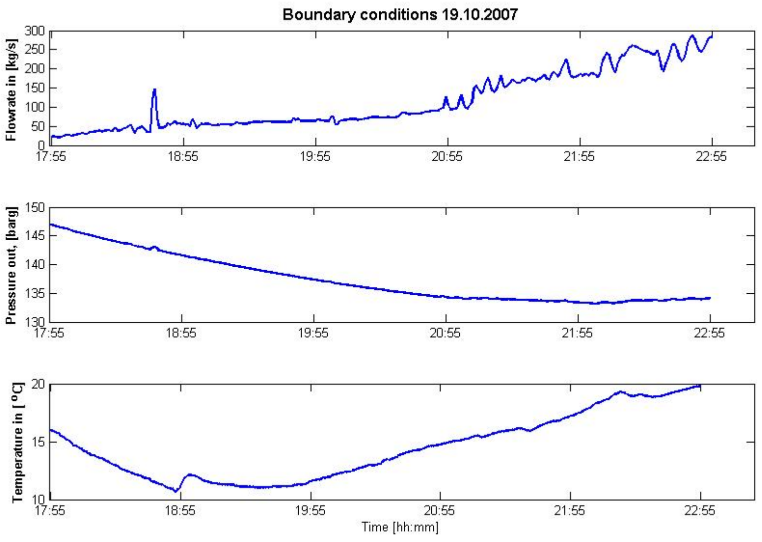


Figure 7.18: Boundary conditions case III, 19.10.2007

Interestingly we see that pressure continues to decrease, but temperature and flowrate increases.

Results of simulation:

EFF Found based on a best fit to be equal to 1.000

The resulting plot of calculated pressure at the inlet compared to measured values is presented below.

The pressure, as before, corresponds well, and we don't see any clear deviations on calculated and measured values. Detailed analysis of the difference of measured and computed results is done. The error between computed and measured were plotted

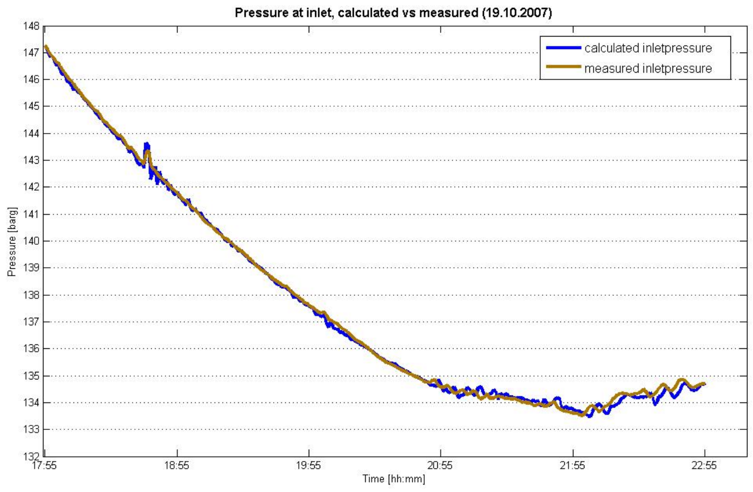
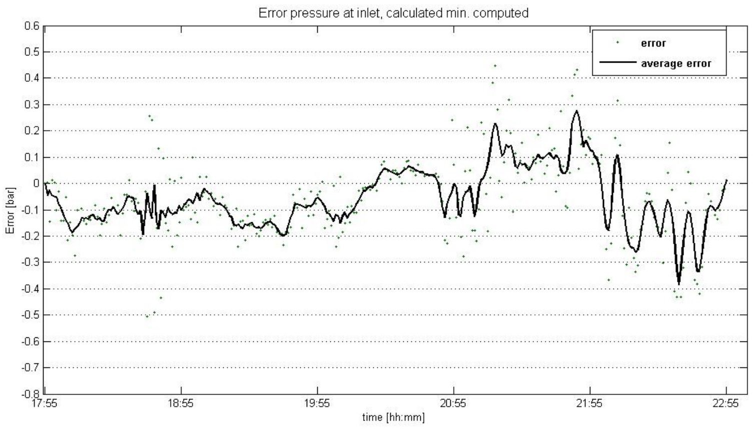


Figure 7.19: Result calculated inletpressure vs measured inletpressure



* avg error is value at node $i \pm 1, 2$

Figure 7.20: Calculated pressures and measured, case III

Mean absolute error in this case became $err = \bar{}0.1124$.

And the resulting temperature distribution shows a rather strange effect. Temperature at the inlet actually goes below the temperature at the outlet, and temperature at the outlet more or less stays constant. As before computed temperature decreases far more than measured values.

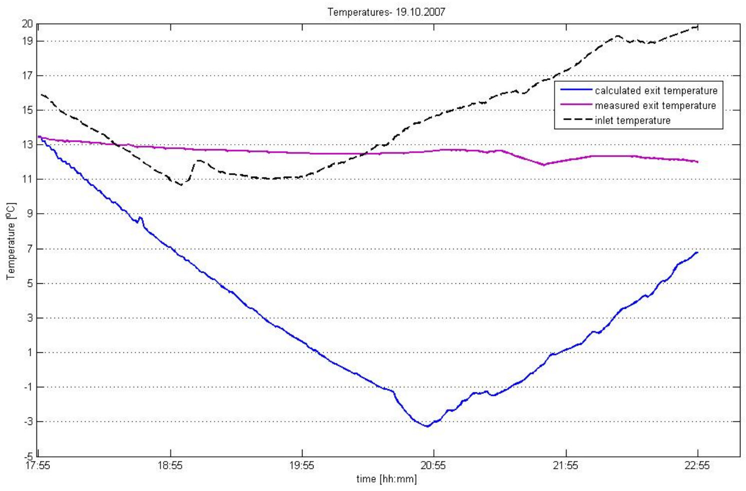


Figure 7.21: Temperatures case III

7.6 Discussions of the results

Friction factor

For the calculations of the frictionfactor the equation of Colebrook and White has been used. This is an iterative formula, but it does not require many iterations to obtain convergence. But the formula is a function of $f = f(f, Re_D, \epsilon)$, where ϵ is the roughness in the pipeline. The value of the roughness previously used was $5 \cdot 10^{-6} m$. This value however seemed to give a too high pressure drop due to friction and the new value of ϵ was set to $3 \cdot 10^{-6} m$, which was also found in similar calculations.

The model uses a correction factor, EFF, to account for the additional drag effects on the gas. When this coefficient is set equal to one the resulting friction factor is exactly equal to the friction factor of Colebrook and White[27]. And increase in the factor gives a reduced friction factor, and increased drag effects, and vice versa. The coefficient is set for each simulation as the value that gave the best and most accurate result of inlet pressure for the initial case. The results show that for the first case calculated inletpressue is too small, when the flow rate is low. As for the second case, the inletpressure seems to be a bit too high. The drag coefficient is set to 1.1015 and 1.010 respectively. The assumption of a constant drag coefficient throughout the entire calculation seems to be wrong. Adjusting the drag coefficient, with respect to flow rate, or Reynolds number and/or change of pressure, would probably give a more accurate result of friction loss along the pipeline.

$$EFF = EFF\left(Re_D, \triangle p\right) \tag{7.8}$$

If this also would help for the accuracy in the transinet is unknown, but one might assume that a more correct real frictionfactor also would improve results in the transients.

Loss of convective term

The equations used to solve for pressure and flow rate are the ones introduced by Wylie et. al [4], and used in the project. This solver neglected the changes in velocity head and thereby the convective term. Yow[28] introduced a remedy for this, by introducing an inertial multiplier. This involves solving the equations similarly as done in the paper, but adjusting the results using a set of equation involving the multiplier. However stability became a major issue using these equations for the method tested, and the inertial multiplier was therefore neglected. The neglecting of the convective term seems to be a reasonable assumption for most situations, but as seen the results in large transients are not sufficiently accurate. The loss of convective term is probably one of the main reasons.

Heat transfer and temperature loss

One of the main uncertainties in this analysis is the situation of the pipeline. No information except the elevation of the pipeline is available, and some information on the pipe is presented at the end of [17]. There the pipe is described as the coated pipe used above, but with slightly different thickness and conductivity. However, unless knowing of the situation regarding exposure to the elements, it becomes difficult to exactly calculate correct temperature drop along the pipeline. Therefore three test cases were done, two cases involving a pipe with no insulation and one coated pipeline. All three cases were tested for case I and the temperatures at the outlet became

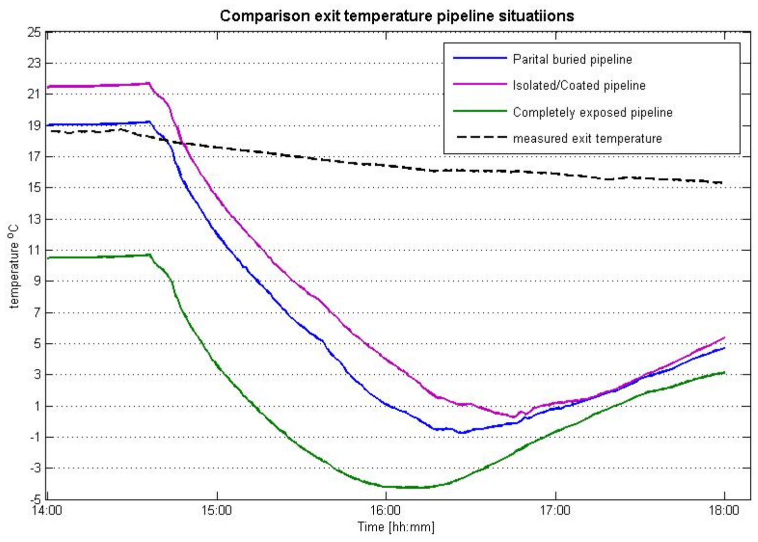


Figure 7.22: Outlet temperature case I, different pipelines

The most accurate results were obtained using the buried pipeline, which was adapted to fit the measured temperature at the outlet. The coated pipeline gave a slightly more inaccurate result, but the magnitude of the difference was in the order of $\backsim 0.005\,bar$ and could therefore only be a coincidence. The completely exposed pipeline gave the biggest error, of approximately $\sim 0.13\,bar$ as the mean absolute error. So we can by this say that the effect of the temperature has a slight impact on the result.

Independent of the pipeline we see that as pressure drops the pressure gradient becomes large and negative the temperature also drops significantly. This is of course due to the Joule Thomson effect, and as the pressure stabilizes the temperature drop is less significant. Interestingly we see that the heat transfer from the surrounding is not high enough to deal with the temperature drop. The lack of seasonal dependent ambient temperature is also a significant source of error, since convective heat transfer is a significant parameter in the calculation of temperature.

