Abstract

This thesis describes a new general purpose dynamic process simulator applied to a natural gas liquefaction plant. More specifically, the Multi Fluid Cascade Process (MFCP). MFCP is the “Statoil Linde LNG Technology Alliance” - a proprietary and patented process for LNG production. This utilizes plate fin and spiral wound heat exchangers, produced by Linde AG, and Nuovo Pignone’s centrifugal compressors. This LNG technology is now being implemented for the Hammerfest baseload LNG project, Snøhvit.

The simulator is based on first principle conservation laws for energy and mass, and a simplified quasi-steady state momentum equation. Unit models for the process equipment (tanks/pipes, separation tanks, valves, liquid expanders, pumps, compressors, heat exchangers, and PI controllers) are described using rigorous thermodynamics. Equilibrium is assumed for all unit models, and equilibrium and physical properties are predicted with the Swoave-Redlich-Kwong or the Peng-Robinson equation of state.

Two different model approaches are compared in this work. One approach conserves energy in an enthalpy state, and the other conserves internal energy, the *HP formulation* and *UV formulation* respectively. The HP formulation defines a dynamic state for the pressure, and splits the integration of the fast and slow dynamics. The pressure states and the algebraic flow relations are solved by a fully implicit Euler method, while the internal unit model equation is solved locally with tailormade integration routines. The UV formulation, utilizes an analytical Jacobian, and is integrated with both a 1-stage Rosenbrock and freeware BDF codes.

The UV formulation generates an analytical Jacobian from physical property partial differentials. These property differentials are calculated from a first order approximation of the equilibrium. The equilibrium equations are linearized in dynamic state variables to produce partial differentials of the internal flash variables.

The simulator is tested on a portable PC. The full MFCP LNG plant is simulated with a fixed time step of 1.0 seconds, for both the HP and UV formulations.

The HP formulation has 611 ODE states. The UV formulation uses the 1-stage Rosenbrock method integrating a system with 1025 ODE states. The case simulated is a set point step for the LNG temperature controller. The plant is simulated over 9000 seconds. The major process dynamics are sampled every second, and plotted. The average performance of both formulation is better that 7 times real time. The worst local performance of the UV formulation is 1.7 times faster than the HP formulation, but more than 6 times faster than real time. The HP and UV formulations gave significantly different dynamic predictions.

The BDF codes proved too slow for practical use on the MFCP configuration.

The dynamic simulators in industry today are typically using a HP formulation, with precalculated thermodynamic data stored in look-up tables. The simulations of the full LNG plant show that simulators, utilizing EOS equilibrium descriptions, soon will be able to compare with modern industry simulators.

Preface

This thesis is submitted in partial fulfilment of the requirements for the *doktor ingeniør* degree at the Norwegian University of Science and Technology (NTNU). The work has been carried out during the period from 2001 to 2004 at Department of Energy and Process Engineering (EPT) with Professor Geir Asle Owren as my supervisor. Professor Morten Hovd, at Department of Engineering Cybernetics, NTNU, has been an adviser during the thesis work. I wish to thank Geir Owren and Morten Hovd for inspiring and enlightening me.

The Research Council of Norway and Statoil ASA are gratefully acknowledged for financing this work.

I would also like to thank my fellow students at the department at NTNU, and my colleagues at the Statoil Research Centre in Trondheim for discussions and support.

Finally, I want to express gratitude to my girlfriend, my parents, my two sisters, my grandparents and my friends for their unconditional support and encouragement during this work.

Contents

Nomenclature

Abbreviations

PM	Set of all pressure models
WM	Set of all flow models
AD	Automatic Differentiation
AE	Algebraic Equations
BDF	Backward Differential Formulas
CD	Central Difference
CFD	Computational Fluid Dynamics
CV	Control Volume
DAE	Differential Algebraic Equations
EOS	Equations Of State
FOM	Full Orthogonalization Method
GMRES	Generalized Minimum Residual Method
GUI	Graphical User Interface
HHC	Heavy HydroCarbon
IPDAE	Integral Partial Differential Algebraic Equation
IS	Index set
LNG	Liquefied Natural Gas
LUT	Look-up table
MCR	Multicomponent Refrigerant
MCR1	Multicomponent refrigerant in pre-cooling loop