7.6.1 Errors

An increase in number of nodes would in most cases increase accuracy. A comparison of error when increasing nodes were done, and the results were

Number of nodes	max$\|err\|$	min$\|err\|$	mean(abs$\|err\|$)
306 nodes	0.3749	-1.4427	0.1010
611 nodes	0.4905	-0.7891	0.0979
1221 nodes	0.5793	-0.7888	0.0932

Table 7.2: Convergence of error

As the table shows, we do not have a clear connection with reduction of error given an increase of nodes by a factor of 2 and the fact that an increase in number of nodes is very costly in terms of computational effort. The error given at each minute becomes

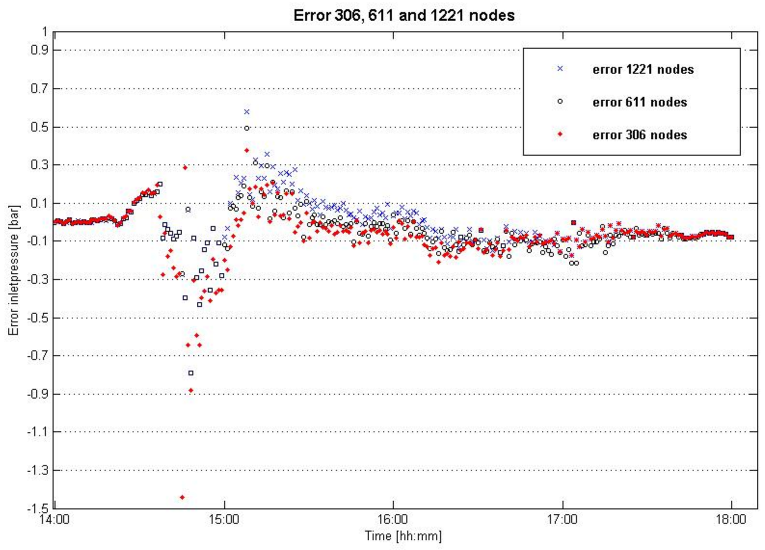


Figure 7.23: Error case I using different number of nodes

The error at relative steady flow rates before and after closure of valve gives a small error, and the difference between different number of nodes. But what clearly can be seen is that the lowest number of nodes gives by far the largest absolute error at the instant the valve closes. One are likely to believe this is due to the loss of convective term, and the fact that a more sparse grid increases the effect of loss of the convective term.

Simplifications of characteristic equations

When deriving the characteristic equations for flow in gas pipeline, Wylie [4] made the simplification that convective term in conservation equation and that velocity head loss were negligible, and introducing the inertial multiplier as a remedy for this. And since velocity head were neglected the total derivative of the term $\dot{U} = U_t + UU_x \approx U_t$. And the convective term in the equation of motion in terms of dependent variables p and M became

$$\frac{dU}{dt} = \frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} \approx \frac{B^2}{Ap}\left(\frac{\partial M}{\partial t} - \frac{M}{p}\frac{\partial p}{\partial t}\right) \tag{7.9}$$

And the resulting equation of conservation became

$$\frac{\partial p}{\partial t} + \frac{B^2}{A}\frac{\partial M}{\partial x} = 0 \tag{7.10}$$

As a consequence the characteristics follows $dx/dt = \pm B$, which is the wave speed. This was done amongst others since velocity inside the pipeline can be considered to be small compared to the speed of sound, or wave speed, $U \ll B$. However, if we derive the equations using complete equations

$$\frac{\partial p}{\partial t} + U\frac{\partial p}{\partial x} + \frac{B^2}{A}\frac{\partial M}{\partial x} = 0 \tag{7.11}$$

Which is the complete continuity equation and the momentum equation becomes.

$$\frac{\partial p}{\partial x} + \frac{1}{A}\left(\frac{\partial M}{\partial t} + U\frac{\partial M}{\partial x}\right) - \frac{M}{pA}\left(\frac{\partial p}{\partial t} + U\frac{\partial p}{\partial x}\right) + \frac{pg}{B^2}sin\theta + \frac{f \cdot B^2M^2}{2 \cdot DA^2p} = 0 \tag{7.12}$$

Where we have simplified by inserting U instead of $\frac{MB^2}{pA}$. Combining these two equations in a characteristics manner using a multiplier

$$C: \lambda \cdot L_1 + L_2 = 0 \tag{7.13}$$

gives

$$\lambda\left[\frac{\partial p}{\partial t} + \left(U + \frac{1}{\lambda}\right)\frac{\partial p}{\partial x}\right] - \frac{M}{pA}\left[\frac{\partial p}{\partial t} + U\frac{\partial p}{\partial x}\right] + \frac{1}{A}\left[\frac{\partial M}{\partial t} + \left(U + \lambda B^2\right)\frac{\partial M}{\partial x}\right] + \frac{pg}{B^2}sin\theta + \frac{f \cdot B^2M^2}{2 \cdot DA^2p} = 0 \tag{7.14}$$

The velocity head loss has been neglected in the previous derivation and we see that the term is small compared to the others since divided by pressure. Neglecting this term and evaluating the term inside the parenthesis we get the result that

$$\lambda = \pm\frac{1}{B} \tag{7.15}$$

And the equation becomes

$$\frac{1}{B}\frac{dp}{dt} + \frac{1}{A}\frac{dM}{dt} + \frac{pg}{B^2}sin\theta + \frac{f \cdot B^2M^2}{2 \cdot DA^2p} = 0 \tag{7.16}$$

which is valid along

$$\frac{dx}{dt} = \begin{cases} U + B & along\, C^{+} characteristic \\ U - B & along\, C^{-} characteristic \end{cases} \tag{7.17}$$

Which corresponds to the result that we came up with in section 3.2.1. For high velocity flows the assumption is obviously wrong, but for very low velocities the assumption is fairly good. The error can be significant for large transients.

7.7 Alternative solutions

7.7.1 Solving the exact equation

When we derived the equation used to solve the energy equation we made the assumption when we integrated the energy equation

$$\rho c_p dT - (1 + \rho c_p \mu_{JT})\, dp = \rho \frac{f}{2 \cdot D} U^3 dt - \frac{4 \cdot U_{W,tot}}{D} (T - T_{env})\, dt \tag{7.18}$$

That the properties were constant at each integral, and given as the average value. The main reason for this assumption was that solving the equations using the method of characteristics became unconditionally unstable when solved with separate values

$$\begin{gathered}(\rho c_p T)_P - (\rho c_p T)_A - ((1 + \rho c_p \mu_{JT})\, p)_P + ((1 + \rho c_p \mu_{JT})\, p)_A \\ -\frac{f}{2 \cdot D} (\rho U^3 t)_P + \frac{f}{2 \cdot D} (\rho U^3 t)_A + \left(\frac{4 \cdot U_{W,tot}}{D} (T - T_{env})\, t\right)_P - \left(\frac{4 \cdot U_{W,tot}}{D} (T - T_{env})\, t\right)_A = 0\end{gathered} \tag{7.19}$$

The reason this is not working for a Newton type of leneariation, is likely to be that the curve changes as the interation process is converging. An alternative algorithm of obtaining the solution was made. The equation itself has a zero value written in terms of $f(T_p) = 0$. And using an iterative method of selecting correct temperature value at point P, where the $f(T_P) = \pm tolerance$ gave a resulting T_p.

7.7.2 Solving the enthalpy equation

Even though the solution of the exact energy equation written in terms of temperature were unstable, the solution of the enthalpy equation were stable given the CFL condition.

$$\rho dh - dp - \rho \frac{f}{2D} U^3 dt + \frac{4U_{W,tot}}{D} (T - T_{env})\, dt = 0 \tag{7.20}$$

Keeping constant density over the integral gives

$$dh - \frac{1}{\rho} dp - \frac{f}{2D} U^3 dt + \frac{1}{\rho} \frac{4U_{W,tot}}{D} (T - T_{env})\, dt = 0 \tag{7.21}$$

Then integrating this from point A to P gives

$$h_P - h_A - \frac{1}{\rho}\left(p_p - p_A\right) - \frac{f}{2D}U^3 \triangle t + \frac{1}{\rho}\frac{4U_{W,tot}}{D}\left(T - T_{env}\right)\triangle t = 0 \qquad (7.22)$$

In which the enthalpy h, at point P is found directly. This is done in an iterative process where f, ρ and $U_{W,tot}$ is used as the average value between the points. This however results in a loss of the Joule Thomson effect and the value might therefore be debatable. Temperature at the outlet however increases at steady-state compared to initial values, but for the transient case results seem to be more accurate, since the temperature drop as a result of the Joule Thomson effect is not accounted for.

7.7.3 Solving for energy, pressure and flow rate simultaneously

Introduction of a new solver could also be done. The temperature is kept constant in between the solution of the energy equation. As the energy equation is solved for the new time-step, we have already calculated the flow rate and pressure based on the old temperature. The pressure and flow rate in the solver of the energy equation are therefore not a function of temperature, but based on the old temperature. Solving all three equations simultaneously could be done by in a new Newton-Rhapson approach, where instead the $f(\Theta)$ is a $3x1$ vector and the jacobi matrix is a $3x3$ matrix. This would also imply that we also have to calculate the derivative of the characteristic equations with respect to T_p, p_p and M_p.

One could choose to solve the energy equation at each time-step, using an interpolation for the temperature, or by the same algorithm as presented as used in the program. If this would improve accuracy is unknown, but there is a good chance it would improve stability of the solution, and some of the assumptions used above, such as constant density in the integral, could be omitted.

Chapter 8

Conclusions

The method of characteristics gives a fairly good and accurate solution to the problem of gas transport. The simplified version such as the one presented by Streeter and Wylie [4], and written for the project, gives surprisingly good results and the solutions is not very different from when the temperature equation is solved and implemented. This proves that the simplifications made by neglecting certain terms such as head velocity, and convective terms, proves sufficiently accurate when the transients are relative small. However, in large transients, such as quick shutdown of the pipeline, the simplified method used in this paper fails. Terms that have been neglected obviously play a more important role and should have been accounted for. I.e. by Yows inertial multiplier [28] as previously mentioned.

Convective terms that were neglected during the derivation of the equations become important as the velocity gradient, or corresponding flow rate gradient becomes significant. In extreme cases the mass flow rate changes with more than $200^{kg}/_{s}$ in less than a minute. The calculated pressure at the inlet show the tendency of being too low in the period right after the closure of a valve. However a clear answer of the errors, and in particular the oscillations of the error, cannot be given since the values cannot get accurately synchronized due to the lack of compatibility in the time and date stamp readings of the excel sheet of data.

As for the friction factor, the equation of Colebrook and White was used together with a correction factor. In the early calculations this corrections factor was neglected, resulting in a too high frictional pressure drop. This correction factor is meant to correct for additional effects of friction, such as drag forces due to high flow rate and pressure. This correction factor was adapted to the steady state conditions, and based on a best-fit approach to the corresponding measured pressure at the inlet. The factor was thereby held constant through the calculation. In one of the cases this constant correction factor gave very good accuracy for a relative steady flow rate at the inlet, but as the valve closed at the inlet, decreasing the flow rate to about 15% of the original flow, and a corresponding pressure drop, the total pressure gradient due to friction was calculated to low. In other cases the result were not that clear, and with larger oscillations in the calculated pressure versus measured values. However one can assume that for a more accurate solution the correction factor must be a function of both flow rate and pressure.

When it comes to the temperature calculations, we see that we get a clear temperature fall at the outlet as the valve closes. This is a result of the Joule-Thomson effect, however the same extreme pressure drop is not shown in the measurements. The temperature drop at steady state due to the Joule Thomson effect is approximately 1.5^oC, which is the result of frictional loss of pressure. But as the valve closes the pressure drops correspondingly, and the pressure gradient does in some cases become significant. However as the measurements shows sign of such a drop of temperature it comes no where near as significant as it does for the computational results. However, it seems as when the flow and pressure stabilizes the temperature goes towards normal values and close to the measured ones.

Another uncertainty regarding the temperature calculation is the whereabouts of the pipeline. No information on the situation of the pipeline in it's slope is given. and situations are only guessed. And for the pipeline specifically is given using the values of [15]. The temperature of the surrounding material is also kept constant, and independent of season and weather it consists of soil, sea-water or air. We clearly see a result of this as the situation of start-up of flow in the pipeline is simulated. In the code the surrounding temperature is kept at 5^oC, both for air, sea-water and soil, which is not correct since the date of the case III, 19.10.2007, reflects that this simulation was done in September, when the temperature of air can be expected to be a bit higher, and at least temperature of seawater which probably is a lot closer to 20^oC, than 5^oC.

The method of characteristics as a computational method for gas transport

The method of characteristics is a fairly simple method to solve partial differential equations, more specific hyperbolic equations. The method forms a system of equations that can be solved explicit by use of a linearization method. As seen from the results above, the method implemented and used in this paper, provides rather accurate results for the problem of natural gas transport, in particular for relative steady cases. However the method seem to struggle when calculating the transients. This is of course a result of some of the simplifications done when deriving the equations. Another major problem of the explicit method of characteristics is the restrictions of the time step due to the CFL condition. This gives a very strict reservation on the length of the time steps, depending on the resolution in space. If a fine resolution in space is desirable, one are also left with having to take very small time steps. The test using 2x and 4x the nodes originally used gave no clear improvement of error, but clear reductions of maximum and minimum value of the errors were seen. However, this again is very costly, and results in massive increase in computational time, since both number of equations and number of time steps are increased. The

CFL condition is also dominant in a finite volume or finite difference solver when done explicit. Implicit methods however have no restrictions in time steps, other than accuracy of the solution.

As said before the method of characteristics can also be done implicitly. Among the first to solve this was Edenhofer and Schmitz [9]. The solution of a natural gas transport problem for a medium range pipeline however requires solution of a non-symmetric stiffness matrix. This requires complex numerical procedures, and was not done for this paper. Yow[28] introduced an inertial multiplier as a remedy for the strict restrictions on the time step of the explicit method of characteristics, providing a method enabling the user to take longer time steps and accounting for the inertial term. The method was tested and implemented in the old project[13] but the implementation proved unstable for most cases, and the method was dismissed.

In general the method of characteristics is a reasonable method used to solve the problem of natural gas transport, however the solution of the energy equation in addition to the momentum and conservation equations does not improve the solution significantly. The use of finite difference methods or formulation of a finite element solver will provide simpler and probably more accurate methods to solve the gas transport problem. These methods can also have the benefit of an easier numerical manipulation, making them more versatile. It also gives an opportunity to freely choose the length of the time steps.

Future improvements and recommendations

If a more accurate solution and description of the problem, the first thing that can be done is to find the specific details on the whereabouts of the pipeline. Meaning that the burial and coverage of the pipeline should be investigated in order to obtain more accurate solutions of the temperature. Now pipeline is sort of placed in order to give the best fit for steady state conditions. A more accurate solution of the temperature equation will probably give more accurate solutions of temperature in transients. Seasonal dependence on ambient temperature should also be considered.

If the method of characteristics is to be utilized further, a new solver of the momentum and conservation equation should be implemented using more accurate equations, not neglecting such as convective term. For smaller transients the method of this paper is fine and very accurate, but exactly at the transient the method fails. But in general for a more complex system, where the temperature equation also is to be resolved, an implementation of a finite difference method would probably be in favor.

Implementation of the problem should also be considered done in a more efficient language than MATLAB. The initial idea here was to write some of the computational procedures using FORTRAN code and combining this with MATLAB. This should be possible by the use of MEX-files as an intermediary. This requires a compatible Fortran compiler to the current operating system and MATLAB version. A solution was sought, but not found, and MATLAB seemed unable to find the correct compiler. C/C++ method can however be used. If a program is made from scratch, an object oriented approach using C++ would probably be the preferred method.

Bibliography

[1] Conversation with tor ytrehus, 2011.

[2] Properties of seawater, web.mit.edu/seawater/, 2011.

[3] M.H Afshar and M. Rohani. Water hammer simulation by implicit method of characteristics. *Inernational Journal of Pressure Vessels and piping*, 2008.

[4] Victor Streeter Benjamin Wylie. *Fluid transients.* McGraw-Hill International Book Co, 1978.

[5] S.W. Churchill and M. Bernstein. A correlation equation for forced convection from gases and liquids to a circular cylinder in cross flow. *J. Heat Transfer*, pages 300–306, 1977.

[6] I. G. Currie. *Fundamental mechanics of fluid.* Marcel and Dekker, 2nd edition edition, 1993.

[7] F.W. Dittus and L. M. K. Boelter. *Univ. California.* PhD thesis, Berkeley, 1930.

[8] P. M. Dranchuk and J. H. Abou-Kassem. Calculation of z factors for natural gases using equations of state. *ournal of Canadian Petroleum Technology*, Volume 14(Number 3), Jul-Sept 1975.

[9] J. Edenhofer and G. Schmitz. Ein implizites charakteristikenverfahren zur lösung von anfangsrandwertaufgaben bie hyperbolishen systemen und seine konvergenz. *Computing 26*, 1980.

[10] Theodore L. Bergman Frank P. Incompera, Daniel P. DeWitt and Adrienne S. Lavine. *Introducion to Heat Transfer.* John Wiley and Sons, 2007.

[11] Gassco. excel sheet measured data.

[12] Aljawad Mohammed S. Ghedan, Shawket G. and Fred H. Poettmann. Compressibility of natural gases. *Journal of Petroleum Science and Engineering*, (10):157–162, 1993.

[13] Joachim Dyrstad Gjerde. Transient gas transport. Project of spring 2010, 2010.

[14] V.N. Gopal. Gas z-facort equations developed for computer. *OGJ*, pages 58–60, Aug. 8, 1977.

[15] Ola J Klock. Følsomhetsanalyse av beregningsmodell for gasstransport. Master's thesis, NTNU, 2006.

[16] Erwin Kreyszig. *Advanced Engineering Mathematics*. 1999.

[17] Leif Idar Langelandsvik. *Modeling of natural gas transport and friction factor for large-scale pipelines*. PhD thesis, NTNU, 2006.

[18] H.H Reamer L.T. Carmichael and B.H Sage. Thermal conduxtivity of fluids. methane. *Ind. Eng. Chem. Fundamentals*, 1965.

[19] Mitchell Luskin. An approximation procedure for nonsymmetric, nonlinear hyperbolic systems with integral boundary conditions. *SIAM J on numerical analysis*, 16(1):145–164, 1979.

[20] Ivan Maric. The joule-thomson effect in natural gas flow-rate measurements. *Flow Measurement and Instrumentation 16 387-395*, 2005.

[21] L. Mattar and G. S. Brar. Compressibility of natural gases. *J. Can. Pet. Technol.*, page p. 77, (1975), (Oct.-Dec.).

[22] Michael J. Moran and Howard N. Shapiro. *Fundamentals of Engineering Thermodynamics*, volume 5th edition. 2006.

[23] Olje og energidepartementet. Fakta norks petroleumsverksemd.

[24] Ross A. Purvis Peter M. Dranchuk and Donald B. Robinson. Computer calculation of natural gas compressibility factors using the standing and katz correlation. *Institute of Petroleum (London)*, 1974.

[25] M.B. Standing and D.L. Katz. Density of natural gases. *Trans AIME*, 1942.

[26] Albert S. Trube. Compressibility of natural gases. *Journal of Petroleum Technology*, Volume 9(Number 1):69–71, 1957.

[27] Frank M. White. *Viscous fluid flow*. McGraw-Hill International Book Co, 2006.

[28] W. Yow. *Analysis and Control of Transient Flow in Natural Gas Pipine Systems*. PhD thesis, Univ. of Michigan, Ann Arbour, 1971.

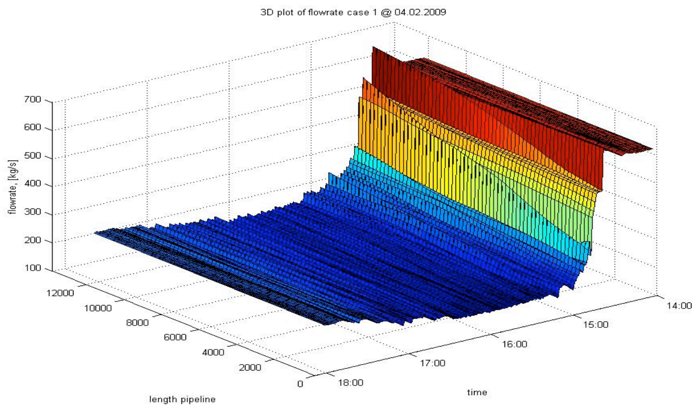
[29] Tor Ytrehus. *Energiligningen i Strømningsmekanikk*. NTNU, September 1996.