MCR2	Multicomponent refrigerant in liquefaction loop	
MCR3	Multicomponent refrigerant in sub-cooling loop	
MOL	Method Of Lines	
NC	Number of Components	
NG	Natural Gas	
NR	Newton-Raphson	
ODE	Ordinary Differential Equations	
PDAE	Partial Differential Algebraic Equation	
PDE	Partially Differential Equations	
PID	Controller with Proportional-Integral-Derivative action	
PM	Pressure model	
PR	Peng-Robinson Equations Of State	
RHS	Right Hand Side	
RK	Runge-Kutta	
RPN	Reverse Polish Notation	
RTSTR	Real Time Simulation Time Ratio	s/s
SEPTIC	Statoil Estimation and Prediction Tool for Identification and Control	
SP	Sub Process	
SRK	Swoave-Redlich-Kwong Equations Of State	
TPlib	Thermodynamic and physical data property package. Function library.	
VLE	Vapour Liquid Equilibrium	
VOL	Short for volume	m^3
WM	Flow model	

Greek letters

α	Determining coefficient of the Rosenbrock method
α	Interpolation weight
α	Line search parameter, $\langle 0, 1]$
α	Parameter in the multistep integration method

β	Parameter in the multistep integration method	
$\Delta_{a,b}$	Difference operator, ex. $\Delta_{a,b}P = P_a - P_b$	
η	Efficiency	(0-1)
Γ	Backward difference polynomial interpolation operator (Equation B.23)	
γ	Arbitary property	
γ	Determining coefficient of the Rosenbrock method	
γ	Isentropic efficiency	
κ	Compressor parameter, heat capacity ratio (constant pressure over constant volume)	
λ	Eigenvalue	
∂	Partial differential operator	
Φ	Flux into/out of a control volume	
Ψ	Arbitrary extensive quantity stored in a control volume	
ρ	Density	kmol/m^3
Σ	Source or sink to a control volume	
σ	Source or sink to a control volume (per area)	
τ	Space time	s
τ	Volume multiplied by density pressure differential	

Subscript

AP1	Flow interpolation to CV boundarys in heat exchanger
AP2	Property interpolation to CV midpoint in heat exchanger
AT	Temperature interpolation in heat exchanger
av	Average
des	Design value
I	Interpolated property
i	Index
In	Inflowing property
j	Index

k	Index	
L	Liquid	
Mass	The basis is kg (Not kmol)	
nom	Nominal value	
Out	Outflowing property	
P	Polytropical	
pi	Flow model index, inlet	
po	Flow model index, outlet	
s	Stream	
Simp	Short for simple	
spec	Short for specification	
tot	Overall property, (Superposition)	
V	Vapor	

Latin letters

$\mathcal{N}$	Set of natural numbers, {0,1,2,....}	
C_{PC}	Compressor performance curve function	
K	Controller gain	
T	Controller constant	
$\mathbf{F}$	General function vector	
$\mathbf{X}$	General variable vector	
v	Differential index	
w	Intermediat variable	
b	Determining coefficient of the Rosenbrock method	
c	General flash specification	
Cv	Valve constant	$1/\%\ m^2$
d	Differential operator	
E	Energy	J
e	Controller error	

e	General error	
H	Enthalpy	J
H	Head	J/kg or m
h	Height	m
h	Specific enthalpy	J/kmol
hv	Valve opening	%
hw	Wall enthalpy	
I	Incidence/occurrence matrix	
intrinsic	Intrinsic function or operation	
J	Jacobian (Equation B.21)	
l	Heat transferring perimeter	m
Mw	Mole weight	kg/kmol
N	Molar holdup	kmol
N	Number of dynamic control volumes is N + 1	
N	Speed	rpm
n	Integer	
P	Pressure	Bar
Q	Volume flow	m^3/h
r	Compressor parameter	
r	Controller reference signal	
r	Residual, $r = b - Ax$	
S	Entropy	J/kmol K
S	Set	
S	Source flow	kmol/s
T	Temperature	K
t	Time	s
Tw	Wall temperature	
U	Heat transfer coefficient	J/m^2 s

U	Internal energy	J
u	Controller output signal	
u	Specific internal energy	J/kmol
V	Vapor flow	kmol/s
V	Volume	m^3
v	Specific volume	m^3/ρ
v	Velocity	
W	Flow	kmol/s
w	Vapor fraction	kmol vapor/kmol total
x	Liquid composition	kmol/kmol
x	Spatial dimension	m
y	Measurement input to controller	
y	Vapor composition	kmol/kmol
Z	Compression factor	
z	Overall mole fraction	kmol/kmol

Supercripts

*	Scaled reference value
D	Derivative
I	Integral
k	Index
P	Proportional
S	Source
tp	Two-Phase

Chapter 1

Introduction

1.1 LNG - the concept

LNG is short for Liquefied Natural Gas, and is the product from fully condensing natural gas. LNG is at a state close to atmospheric pressure and at temperature of about -163^{o}C. The specific volume ratio of natural gas at atmospheric pressure to LNG is approximately 600 to 1.

The process of liquefying natural gas is demanding in terms of energy and expensive. So why do it?

Natural gas must be transported from the production location to the customer. If the production site is located far from customers, it may not be possible to transport the natural gas in pipelines. The natural gas must then be shipped with some motorized transport. When condensing the natural gas to LNG the means of transport can carry 600 times more mass in the same volume. The shipping cost of LNG is therefore much smaller than for natural gas.

Other benefits when shipping LNG by motorized transport compared to a pipeline, is that the customers can be at different locations, and there is the possibility to change custumers.

The liquefaction of natural gas is not a new technology. The first land-based LNG plant was operational in Algeria, in 1964. Since then, the NG liquefaction process has been developing, and has become increasingly more efficient.

1.2 Background

Over the last two decades what is now the Norwegian University of Science and Technology (NTNU) has produced several PhD theses on the subject of LNG.

The main focus of the research has been on pressure drop and heat transfer in coil-wound heat exchangers, and on thermodynamics. Fredheim (1994), Neeraas (1994) and Aunan (2000) did measurements on the shell and tube side of a coil-wound heat exchanger. Owren (1988), Melaaen (1993), Jørstad (1993) and Grini (1994) worked with thermodynamic properties and equations of states. Melaaen (1994) produced a thesis on the dynamic modelling of the liquefaction in a LNG plant.

The Norwegian University of Science and Technology together with SINTEF Energy Research, have worked in close cooperation with Statoil R&D on LNG research since 1985. This cooperation has produced three applications for LNG simulation. CryoPro, and SCoil, Fredheim et al. (2000), are thermal design tools for LNG heat exchangers, and static simulation tools. CryoPro simulates an entire liquefaction process. SCoil simulates the spiral wound heat exchanger. The DCoil application, Vist et al. (2003) and Hammer et al. (2003) dynamically describe the spiral-wound heat exchanger. All these applications use heat transfer and pressure drop correlations based on the measurements made at Department of Energy and Process Engineering, NTNU.

The academic work on dynamic LNG simulation is limited. The only work found and studied are Zaïm (2002) and Melaaen (1993).

The background of this thesis, and the reasons to develop a dynamic simulator can be summarized in the following six points.

- Help in understanding process dynamics

  Processes are getting more and more integrated, and it is therefore hard to foresee and understand all the dynamics of a process. Working with a model can therefore give valuable information about the process dynamics.

- Verification of existing design and guide to redesign control system

  A simulator will be a platform for the preliminary implementation of the control system. Given a control structure it is possible to perform operability studies or safety and risk analysis. These studies will give feedback on the performance of the current control system, and if redesign is a necessity, and/or if tuning of control parameters is needed.

- On-line optimization

  On-line optimization is needed for model-based control systems.

- Identification of process information

  Process variables not directly available from measurements can be estimated using a model.

- Training simulator

If the simulator is robust and simulates real time or faster, it can be used for operator training.

- Static simulations

  A dynamic simulator can be used as a static simulator, and in static optimization.

1.3 Topic of the thesis

This thesis presents results from dynamic modeling and simulation work for general LNG processes. On the superior level two different solution methods are implemented and compared using a test case. The test case is the Mixed Fluid Cascade Process (MFCP), Foerg et al. (1998); a liquefaction strategy utilizing multicomponent refrigerants, MCR, and spiral-wound heat exchangers.

During the period from January 2001 to December 2003 unit models for the process equipment have been modelled based on first principle conservation laws. Equilibrium, enthalpy and density data have been calculated using the SRK equation of state.

1.3.1 The simulator

A simulator with keyword configuration and a graphical user interface (GUI) is developed. The keyword configuration defines the unit models and their connections, and the low level controllers. The implemented unit models reflect the need, when simulating LNG processes. The GUI shows plots of simulated properties and gives the possibility to start and stop the simulation.