Appendix A

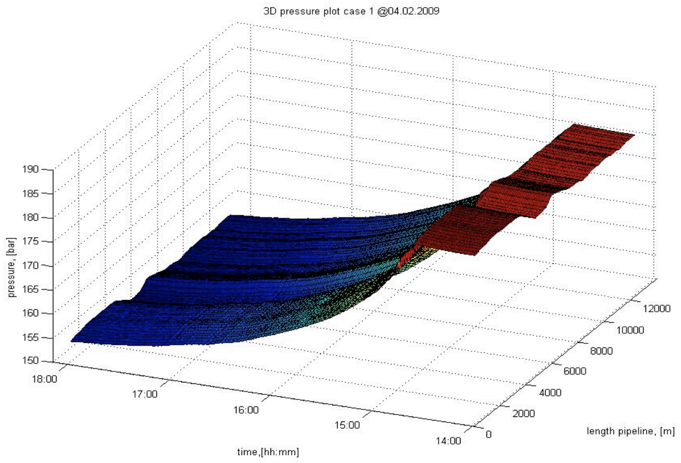
Simulations

3 dimensional plots Case I, 04.02.2009:

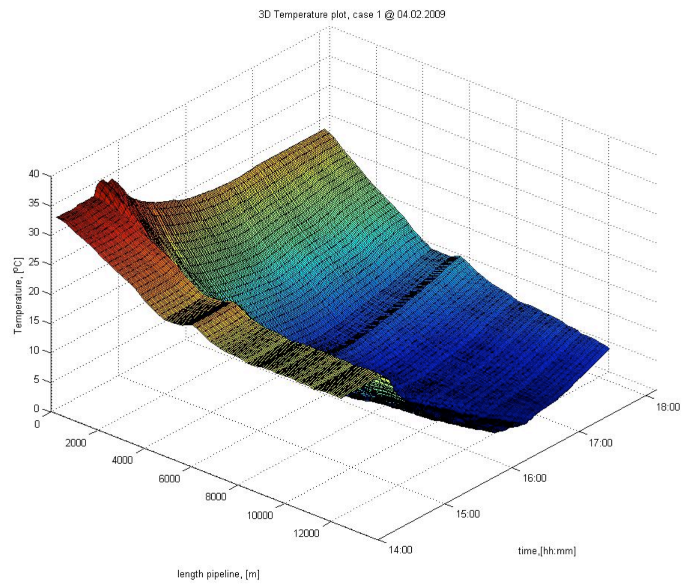
Flowrate:



Pressure:

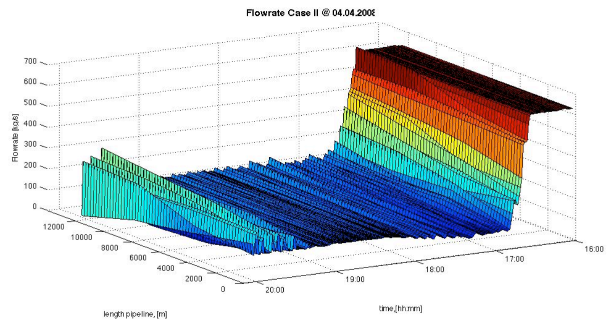


Temperature:

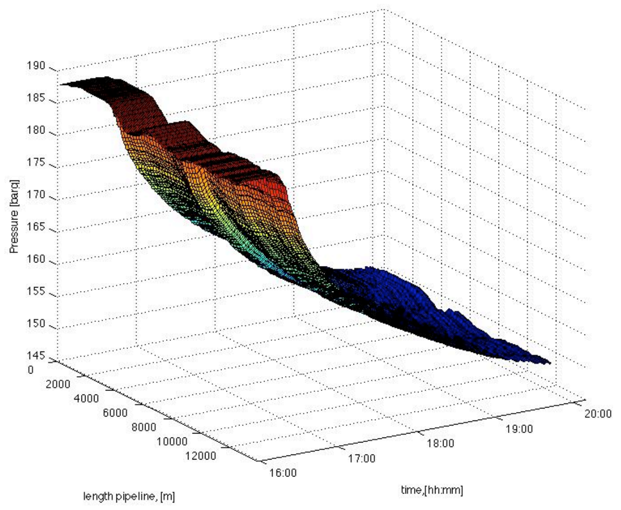


3-dimensional plots case II, 04.04.2008

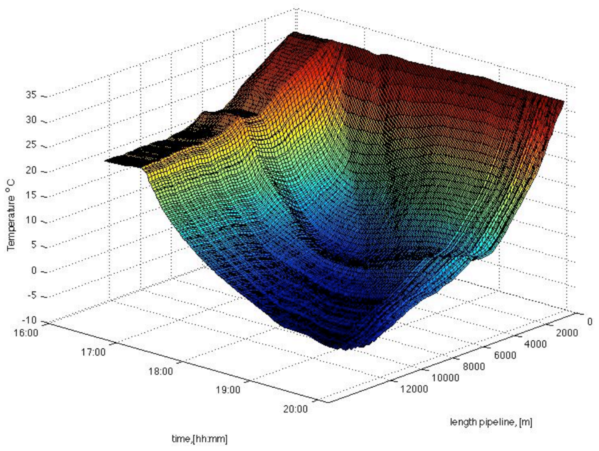
Flow rate:



Pressure:

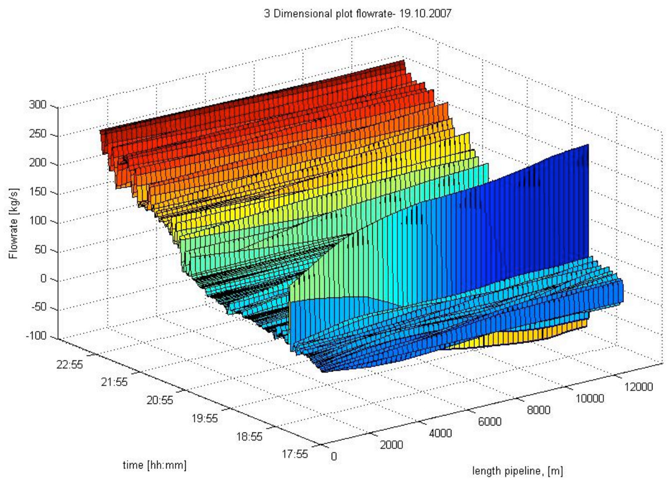


Temperature:

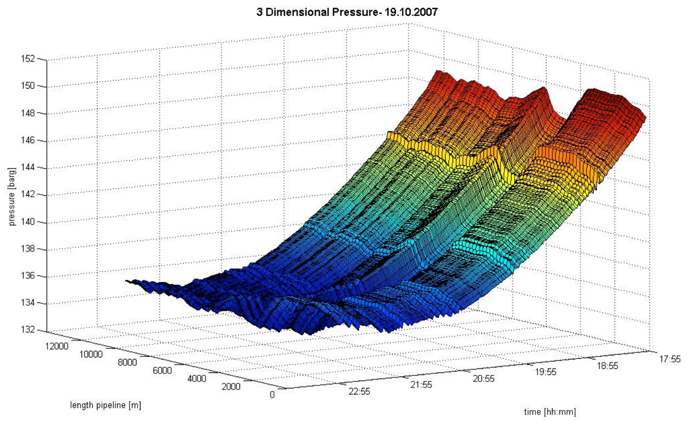


3-dimensional plots case III, 19.10.2007

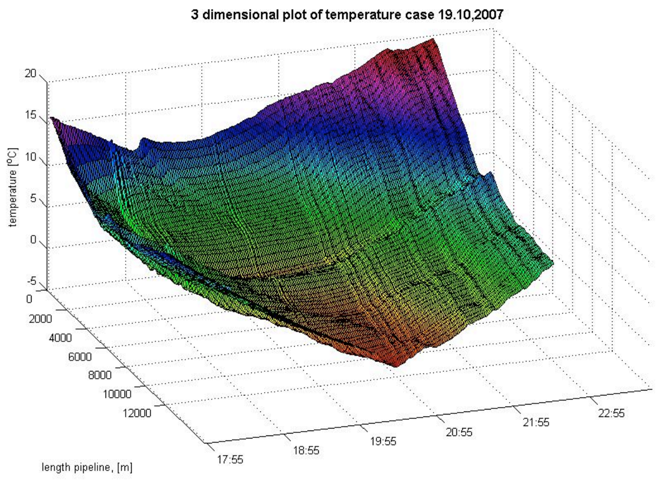
Flow rate:



Pressure:



Temperature:



Appendix B

Matlab code

B.1 main.m

```
%   ********************************
%   * MAIN METHOD FOR NATURAL GAS *
%   * PIPELINE CALCULATION        *
%   ********************************
clear all
clc
%***PRE-PROCESSING***
%-----------------------------------------------------
%   GLOBAL CONSTANTS
g=9.81;     %   R-Universal gas constant
R=8314;     %   g-gravity
atm=1.013e5;%   one atmosphere(gauge pressure)
%-----------------------------------------------------
global date;
global sim_time;
%   READING DATA
global pipelinecase;
situation=input('pipeline situation: 1-exposed, 2-coated,3-
partial buried : ');
if situation==1
    profile=xlsread('europipe2_fullyExposed.xls');  % reading
data for thin and exposed pipeline
    pipelinecase='exposed';
elseif situation==2
    profile=xlsread('europipe2_coated.xls');          % reading
data for coated pipeline
    pipelinecase='coated';
elseif situation==3
    profile=xlsread('europipe2_partiallyBuried.xls');  % reading
data for thin but buried pipeline
    pipelinecase='buried';
else
    disp('select either 1, 2 or 3')
    break
end
```

```
% reading flow data
global pReadingPrMin;
[flow_in pressure_in pressure_out h pReadingPrMin
temperature]=select_testcase();
% adapting BC's

ReqFlowLength=60*h+1;                               % required of
the flow input
ReqPressLength=60*h*pReadingPrMin+1;     % required length of
pressure input
PinRemove=numel(pressure_in)-(60*h*pReadingPrMin)-1;
PoutRemove=numel(pressure_out)-(60*h*pReadingPrMin)-1;
[pressure_in pressure_out flow_in temperature]=...

boudaryAdapt(PinRemove,PoutRemove,ReqFlowLength,ReqPressLength,..
.
    pressure_in,pressure_out,flow_in,temperature,h);


%-----------------------------------------------------
%   PIPELINE PROPERTIES
%-----------------------------------------------------
global nodeMultiplicationfactor;
nodeMultiplicationfactor=1;
global pipeline;
%---pipeline constants--------------------------------
pipeline.diameter=1.016;                    % diameter of the
pipe[m]
pipeline.area=0.25*pi*pipeline.diameter^2;  % area of the
pipe[m2]
pipeline.roughness=3e-6;                    % Pipeline
roughness[m]
pipeline.wallLayers=1;                      % number of layers in
pipeline coating

%---pipeline properties-------------------------------
pipeline.height=profile(:,3);                     % height of the
pipeline at each initial node[m]
pipeline.length=profile(length(profile(:,1)),2);  % length of
pipeline from K?rst?-Bokn[m]
pipeline.initNodes=profile(length(profile(:,1)),1); % number of
nodes from the original pipeline
% reqTotalNumberNodes= input('the total number of wanted nodes in
calculation: ');
reqTotalNumberNodes=305*nodeMultiplicationfactor+1;
if reqTotalNumberNodes<62
    reqTotalNumberNodes=62;
end
```

```
pipeline.burialType=profile(:,4);
pipeline.nodesPrDx=round(reqTotalNumberNodes/pipeline.initNodes);
pipeline.DX=pipeline.length/(pipeline.initNodes-1);
pipeline.dx=pipeline.DX/pipeline.nodesPrDx;
pipeline.nodes=pipeline.nodesPrDx*(pipeline.initNodes-1)+1;
pipeline.dXprofile=linspace(0,pipeline.length,pipeline.nodes);
fprintf('total number of pipeline nodes: %3.4g\n',
pipeline.nodes);

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

%------------------------------------------------------
%   NATURAL GAS PROPERTIES
%------------------------------------------------------
global gas;
gas.SG=0.65;                         % specific gravity of gas
gas.MW=18.01;                        % molecular weight
gas.R=R/gas.MW;                      % specific gas constant
gas.Tc=191;                          % critical temperature gas[K]
gas.pc=46.4e5;                       % critical gas pressure[Pa]

%------------------------------------------------------
%   INITIAL CONDITIONS
%------------------------------------------------------
global K; K=273.15;                  % Kelvin
initial.p_in=pressure_in(1)*atm;
initial.temperature=temperature.in(1)+K;
initial.Tr=initial.temperature/gas.Tc;
initial.pr=initial.p_in/gas.pc;
initial.Z=compressibilityfactor(initial.pr,initial.Tr);
initial.rho=initial.p_in/(initial.Z*gas.R*initial.temperature);
initial.my=viscosity(initial.temperature,initial.rho);
initial.wavespeed=sqrt(initial.Z*gas.R*initial.temperature);

%------------------------------------------------------
%   STEADY-STATE
%------------------------------------------------------
%***steady state calculations*******
MSM3_D=flow_in(1);
global flowCorr; flowCorr=8.831;
steadyFlowRate=MSM3_D*flowCorr;
steadyVelocity=steadyFlowRate/(initial.rho*pipeline.area);
pipeline.s=2*g*pipeline.dx*sin(pipeline.alfa)./initial.wavespeed^
2;

initial.pressure(1)=initial.p_in;
```

```
friction=frictionfactor(steadyVelocity,initial.rho,initial.temper
ature);
C=friction*initial.wavespeed^2*steadyFlowRate^2*pipeline.dx/(pipe
line.diameter*pipeline.area^2);

%*****case 1*****
if pipeline.nodesPrDx==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.initNodes-2
                if node>=(2+pipeline.nodesPrDx*n) &&
node<=(pipeline.nodesPrDx*(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
%-----------------------------------------------
% Initiating class FLUID
%-----------------------------------------------
global xScale;
xScale=linspace(0,pipeline.length,pipeline.nodes);
temperatureScale=linspace(0,pipeline.length,pipeline.nodes*2-1);

fluid.temperature=calcInitialTemperatureDistribution(temperature)
;
fluid.flowrate=ones(1,pipeline.nodes)*steadyFlowRate;
fluid.pressure=initial.pressure;

for iter=1:pipeline.nodes
    redPressure=initial.pressure(iter)/gas.pc;
    redTemperature=fluid.temperature(iter)/gas.Tc;

fluid.Z(iter)=compressibilityfactor(redPressure,redTemperature);

fluid.wavespeed(iter)=sqrt(fluid.Z(iter)*gas.R*fluid.temperature(
iter));

fluid.density(iter)=initial.pressure(iter)/(gas.R*fluid.Z(iter)*f
luid.temperature(iter));

fluid.velocity(iter)=fluid.flowrate(iter)/(fluid.density(iter)*pi
peline.area);
    [cp
cv]=specificHeat(fluid.temperature(iter),fluid.pressure(iter));
    internalEnergy=cv*fluid.temperature(iter);

fluid.enthalpy(iter)=internalEnergy+fluid.pressure(iter)/fluid.de
nsity(iter);
end
%oldFluid=fluid; % initiating old class to interpolate

%-----------------------------------------------------
```

```
% TIME EVALUATION
%-----------------------------------------------------
dt=pipeline.dx/max(fluid.wavespeed);
dt_old=0;
%-----time evaluation in seconds----------------------
seconds=h*3600;
timesteps=ceil(seconds*1/dt);
% time_new=0;
fluid.time=0;
time_old=0;
%-----------------------------------------------------
%   BOUNDARY CONDITION
%-----------------------------------------------------
global inletbc;
global outletbc;
inletbc.pressure=0;
outletbc.flowrate=0;
inletbc.flowrate=flow_in;
outletbc.pressure=pressure_out*atm;
outletbc.pressure(1)=initial.pressure(pipeline.nodes);
if sum(inletbc.flowrate )~=0
    inletbc.type='Flow rate  is given at inlet';
else
    inletbc.type='pressure is given at inlet';
end
if sum(outletbc.flowrate) ~=0
    outletbc.type='Flow rate  is given at outlet';
else
    outletbc.type='pressure is given at outlet';
end
%-----------------------------------------------------
%***MAIN PROCESSING***
% creating class fluid
% saveFile=input('save results?[Y/N] ','s');
saveFile='N';
status=0;
while (status==0)
    if saveFile=='Y'
        nodeSave=input('Save result for each node number: ');
        FlowFileName=input('Save flow rate result as filename:
','s');
        PressureFileName=input('Save pressure result as filename:
','s');
        inltPressureFileName=input('Save the inletpressure as
filename: ','s');
        %time_save=round(sec_save/dt);
        status=1;
    elseif saveFile=='N'
        status=1;
    else
        status=0;
        saveFile=input('save results?[Y/N] ','s');
    end
end
minSave=round(60/dt);
if saveFile=='N'
    nodeSave=round(pipeline.nodes/62);
end
row=round(pipeline.nodes/nodeSave)+1;
numColumn=ceil(timesteps/(60/dt));
flowSave=zeros(numColumn,row);
pressureSave=zeros(numColumn,row);
temperatureSave=zeros(numColumn,row);
% fig4=figure(4);
% title('dynamic pressure at the inlet, calc vs measured')
dynamicPlot=input('Dynamic plot of results(aprrox 2x  time)?
[Y/N]: ','s');
%---plot---------------------------------------------------
if dynamicPlot=='Y'
fig2=figure(2);
title('dynamic plot of inletpressure, flowrate and temperature')
subplot(2,2,1)
p1=plot(fluid.pressure(1)/atm);
xlabel('time')
ylabel('pressure[bar]')
subplot(2,2,2)
p2=plot(xScale,fluid.temperature-273.15);
xlabel('meter[m]')
ylabel('Temperature[oC]')
subplot(2,1,2)
p3=plot(xScale,fluid.flowrate);
ylabel('flow rate[kg/s]')
xlabel('meter[m]')
end
%----------------------------------------------------------

% hold on
it=2;
pIndex=1;
% pressureSaveTime=round((60/pReadingPrMin/dt));
pressureAtInlet=zeros(1,seconds/(60/pReadingPrMin));
pressureAtInlet(1)=fluid.pressure(1);
%oldTemperature=0;
%fluid.temperature=zeros(1,pipeline.nodes);

%***Variables to solve the energy equation***
% iterator=1;
% for j=2:2:pipeline.nodes*2-1
%     previousA.velocity(j-1)=fluid.velocity(iterator);
%
previousA.velocity(j)=0.5*(fluid.velocity(iterator)+fluid.velocit
y(iterator+1));
%     previousA.density(j-1)=fluid.density(iterator);
%
previousA.density(j)=0.5*(fluid.density(iterator)+fluid.density(i
terator+1));
%     previousA.pressure(j-1)=fluid.pressure(iterator);
%
previousA.pressure(j)=0.5*(fluid.pressure(iterator)+fluid.pressur
e(iterator+1));
%     previousA.Z(j-1)=fluid.Z(iterator);
%     previousA.Z(j)=0.5*(fluid.Z(iterator)+fluid.Z(iterator+1));
%     iterator=iterator+1;
% end
previousA=fluid;
% previousA.velocity(pipeline.nodes*2-
1)=fluid.velocity(pipeline.nodes);
% previousA.density(pipeline.nodes*2-
1)=fluid.density(pipeline.nodes);
% previousA.pressure(pipeline.nodes*2-
1)=fluid.pressure(pipeline.nodes);
% previousA.Z(pipeline.nodes*2-1)=fluid.Z(end);
% previousA.temperature=fluid.temperature;

previousA.time=0;
C0char_dx=0;
previousD.temperature=fluid.temperature;
previousD.density=previousA.density;
previousD.velocity=previousA.velocity;
previousD.pressure=previousA.pressure;
previousD.time=0;
%*******************************

% initiatin class  to interpolate
oldFluid.time=0;
oldFluid.pressure=fluid.pressure;
oldFluid.flowrate=fluid.flowrate;
%-------------------------------

iterator=1;
timeD=1;
mincount=0;
alfa=1;
tic
contd=0;
energyCalculations=1;
SecondIterator=1;
```

```
iterator2=1;
% break
% for iter=1:timesteps
while fluid.time<seconds

% for iter=1:timesteps
%     previousA.time=fluid.time;
    fluid.time=fluid.time+dt;
    fluid.time;
    fluid=solver(fluid,oldFluid,dt,time_old);

    tempDxAdd=fluid.velocity*dt;
    C0char_dx=C0char_dx+tempDxAdd;
    diffDx=pipeline.dx-max(C0char_dx);
%   ***Soling the energy equation***
    if iterator==1
    previousA.time=fluid.time;
    % at the first iteraton the Energy equation is not resolved
due to
    % the integration in time, and the fact that integration at
time=0
    % will lead to unreliable results

    elseif (diffDx)<tempDxAdd

    %------------------------------------------------------------
    %     solving the energy equation
    [fluid]=energyEquationSolved(fluid,previousA,temperature);
    previousA=fluid;
    %------------------------------------------------------------

    %     fprintf('remaining length left of the charactertistic:
%6.4f\n ',diffDx)
    %------------------------------------------------------------
    % Energy 2 function
    % [fluid previousA
testdata]=energy2(fluid,previousA,previousD);
    %------------------------------------------------------------

    %------------------------------------------------------------
    % Enthalpy equation
%     [fluid
previousA]=enthalpySolver(fluid,previousA,temperature);
    %------------------------------------------------------------
    %------------------------------------------------------------
    % Complete energy equation
    % [fluid
previousA]=newEnergyEqSolved(fluid,previousA,temperature);
    % break
    % [fluid
previousA]=newEnergyEqSolved2(fluid,previousA,temperature);
    %-------------------------------------------------------------


            C0char_dx=0;
            iterator2=iterator2+1;
    end

    if iterator==100
        time_evaluator=toc;
        completed=fluid.time/seconds;
        remaining_time=1/completed*time_evaluator;
        remaining_min=round(remaining_time/60);
        fprintf('Approximately time of calculation[min]:
%g\n',remaining_min);
    end

        if iterator==1
             for iter=nodeSave:nodeSave:pipeline.nodes
                flowSave(1,1)=fluid.flowrate(1);
                flowSave(1,iter/nodeSave+1)=fluid.flowrate(iter);
                flowSave(1,row)=fluid.flowrate(pipeline.nodes);
                pressureSave(1,1)=fluid.pressure(1);

pressureSave(1,iter/nodeSave+1)=fluid.pressure(iter);

pressureSave(1,row)=fluid.pressure(pipeline.nodes);
                temperatureSave(1,1)=fluid.temperature(1);

temperatureSave(1,iter/nodeSave+1)=fluid.temperature(iter);
                temperatureSave(1,end)=fluid.temperature(end);
             end
        end


        timeToFullMin=60-mod(fluid.time,60);
        if timeToFullMin<dt

            for iter=nodeSave:nodeSave:pipeline.nodes
                flowSave(it,1)=fluid.flowrate(1);

flowSave(it,iter/nodeSave+1)=fluid.flowrate(iter);
                flowSave(it,row)=fluid.flowrate(pipeline.nodes);
                pressureSave(it,1)=fluid.pressure(1);

pressureSave(it,iter/nodeSave+1)=fluid.pressure(iter);

pressureSave(it,row)=fluid.pressure(pipeline.nodes);
```

```
                temperatureSave(it,1)=fluid.temperature(1);

temperatureSave(it,iter/nodeSave+1)=fluid.temperature(iter);
                temperatureSave(it,end)=fluid.temperature(end);
            end
            it=it+1;
             minSim=round(fluid.time/60);
             fprintf('time in minutes: %3.4g\n', minSim)
             if minSim==sim_time*60;
                 disp('simulation is complete')
                 break
             end
        end

       SaveInletPressure=60/pReadingPrMin-
mod(fluid.time,60/pReadingPrMin);
       if SaveInletPressure<dt

           pIndex=pIndex+1;
           pressureAtInlet(pIndex)=fluid.pressure(1);
       end
       if dynamicPlot=='Y'
        if iterator>1

                fig2;
                subplot(2,2,1)

p1=plot(linspace(1,pIndex,pIndex),pressureAtInlet(1:pIndex)/atm,'
.-',linspace(1,pIndex,pIndex),pressure_in(1:pIndex),'r');
                xlabel('time')
                ylabel('pressure[barg]')

                subplot(2,2,2)
                p2=plot(xScale,fluid.temperature-273.15);
                xlabel('meter[m]')
                ylabel('Temperature[oC]')
                subplot(2,1,2)
                p3=plot(xScale,fluid.flowrate);
                ylabel('flowrate[kg/s]')
                xlabel('meter[m]')
        end
       end

      %----------------------------------------------
      % evaluate the timestep for the next iteration
      dt=pipeline.dx/max(fluid.wavespeed);
      if iterator~=1
        time_old=oldFluid.time;
      end
```

```
        oldFluid.time=fluid.time;
        oldFluid.pressure=fluid.pressure;
        oldFluid.flowrate=fluid.flowrate;

      if dynamicPlot=='Y'
            pause(0.01);
            delete(p1)
            delete(p2)
            delete(p3)
      end
        iterator=iterator+1;

end


%
save_flow=strcat('flow_',strcat(pipelinecase,'_',num2str(pipeline
.nodes),'_nodes','.dat'));
%
save_pressure=strcat('pressure_',strcat(pipelinecase,'_',num2str(
pipeline.nodes),'_nodes','.dat'));
%
save_temperature=strcat('temperature_',strcat(pipelinecase,'_',nu
m2str(pipeline.nodes),'_nodes','.dat'));

% save flow_040022009_isotherm.dat flowSave -ASCII
% save save_040022009_isotherm.dat pressureSave -ASCII
% save save_040022009_isotherm.dat temperatureSave -ASCII


if strcmp(date,'04.02.2009')==1
    filename=strcat('04022009_',pipelinecase,'.xls');
elseif strcmp(date,'04.04.2008')==1
    filename=strcat('04042008_',pipelinecase,'.xls');
elseif strcmp(date,'24.01.2008')==1
    filename=strcat('24012008_',pipelinecase,'.xls');
elseif strcmp(date,'29.12.2007')==1
    filename=strcat('29122007_',pipelinecase,'.xls');
elseif strcmp(date,'19.10.2007')==1
    filename=strcat('19102007_',pipelinecase,'.xls');
end


xlswrite(filename,flowSave,'Flow');
xlswrite(filename,pressureSave,'Pressure');
xlswrite(filename,temperatureSave,'Temperature');
```

```
figure(11)
surf(flowSave)
figure(12)
surf(pressureSave/atm)
figure(13)
surf(tempeartureSave-K)

%     mass_size=size(flowSave);
%     press_size=size(pressureSave);
% if saveFile=='Y'
%     save FlowFileName flowSave -ASCII
%     save PressureFileName pressureSave -ASCII
%     save inletPressureFileName pressureAtInlet -ASCII
% end
% save FlowPressureJTonly.dat flowSave -ASCII
% xlabel('length pipeline, [m]')
% set(gca,'XTick',0:10:61)
% set(gca,'XTickLabel',(0:2000:12000))
% ylabel('time,[hh:mm]')
% set(gca,'YTick',1:60:241)
% set(gca,'YTickLabel',['14:00';'15:00';'16:00';'17:00';'18:00'])
%
% zlabel('pressure, [bar]')
%  xlabel('length pipeline, [m]')
% set(gca,'XTick',0:10:61)
% set(gca,'XTickLabel',(0:2000:12000))
% ylabel('time,[hh:mm]')
% set(gca,'YTick',1:60:241)
% set(gca,'YTickLabel',['16:00';'17:00';'18:00';'19:00';'20:00'])
%  set(gca,'XTick',1:60:241)
% set(gca,'XTickLabel',['16:00';'17:00';'18:00';'19:00';'20:00'])
```