The simulator should be able to estimate the dynamic behavior of the configured process around the predefined steady state. Extreme situations such as startup/shutdown and damage to equipment are not considered.

The units conserve mass and energy, but the momentum conservation is treated in a quasi steady state fashion. Therefore the mass flow relations are described using simple algebraic pressure drop equations.

The controllers have proportional and integral action. That is; PI-controllers, that can be used in cascade, where one controller gives the set point to another controller.

The integration of the dynamic states is done in two different ways. With the first approach the fast flow-pressure dynamic is integrated separately from the slower energy and compositional dynamics. To do this, the pressure must be a dynamic state. This requires that the energy is conserved as enthalpy giving rise to an HP flash. The pressure-flow integration is then done in a linear implicit or in a fully implicit way, succeeded by tailormade unit integrations.

The second approach is equation oriented, and all dynamic states are integrated simultaneously. The Jacobian is written out analytically. The unit models calculate their own differentials, and the solver maps the entire Jacobian matrix. In order to produce differentials of state variables, normalized equilibrium equations are linearized.

1.3.2 The LNG plant - MFCP

Figure 1.1 shows a simplified layout of the MFCP, NG liquefaction plant. The figure drawing is based on Stockmann et al. (1998) and Foerg et al. (1998). The process units and the fluid flows are described in the following sections.

The simulator could easily be configured to simulate other NG liquefaction plants. This only requires a new configuration file.

In this process, the natural gas is entering on the top of Figure 1.1. After being heat exchanged through four heat exchangers, E1A, E1B, E2 and E3, the gas is fully condensed to LNG after being pressure relieved to about atmospheric pressure. Three multicomponent refrigerant loops are used to remove the require energy from the natural gas.

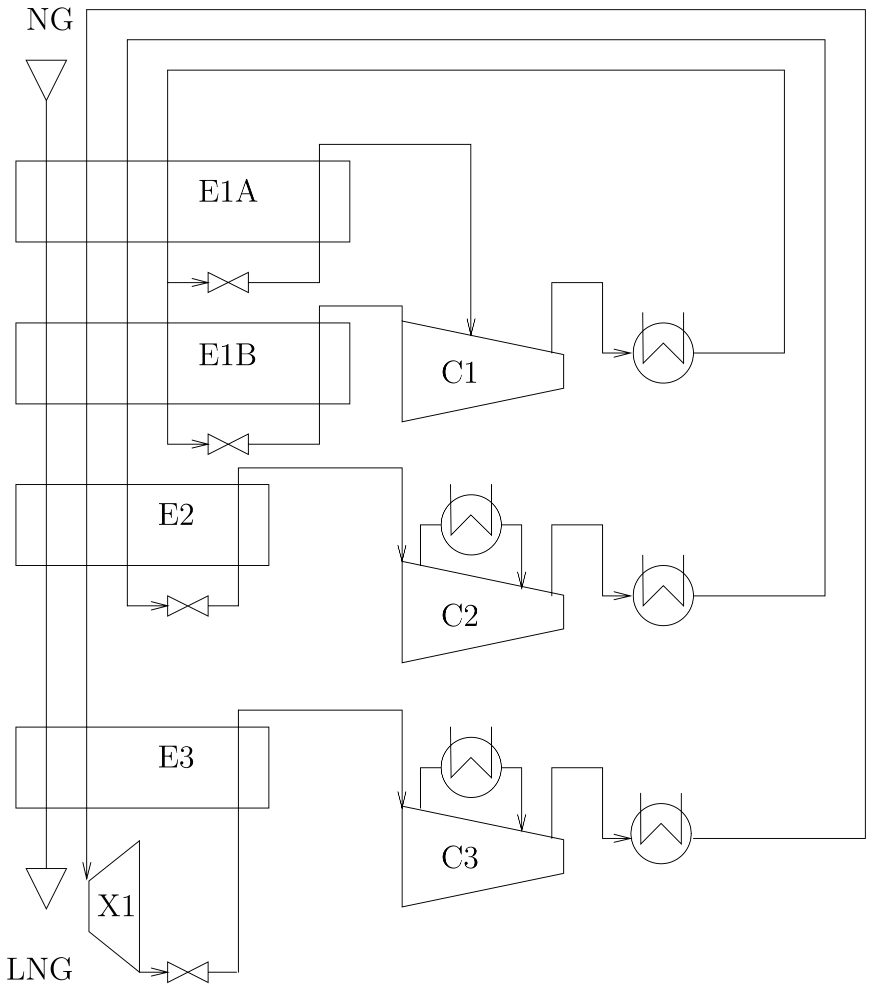


Figure 1.1: The Mixed Fluid Cascade Process (MFCP)

Pre-cooling

The pre-cooling consists of the heat exchangers E1A and E1B. The natural gas enters as overheated gas and is cooled to a state approximately at the dew point. These heat exchangers also pre-cool themselves and the refrigerants for liquefaction and sub-cooling.

The refrigerant is a mixture of ethylene, ethane, propane and butane. The refrigerant enters the heat exchanger E1A at the top of E1A, and is used at two pressure levels. Between E1A and E1B a fraction of the refrigerant is taken to a Joule-Thompson valve to lower pressure, and used as cooling agent in E1A.

The fraction, not pressure released between E1A and E1B, is lead through the E1B heat exchanger before it is depressurized and used as a cooling agent in E1B.

The refrigerant is compressed in two stages. The cooling agent of E1A first enters at the second compression stage. The refrigerant is cooled in a seawater heat exchanger before returning to the inlet of EA1.

Liquefaction

The liquefaction consists of the E2 heat exchanger. The natural gas entering at dew point is partly condensed after leaving E2.

The refrigerant is a mixture of methane, ethylene, ethane and propane. The refrigerant enters at the inlet to E1A and is cooled in E1A, E1B and E2 before being pressure relieved through a Joule-Thompson valve and used as a cooling agent in E2.

The compression of the refrigerant is in two stages with an intermediate seawater cooler. The refrigerant is also cooled in a seawater heat exchanger before entering E1A.

Sub-cooling

The sub-cooling consists of the E3 heat exchanger. The partially condensed natural gas is fully condensed and the LNG is further sub-cooled.

The refrigerant is a mixture of nitrogen, methane, ethylene and ethane. The refrigerant is cooled through E1A, E1B and E2 before being pressure relieved in the X1 turbine and further through the valve.

The refrigerant is compressed in the same manner as the liquefying refrigerant.

1.4 Challenging aspects of the LNG system

The LNG system contains almost only standard process equipment that has well known models that function acceptably in dynamic process simulation tools. The non-standard process units are the main heat exchangers, for cooling and condensing the natural gas. These heat exchangers are either plate-fin or coil-wound heat exchangers for large LNG plants. The greatest challenge when modelling and simulating an LNG plant will be modelling these heat exchangers, with phase transitions, and the high pressure of the natural gas stream.

1.4.1 The Heat Exchanger

Figure 1.2 shows the layout of a coil/spiral wound heat exchanger with four process streams. From the picture it is seen that one process stream, the shell side stream, counterflows the three other streams, the tube streams. The orientation of the heat exchanger when installed, is the same as in the picture. The shell side stream is therefore down-flowing, and the tube stream are up-flowing. Heat is only exchanged directly between the shell stream and the tube streams.

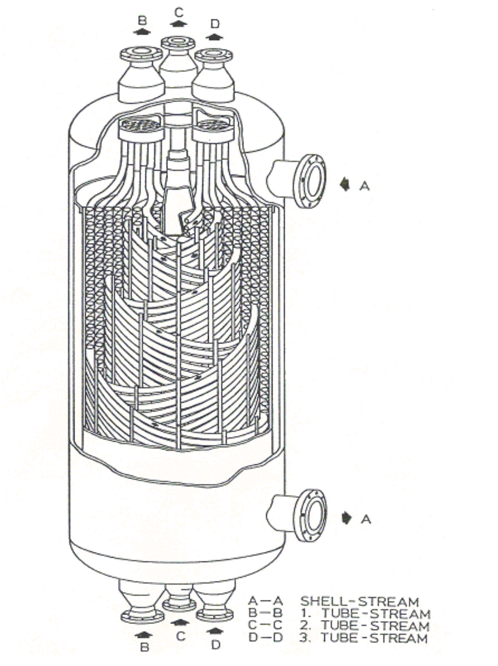


Figure 1.2: Coil-wound heat exchanger

The tube side flows are relatively high to high pressure, 10-65 bar, streams, while the shell stream is a low pressure, 2-6 bar, stream. The NG stream will typically have a pressure of 40-65 bar, and may be close to its critical point as it flows through the heat exchanger system. The critical point of pure methane is, according to Perry, Green, and Maloney (1999), $T_c = 190.56$K, $P_c = 45.9$ bar. The critical point for the natural gas mixture will be somewhat close to this.