B.2 select_testcase.m

```
function [flow_rate_in pressure_in pressure_out sim_time
pressure_reading temperature]=select_testcase()
global date;
% global sim_time;
disp('following dates can be selected \n')
disp('04.02.2009 max simulation time: 4 hours')
disp('04.04.2008 max simulation time: 4 hours')
disp('24.01.2008 max simulation time: 4 hours')
disp('29.12.2007 max simulation time: 3 hours')
disp('19.10.2007 max simulation time: 5 hours')
date=input('insert the date of simulation[DD.MM.YYYY]: ','s');
% date='04.04.2008';
% global sim_time;
sim_time=input('simulation time[h]: ');
% sim_time=1;
% start_time=input('insert start time of simulation: ', 's');
% end_time=input('insert the end time of the simulation:
','s');
temperatureTest1=[0;0];
temperatureTest2=[0;0];
temperature.constAtEnd=0;
testvar=0;
testfile=0;
while (testvar==0)
if  strcmp('04.02.2009',date)==1
%     simCase=input('normal or special case?[1 or 2] ');
    simCase=1;
    if simCase==1
        pressure_reading=4;
        testvar=1;
        flow_data=[669 729 789 849 909 969];
        flow_read=[flow_data(1) flow_data(sim_time+1)];
        p_in_data=[2644 2884 3122 3366 3606 3849];
        p_in_read=[p_in_data(1) p_in_data(sim_time+1)];
        p_out_data=[2569 2805 3043 3287 3522 3758];
        p_out_read=[p_out_data(1) p_out_data(sim_time+1)];
        temperature_in_data=[506 670 842 950 0 0];
        if temperature_in_data(sim_time+1)==0
            temperature_in_data(sim_time+1)=950;
            temperatureTest1=[1;3];
        end
        t_in_read=[temperature_in_data(1)
temperature_in_data(sim_time+1)];
```

```
temperature_out_data=[1081 1184 1254 1306 1367 1436];
        t_out_read=[temperature_out_data(1)
temperature_out_data(sim_time+1)];
    elseif simCase==2
        pressure_reading=4;
        testvar=1;
        flow_data=[699 729];
        flow_read=flow_data;
        p_in_data=[2764 2884];
        p_in_read=p_in_data;
        p_out_data=[2685 2805];
        p_out_read=p_out_data;

    else
        return
    end
elseif strcmp('04.04.2008',date)==1
    pressure_reading=5;
    testvar=1;
    flow_data=[403 455 510 565 620];
    flow_read=[flow_data(1) flow_data(sim_time+1)];
    p_in_data=[2369 2692 3019 3347 3669];
    p_in_read=[p_in_data(1) p_in_data(sim_time+1)];
    p_out_data=[2150 2448 2752 3049 3347];
    p_out_read=[p_out_data(1) p_out_data(sim_time+1)];
    temperature_in_data=[466 573 738 899 0];
    if temperature_in_data(sim_time+1)==0
        temperature_in_data(sim_time+1)=899;
        temperatureTest1=[1;3];
    end
    t_in_read=[temperature_in_data(1)
temperature_in_data(sim_time+1)];
    temperature_out_data=[680 797 877 935 0];
    if temperature_out_data(sim_time+1)==0
        temperature_out_data(sim_time+1)=935;
        temperatureTest2=[1;3];
    end
    t_out_read=[temperature_out_data(1)
temperature_out_data(sim_time+1)];
elseif strcmp('24.01.2008',date)==1
     pressure_reading=5;
    testvar=1;
    flow_data=[430 484 540 596 652];
    flow_read=[flow_data(1) flow_data(sim_time+1)];
    p_in_data=[2480 2796 3130 3465 3792];
    p_in_read=[p_in_data(1) p_in_data(sim_time+1)];
    p_out_data=[2268 2560 2866 3174 3477];
    p_out_read=[p_out_data(1) p_out_data(sim_time+1)];
     temperature_in_data=[608 686 924 1110 1315];
     t_in_read=[temperature_in_data(1)
temperature_in_data(sim_time+1)];
        temperature_out_data=[1037 1170 1316 1411 1556];
        t_out_read=[temperature_out_data(1)
temperature_out_data(sim_time+1)];
elseif strcmp('29.12.2007',date)==1
     pressure_reading=5;
    testvar=1;
    flow_data=[463 523 582 642];
    flow_read=[flow_data(1) flow_data(sim_time+1)];
    p_in_data=[2669 3024 3380 3735];
    p_in_read=[p_in_data(1) p_in_data(sim_time+1)];
    p_out_data=[2422 2754 3083 3407];
    p_out_read=[p_out_data(1) p_out_data(sim_time+1)];
     temperature_in_data=[880 1024 1203 1378];
         t_in_read=[temperature_in_data(1)
temperature_in_data(sim_time+1)];
        temperature_out_data=[1045 1171 1279 1376];
        t_out_read=[temperature_out_data(1)
temperature_out_data(sim_time+1)];
elseif strcmp('19.10.2007',date)==1
     pressure_reading=5;
    testvar=1;
    flow_data=[432 492 552 611 671 731];
    flow_read=[flow_data(1) flow_data(sim_time+1)];
    p_in_data=[2937 3295 3653 4009 4364 4718];
    p_in_read=[p_in_data(1) p_in_data(sim_time+1)];
    p_out_data=[2698 3033 3363 3689 4015 4342];
    p_out_read=[p_out_data(1) p_out_data(sim_time+1)];
     temperature_in_data=[1530 1773 1883 2000 2163 2377];
         t_in_read=[temperature_in_data(1)
temperature_in_data(sim_time+1)];
        temperature_out_data=[1031 1133 1238 1344 1421 1563];
        t_out_read=[temperature_out_data(1)
temperature_out_data(sim_time+1)];
elseif strcmp('test1440to1450',date)==1
    testvar=1;
    testfile=1;
    pressure_reading=4;
else
    date=input('the date you entered was not a test case, please
try again[DD.MM.YYYY]: ','s');
end
end
if testfile==1
    flow_rate_in=xlsread('test1440to1450.xls','sheet1','A1:A11');
    pressure_in=xlsread('test1440to1450.xls','sheet1','B1:B41');
    pressure_out=xlsread('test1440to1450.xls','sheet1','C1:C41');
else
```

```
flow_column=strcat(strcat('C',num2str(flow_read(1))),':',strcat('
C',num2str(flow_read(2))));

p_in_column=strcat(strcat('E',num2str(p_in_read(1))),':',strcat('
E',num2str(p_in_read(2))));

p_out_column=strcat(strcat('I',num2str(p_out_read(1))),':',strcat
('I',num2str(p_out_read(2))));

t_in_column=strcat(strcat('G',num2str(t_in_read(1))),':',strcat('
G',num2str(t_in_read(2))));

t_out_column=strcat(strcat('K',num2str(t_out_read(1))),':',strcat
('K',num2str(t_out_read(2))));

    flow_rate_in=xlsread('data.xls',date,flow_column);
    pressure_in=xlsread('data.xls',date,p_in_column);
    pressure_out=xlsread('data.xls',date,p_out_column);
     temperature.in=xlsread('data.xls',date,t_in_column);
     temperature.out=xlsread('data.xls',date,t_out_column);
end
if temperatureTest1(1)==1

temperature.readingPrMin=ceil(numel(temperature.in)/(temperatureT
est1(2)*60))-1;
%     addTempElem=((sim_time)-
temperatureTest1(2))*temperature.readingPrMin*60;
    temperature.constAtEnd=temperatureTest1(2);
else

temperature.readingPrMin=ceil(numel(temperature.in)/(sim_time*60)
)-1;
    temperature.constAtEnd=0;
end

end
```