When modelling these units several simplifications must be considered. The main objective when making simplifications to models, is to get a simple model as possible that captures the dynamics of the desired time scale. The other, faster, dynamics are neglected. A simple model is wanted, due to clarity and model understanding, and the CPU demand when integrating the model. The detailed model for the heat exchanger is given in Section 5.2. The simplifications will be discussed in Section 5.2.3.

1.4.2 High pressure - critical point

The generally high pressure of the natural gas will place requirements on the modelling. High pressure requires an EOS approach to simulate equilibrium, and to detect the correct phase.

This creates problems especially when modelling columns, where one of the common simplifications is to drop hold-up in the gas phase. According to Choe and Luyben (1987) simplifications are not viable over moderate pressures (10ATM). Also if one of the streams into the column is in a dense gas state, it can create difficulties when calculating the various flows in simplified schemes.

1.5 Available software for dynamic process simulation

Today there are several commercial software packages available for simulating general chemical process plants. The number of academic modelling and simulation environments are huge, and only a few can be mentioned.

The most common commercial products are the Aspen Dynamics, HYSYS Dynamic^{TM} and gPROMS. Both Aspen Dynamics and HYSYS are products from AspenTech, and are extendable with the Aspen Custom Modeler for the incorporation of custom models. Aspen Dynamics is the commercialization and further development of SPEEDUP, an equation oriented simulator developed at Imperial College over a period of 25 years (Pantelides (1988)). HYSYS Dynamic^{TM} is a dynamic pressure-flow network solver.

gPROMS, Oh and Pantelides (1996) and Barton (1992), the general PROcess Modelling System, is a product of Process Systems Enterprise Limited (PSE), which is a spin-off company from Imperial College. Costas C Pantelides at Imperial College was central both in development of SPEEDUP and gPROMS modelling software. Paul I. Barton contributed to the first version of gPROMS, but is currently involved with the academic software ABACUSS (Advanced Batch And Continuous Unsteady State Simulator), (Tolsma, Clabaugh, and Barton (2004)), at the department of Chemical Engineering Massachusetts Institute of Technology.

ABACUSS, gPROMS and SPEEDUP share a common intellectual heritage, so their use should be similar. Another academic software product that is developed over many years at department of Chemical Engineering, Carnegie Mellon University, is ASCEND (Piela, McKelvey, and Westerberg (1993) and Piela et al. (1991)). ASCEND is an acronym for 'Advanced System for Computations in ENgineering Design'. Arthur W. Westerberg has been central in its development. ASCEND like ABACUSS, gPROMS and SPEEDUP is a equation-oriented simulator with its own modelling language.

DIVA, 'Dynamische sImulation Verfahrenstechnischer Prozesse', has been developed over the last 15 years mainly at the 'Institut für Systemdynamik und Regelungstechnik' at the University of Stuttgart and since 1998 also at the 'Max-Planck-Institut für Dynamik komplexer technischer Systeme' in Magdeburg. DIVA, Trankle et al. (2000), Kroner et al. (1990) and Holl, Marquardt, and Gilles (1988), is an academic software product and the two people who are central in the developmnet are Ernst D. Gilles and Wolfgang Marquardt. Marquardt is currently at Process Systems Engineering, RWTH Aachen University, and has been developing the process modeling language VeDa (Verfahrenstechnisches Datenmodell).

The capabilities and status of the different software packages are presented in Section 2.11.

1.6 Motivation for the thesis

The goal of this work is to produce a general simulator for LNG plant simulation, simulating the test case LNG plant faster than real time using rigorous thermodynamics. The models are based on traditional first principles modelling.

Two methods are investigated, and the different dynamic responses they produce are demonstrated. The first method is simultaniously solving all the equations. The second is a network approach using a split integration scheme. The pressure flow dynamics is solved in an implicit scheme while energy and compositional dynamics are integrated assuming fixed pressures and flows. This split integration is a common industrial approach for solving process flowsheet problems.

The unit models that describe the process modules typically present in an LNG plant are implemented in a modular manner that made it possible to use the same implementation for both integration schemes.

Chapter 2

Methods for Dynamic Process Simulation

Process simulation revolves around the flowsheet. A flowsheet is an abstract graphical representation of a chemical plant. In its nodes it contains the relevant single process step, a unit operation. The term flowsheeting is used as a synonym for the simulation of chemical plants.