B.3 calcInitialTemperatureDistribution.m

```
function [T]=calcInitialTemperatureDistribution(temperature)
global xScale;
global pipeline;
global K;
global date;
global pipelinecase;
global nodeMultiplicationfactor;
% calculate a temperaturefunction of the form T(x)=Ae^bx
% A=temperature.in(1);
%
b=(1/xScale(end))*log((temperature.out(1))/(temperature.in(1))
);
%
% T=A*exp(b*xScale);
%
% T=T+K;
    if strcmp(pipelinecase,'coated')==1
        T=xlsread('initialTemperature_coated_case1.xls');
    elseif strcmp(pipelinecase,'buried')==1
        T=xlsread('initialTemperature_buried_case1.xls');
    elseif strcmp(pipelinecase,'exposed')==1
        T=xlsread('initialTemperature_exposed_case1.xls');
    else
        disp('no valid initial temperature distribution');
    end
% Tnew=ones(pipeline.nodes,1)*T(1);
% if strcmp(date,'04.02.2009')==1
    % use original temperaure
if strcmp(date,'04.02.2009')==0
%     newTemperatureVector=zeros(pipeline.nodes,1);
    inletDiff=temperature.in(1)+K-T(1);
    outletDiff=temperature.out(1)+K-T(end);

temperatureDifference=linspace(inletDiff,outletDiff,pipeline.n
odes)';
    newTemperatureVector=T+temperatureDifference;
    T=newTemperatureVector;
end

% T=Tnew;

% nodeMultiplicationfactor=2;
if nodeMultiplicationfactor~=1
    tmpT=zeros((numel(T)-1)*nodeMultiplicationfactor+1,1);
    % itr=1;
    % factor=nodeMultiplicationfactor/2;
    % delta=1/nodeMultiplicationfactor;
    % for numIter=1:factor
    %     for ii=1:nodeMultiplicationfactor:numel(tmpT)
    %         if ii==numel(tmpT)
    %             tmpT(ii)=T(end);
    %         else
    %             tmpT(ii)=T(itr);
    %
tmpT(ii+nodeMultiplicationfactor/factor)=0.5*(T(itr)+T(itr+1));
    %             itr=itr+1;
    %         end
    %     end
    % end

    delta=1/nodeMultiplicationfactor;

    tmpT(1)=T(1);
    itr1=1;
    for ii=1:numel(T)-1
    % for ii=45:46
        temperatureList=[T(ii) T(ii+1)];
        itr1=itr1+1;
        for jj=1:nodeMultiplicationfactor-1
    %         interp1([1 2],temperatureList,1+delta*jj)
            tmpT(itr1)=interp1([1 2],temperatureList,1+delta*jj);
            itr1=itr1+1;
        end
        tmpT(itr1)=T(ii+1);
    end
    tmpT(end)=T(end);
    % plot(tmpT)


    T=tmpT;
end

% T=xlsread('initialTemperature.xlsx',A1:A306);
end
```

B.4 frictionfactor.m

```
%   ******************************
%   * FRICTION FACTOR            *
%   ******************************
%***Colebrook***
function f=frictionfactor(U,rho,Temperature)
global date;
%global gas
global pipeline;
if strcmp(date,'04.02.2009')==1
    EFF=1.015;
elseif strcmp(date,'04.04.2008')==1
    EFF=1.010;
else
    EFF=1.000;
end
my=viscosity(Temperature,rho);
Re=rho*U*pipeline.diameter/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)*EFF;
            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);
%----------------------
```

B.5 solver.m

```
%   **********************************
%   *The solver used to solve        *
%   *the transient problem using the *
%   *method of characteristics       *
%   **********************************
function fluid=solver(fluid,oldFluid,dt,time_old)
%-------------------------------------------------
%   set global variables
global pReadingPrMin;
global flowCorr
%global pipeline.nodesPrDx
global inletbc
global outletbc
% global dt
global gas
global pipeline
readDivide=60/pReadingPrMin;
tolerance=1e-6;
maxiter=100;
newvalue=[zeros(1,pipeline.nodes);zeros(1,pipeline.nodes)];
nodeCount=0;
slopeCount=1;
for node=1:pipeline.nodes
    %---inlet boundary---
    if node==1

%avgVelocity=0.5*(fluid.veloscity(1)+fluid.velocity(2));
        %avgDensity=0.5*(fluid.density(1)+fluid.density(2));
%         pr=fluid.pressure(1)/gas.pc;
%         Tr=fluid.temperature(1)/gas.Tc;
%         fluid.compFactor(1)=compressibilityfactor(pr,Tr);
%
speedOfSound=sqrt(fluid.compFactor(1)*gas.R*fluid.temperature(
1));
%
C=fr*gas.wavespeed^2*pipeline.dx/pipeline.diameter/pipeline.ar
ea^2;
```

```
fr=frictionfactor(fluid.velocity(1),fluid.density(1),fluid.temper
ature(1));

C=fr*fluid.wavespeed(1)^2*pipeline.dx/(pipeline.diameter*pipeline
.area^2);
        if pipeline.s(1)==0
            s_tempval=1;
        else
            s_tempval=(exp(pipeline.s(1))-1)/pipeline.s(1);
        end

        switch inletbc.type
            case 'pressure is given at inlet'
                newvalue(2,1)=inletbc.pressure;
                M_1=fluid.flowrate(1);
                M_2=fluid.flowrate(2);

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

                err=inf;
                iter1=0;
                while err>tolerance
                    newvalue(1,1)=M_1-
((fluid.wavespeed(1)/pipeline.area)*(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(pipeline.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
                end



            case 'Flow rate  is given at inlet'


newvalue(1,1)=flowCorr*interp1(linspace(1,length(inletbc.flowrate
),...
```

B.6 compressibilityfactor.m

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

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

function Z=compressibilityfactor(pr,Tr)
% Z=1;

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
elseif pr>5.4 && pr<15
    a=0.711;
    b=3.66;
    c=-1.447;
    d=-1.637;
    e=0.319;
    f=0.522;
    g=2.071;
else return
end
if pr<=5.4
   Z=pr*(A*Tr+B)+C*Tr+D;
elseif pr>5.4
    Z=pr*(a+b*Tr)^c+d/(e*Tr+f)+g;
end


%    Z=1;
end
```

B.7 overallHeatTranfer.m

```
% Calculation of the overall heat transfer coefficient

function
U_Wtot=overallHeatTransfer(avgVelocity,burial,rho,my_gas,cp_ga
s)
global pipeline;
global pipelinecase;
% global gas;
%global flowCorr;
diameter.inner=1.00;        % pipeline inner diameter
if strcmp(pipelinecase,'coated')==1
    wallLayers=3;               % number of layers in coated
pipeline
    diameter.outer=1.160;       % pipeline outer diameter
    radius=[0.500;0.520;0.530;0.58];
    k_wall=[50;0.74;2.9];
    h_wall=0;
else
    wallLayers=1;
    diameter.outer=1.002;
    k_wall_single=100;
end
% diameter.outer=1.002;
%abientTemperature=5+273.15; % simplified ambient temperature
5 degrees Celcius

shallow_Dc=diameter.outer/2+0.001;                    % depth
to centreline[m]
% shallow_Dc=0.75;
% controlDepth=(2*shallow_Dc/diameter.outer);
% if controlDepth<=1
%     shallow_Dc=1*diameter.outer/2;
% end
deep_Dc=1.0;
k_soil=2;                   % soil conductivity[W/mK]
if burial==1
%if strcmp('shallow',burialType)==1;
    % shallow burial of pipeline

h_outer=(2*k_soil/diameter.outer)/log((2*shallow_Dc/diameter.o
uter)...
        +sqrt((2*shallow_Dc/diameter.outer)^2-1));  % shallow
burial coefficient
```

```
elseif burial==2
% elseif strcmp('deep',burialType)==1
    % deep burial of pipeline

h_outer=(2*k_soil/diameter.outer)/log(4*deep_Dc/diameter.outer);
% deep burial coefficient
elseif burial==3
%elseif strcmp('water',burialType)==1
    % if exposed to water
    U_sea=0.1;              % average sea velocity
    cp_sea=3985;            % specific heat[W*kg^-1*K^-1]
    k_sea=0.563;            % thermal conductivit seawater[W*m^-
1*K^-1]
%     k_sea=0.5;
    rho_sea=1030;           % sea water density[kg/m^3]
    my_sea=1.881e-3;        % sea water viscosit[N/m^2*s]

    Pr_sea=cp_sea*my_sea/k_sea;
    Re_sea=rho_sea*U_sea*diameter.outer/my_sea;
    Nu_sea=0.26*Re_sea^0.6*Pr_sea^0.3;
    h_outer=k_sea*Nu_sea/diameter.outer;
% water burial coefficient

    % nusselt forced convection Klock thesis
%
Nu_sea_forced=0.3+0.62*Re_sea^0.5*Pr_sea^0.333/(1+(0.4/Pr_sea)^0.
6667)^0.25*(1+(Re_sea/282000)^(5/8))^(4/5);
elseif burial==4
    %test exposure to air
    U_air=3.0;
    cp_air=1.0035;
    k_air=0.025;
    rho_air=1.250;
    my_air=16.625e-6;
    Pr_air=cp_air*my_air/k_air;
    Re_air=rho_air*U_air*diameter.outer/my_air;
    Nu_lam=0.664*Re_air^(1/2)*Pr_air^(1/3);
    Nu_turb=0.037*Re_air^0.8*Pr_air/(1+2.443*Re_air^-
0.1*(Pr_air^(2/3)-1));
    Nu_air=0.3+sqrt(Nu_lam^2+Nu_turb^2);
%
Nu_air_test=0.0266*Re_air^0.805*Pr_air^(1/3)*(k_air/diameter.oute
r);
    %     Nu_air=0.26*Re_air^0.6*Pr_air^0.3;
    h_outer=k_air*Nu_air/diameter.outer;
end
% h_outer
% heat conduction through pipe wall
if wallLayers==1
```

B.8 JouleThomsonCoefficientCalc.m

```
function [mu_JT
dmudT]=JouleThomsonCoefficientCalc(Temperature,pressure)
global gas;

% Temperature=(previousA.temperature(node-1)+Temperature)/2;
% pressure=(fluid.pressure(node)+previousA.pressure(node-1))/2;
[cp, cv]=specificHeat(Temperature,pressure);
Tr=Temperature/gas.Tc;
pr=pressure/gas.pc;
% Z=compressibilityfactor(pr,Tr);
% [Zmattar dZdpr dZdT ddZddT]=mattarZfunction(Tr,pr);
% [dZdT]=Zderivative(Tr,pr,Z);
% mu_JT=gas.R*Temperature^2/(cp*pressure)*dZdT;
% [Z, dZdTmattar]=mattarZfunction(Tr,pr);
% % dZdTmattar
% mu_JT=gas.R*Temperature^2/(cp*pressure)*dZdTmattar;

Z=DranchukAbouKassemZ(Tr,pr);
dZ=DranchukAbouKassemZderivative(Tr,pr,Z);
% dZ
mu_JT=gas.R*Temperature^2/(cp*pressure)*dZ;

dmudT=gas.R*Temperature/(cp*pressure)*dZ;
return
```

B.9 DranchukAbouKassemZ.m

```
function Z=DranchukAbouKassemZ(Tr,pr)

a=[0.3265;-1.0700;-0.5339;0.01569;...
    -0.05165;0.5475;-0.7361;0.1884;...
    0.1056;0.6134;0.7210];
J=1;
Dr=1;


Tr1=Tr;
Tr2=Tr^2;
Tr3=Tr^3;
Tr4=Tr^4;
C0=a(1)*Tr+a(2)+a(3)/Tr2+a(4)/Tr3+a(5)/Tr4;
C1=a(7)+a(8)/Tr;
C2=a(6)*Tr+C1;
C3=-C1*a(9);
C4=a(10)/Tr2;



if Tr<1
    Dr=0;
    DDr=.1;
    J=0;
end

% J=0;
%     Dr=0;
%     DDr=.1;
for iflag=1:10
% some how inside the loop
    if J==0
        Dr1=Dr;
        Dr=Dr+DDr;
        Dr2=Dr^2;
        Dr4=Dr^4;
        Dr5=Dr^5;
    elseif J>0
        Dr2=Dr^2;
        Dr4=Dr^4;
        Dr5=Dr^5;
    end
```

```
T1=C0*Dr;
    T2=C2*Dr2;
    T3=C3*Dr5;
    T4=C4*Dr2;
    T5=a(11)*Dr2;
    T6=exp(-T5);

    ZCPR=(Tr+T1+T2+T3)*Dr+T4*(1+T5)*T6*Dr;
% break
    DZCPR=Tr+2*T1+3*T2+6*T3+T4*T6*(3+3*T5-2*T5^2);

    PrCalc=ZCPR/0.27;
    if J>0

        Dr1=Dr-(ZCPR-.27*pr)/DZCPR;
        if Dr1<0
            Dr1=Dr*0.5;
        elseif Dr1>2.2
            Dr1=Dr+0.9*(2.2-Dr);
        end

        if abs(Dr-Dr1)<=1e-5
%               disp('Dr-Dr1')
            break
        end


    elseif J==0
        if abs(PrCalc-pr)<1e-3
%               disp('PrCalc-pr')
            break
        elseif PrCalc>pr
            DDr=DDr/2;
        end
    end

    Dr=Dr1;


end

Z=0.27*pr/(Dr*Tr);



% Zold=1;
% error=inf;
% iter=1;
% while error>1e-4
```

B.10 DranchukAbouKassemZderivative.m

```
function dZdT=DranchukAbouKassemZderivative(Tpr,p_pr,Z)
global gas;
a=[0.3265;-1.0700;-0.5339;0.01569;...
    -0.05165;0.5475;-0.7361;0.1844;...
    0.1056;0.6134;0.7210];

rho_r=0.27*p_pr/Z/Tpr;
iterator=1;
drho_r=1;
dZtemp=1e-3;
error=inf;
while error>1e-4
    dZdTpr=a(1)*drho_r+a(2)*(drho_r-
rho_r/Tpr)*(1/Tpr)+a(3)*(drho_r-3*rho_r/Tpr)*(1/Tpr^3)+...
        a(4)*(drho_r-4*rho_r/Tpr)*(1/Tpr^4)+a(5)*(drho_r-
5*rho_r/Tpr)*(1/Tpr^5)+...
        2*a(6)*rho_r*drho_r+a(7)*rho_r*(2*drho_r-
rho_r/Tpr)*(1/Tpr)+...
        2*a(8)*rho_r*(drho_r-rho_r/Tpr)*(1/Tpr^2)...
        -a(7)*a(9)*rho_r^4*(5*drho_r-rho_r/Tpr)*(1/Tpr)...
        -a(8)*a(9)*rho_r^4*(5*drho_r-2*rho_r/Tpr)*(1/Tpr^2)+...
        a(10)*rho_r*(2*drho_r-3*rho_r/Tpr-
2*a(11)*rho_r^2*drho_r)*exp(-a(11)*rho_r^2)/Tpr^3+...
        a(10)*a(11)*rho_r^3*(4*drho_r-3*rho_r/Tpr-
2*a(11)*rho_r^2*drho_r)*exp(-a(11)*rho_r^2)/Tpr^3;

    drho_r=-rho_r*(1/Tpr+1/Z*dZdTpr);
    error=norm(dZdTpr-dZtemp);
    dZtemp=dZdTpr;
    if iterator>100
        disp('could not find a proper correlation to dZdT');
        break
    end
end
dTprdT=1/gas.Tc;
dZdT=dZdTpr*dTprdT;
return
```

```
function dZdT=DranchukAbouKassemZderivative(Tpr,p_pr,Z)
global gas;
a=[0.3265;-1.0700;-0.5339;0.01569;...
    -0.05165;0.5475;-0.7361;0.1844;...
    0.1056;0.6134;0.7210];

rho_r=0.27*p_pr/Z/Tpr;
iterator=1;
drho_r=1;
dZtemp=1e-3;
error=inf;
while error>1e-4
    dZdTpr=a(1)*drho_r+a(2)*(drho_r-
rho_r/Tpr)*(1/Tpr)+a(3)*(drho_r-3*rho_r/Tpr)*(1/Tpr^3)+...
        a(4)*(drho_r-4*rho_r/Tpr)*(1/Tpr^4)+a(5)*(drho_r-
5*rho_r/Tpr)*(1/Tpr^5)+...
        2*a(6)*rho_r*drho_r+a(7)*rho_r*(2*drho_r-
rho_r/Tpr)*(1/Tpr)+...
        2*a(8)*rho_r*(drho_r-rho_r/Tpr)*(1/Tpr^2)...
        -a(7)*a(9)*rho_r^4*(5*drho_r-rho_r/Tpr)*(1/Tpr)...
        -a(8)*a(9)*rho_r^4*(5*drho_r-2*rho_r/Tpr)*(1/Tpr^2)+...
        a(10)*rho_r*(2*drho_r-3*rho_r/Tpr-
2*a(11)*rho_r^2*drho_r)*exp(-a(11)*rho_r^2)/Tpr^3+...
        a(10)*a(11)*rho_r^3*(4*drho_r-3*rho_r/Tpr-
2*a(11)*rho_r^2*drho_r)*exp(-a(11)*rho_r^2)/Tpr^3;

    drho_r=-rho_r*(1/Tpr+1/Z*dZdTpr);
    error=norm(dZdTpr-dZtemp);
    dZtemp=dZdTpr;
    if iterator>100
        disp('could not find a proper correlation to dZdT');
        break
    end
end
dTprdT=1/gas.Tc;
dZdT=dZdTpr*dTprdT;
return
```