The process equipment is modelled as unit operations that generally must satisfy three physical laws.

1. Mass conservation
2. Energy conservation
3. Momentum balance

A unit operation can also be a distinct part of a more complex process step.

The time constants arising from the equations describing the momentum balance are several orders of magnitude smaller than the time constant associated with mass and energy conservation. Also the mass and energy dynamics dominate the system dynamics, therefore mass and energy conservation are formulated dynamically. The time dependence in the momentum transport is simplified, and then implemented on a quasi-stationary form, (Eich-Soellner et al. (1997)).

The mass and energy balance therefore gives rise to ordinary differential equations, ODE, and/or partially differential equations, PDE. The momentum balance in stationary form is an algebraic equation, AE.

Mathematically, dynamic process simulations are governed by large systems of differential-algebraic-equations, DAE, with discontinuities. Principally simulators for dynamic processes follow two different concepts in setting up and solving the plant model.

- modular approach
    - simultaneously modular
    - sequential modular
    - network modular
- equation-oriented approach

In the modular approach each module is a "black box" for the flowsheet integration, and produce output given input. In an equation-oriented system the entire equation set is available for the integration routine, and all equations are solved simultaneously with the same algorithm. Further information about equation-oriented and modular simulation will be given in Section 2.10 and Section 2.12, respectively.

2.1 Differential Algebraic Equations - DAE

The equation set depends on the considerations made during process abstraction. Assuming well-mixed (quasi-homogeneous) control volumes the set will be a lumped parameter model, containing explicit ODEs and AEs, see Marquardt (1995). If spatial effects are considered in the modelling, the resulting equation set will be a combination of PDEs, ODEs and AEs. That is PDAEs, partial differential algebraic equations.

In the case of PDEs it is common to transform such equations to an ODE form, using the method of lines (MOL). Then the spatial differentials are typically substituted by a first order approximation. Time then becomes the only independent variable, and the equation set has a DAE form.

The ODEs will have the general form given in Equation 2.1.

$$\frac{d\Psi}{dt} = \Phi_\Psi + \Sigma_\Psi \tag{2.1}$$

Here Ψ is an arbitrary extensive quantity stored in a control volume. Φ_Ψ represents the fluxes of Ψ in and out of the control volume, and Σ_Ψ are the sources/sinks of Ψ in the control volume. The Ψ is, in this thesis, an abstraction for mole numbers, pressure, integral controller error, enthalpy or internal energy.

The PDEs can be expressed in the same manner, and the general one-dimensional PDE is shown in Equation 2.2.

$$\frac{\partial\Psi}{\partial t} + \frac{\partial}{\partial x}(\Psi v + \Phi_\Psi) = \sigma_\Psi \tag{2.2}$$

Here, in addition to the definitions given under Equation 2.1, the v represents the velocity. The σ_Ψ is a source. Transformation of Equation 2.2 by MOL will give an equation of the form in Equation 2.1.

The mass and energy conservation give explicit ODEs, and the momentum equation gives AEs when treated as quasi-stationary. The majority of the AEs are originated from constitutive equations which relate the extensive quantities as well as fluxes and sources to intensive quantities such as temperature, pressure and concentration. These relations can be in explicit form, but generally implicit AEs also occur. One example of implicit AEs is the thermodynamic equation of state (EOS).

The general abstraction for a implicit DAE equation set is given in Equation 2.3.

$$\mathbf{h}(\dot{\mathbf{y}}, \mathbf{y}, \mathbf{z}) = \mathbf{0} \tag{2.3}$$

The dynamic state variables, only denoted as state variables hereafter, are defined as the vector $\mathbf{y}$. Vector $\mathbf{z}$ represent the algebraic variables. The function vector $\mathbf{h}$, is generally nonlinear. A linear implicit DAE equation set is given in Equation 2.4.

$$\begin{aligned} \mathbf{B}(\mathbf{y}, \mathbf{z})\dot{\mathbf{y}} &= \mathbf{f}(\mathbf{y}, \mathbf{z}) \\ \mathbf{0} &= \mathbf{g}(\mathbf{y}, \mathbf{z}) \end{aligned} \tag{2.4}$$

The matrix $\mathbf{B}$, and the AE function vector $\mathbf{g}$, can be nonlinear. The modelling in this thesis results in a constant matrix $\mathbf{B}$, where $\mathbf{B} = \mathbf{I}$. The AE function vector is nonlinear.

The equation set considered in this thesis becomes:

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{f}(\mathbf{y}, \mathbf{z}) \\ \mathbf{0} &= \mathbf{g}(\mathbf{y}, \mathbf{z}) \end{aligned} \tag{2.5}$$

The state variables will be a set of holdup, energy and controller variables. Adding controllers with integral action results in dynamic states, and algebraic equations.

It is possible to substitute the AEs, of an index one DAE system, into the right hand side of the ODEs in Equation 2.5, and the equation set becomes a pure ODE system.

2.2 Numerical integration

The modelling of chemical engineering systems will almost always end up as a stiff system. A problem is stiff if the integration step length is restricted by stability rather than accuracy. There are several definitions of stiff models and stiffness, as shown by Hairer and Wanner (1996). The stiffness is caused by large eigenvalues of the Jacobian matrix.

If λ_1,.....,λ_n are the eigenvalues of the Jacobian matrix, the stiffness can be defined as in Equation 2.6.

$$\text{Stiffness} = \frac{\underset{1\leq i\leq n}{max} |\lambda_i|}{\underset{1\leq j\leq n}{min} |\lambda_j|} \tag{2.6}$$

At the same time discontinuities will occur both in time and state. This will require special attention when integrating the system. The system needs to detect discontinuities, and when they occur in time. After locating them the integration needs to be restarted efficiently.

When integrating stiff differential equations, the integration step size will not be controlled by accuracy, but rather by stability. In order to solve these types of problems it is necessary to use an A-stable integration method. A-stable methods are stable if the system's eigenvalues lie in the left hand plane of a real-imaginary Cartesian coordinate system.

Explicit methods are not generally A-stable, and therefore cannot be used on stiff systems, such as large scale chemical engineering problems. Therefore, an implicit algorithm must be used. The implicit algorithms will generate implicit algebraic equations, that need to be solved at every time step. To solve this equation set, Newton's method is usually the algorithm of choice. The Newton approach requires the Jacobian of the system. In general when solving large and sparse linear systems, iterative methods with sparse handling are preferred. This is partly because of the reduced number of mathematical operations, and partly because the truncation error grows large when solving linear systems with a direct method, see Golub and Van Loan (1996). A direct method will in practice mean solving the system with LU decomposition. An iterative approach will converge fast when using a good starting point, as is available in dynamic simulations.

According to Hairer and Wanner (1996), one of the following methods for integration must be used to solve a stiff DAE/ODE problem:

- Semi-implicit or fully implicit one-step methods
- Multi-step methods for stiff systems

The most common and most used one-step approach is the Runge-Kutta (RK) method. The most common multi-step method is the backward differential formula (BDF) method. The BDF computer codes are the most prominent and most widely used for all kinds of stiff problems. Both are described below.

Several free software packages are available for download, e.g. www.netlib.net.

2.2.1 Runge-Kutta (RK)

The methods named after Carl Runge and Wilhelm Kutta are designed to imitate a Taylor integration without requiring analytical differentiation of the original differential equation, see Cheney and Kincaid (1999). The RK method can imitate a Taylor expansion of any order, and be both explicit and implicit. Only methods with implicitness will suffice in the current case.

Because of the formulation, the RK method will need cheap first derivative information in order to integrate efficiently.

Considering the function, $\dot{y} = f(z, y)$, the generalized implicit RK method is given in Equation 2.7.

$$\begin{aligned} g_i =& y_0 + h\sum_{j=1}^{s} a_{ij} f(z_0 + c_j h, g_j) \\ y_1 =& y_0 + h\sum_{j=1}^{s} b_j f(z_0 + c_j h, g_j) \end{aligned} \tag{2.7}$$

A special type of RK methods is the Rosenbrock type. The Rosenbrock approach tries to avoid nonlinear systems, found when solving ordinary RK, by replacing them with a sequence of linear systems. These methods are therefore called "semi-implicit" or linear implicit RK methods. These methods are equal to RK, using only one Newton iteration before accepting the new state vector.

Considering the function, $\dot{y} = f(y)$, the general s-stage Rosenbrock method is given in Equation 2.8.

$$\begin{aligned} k_i =& hf(y_0 + \sum_{j=1}^{i-1} \alpha_{ij} k_j) + hJ\sum_{j=1}^{i} \gamma_{ij} k_j, \quad i = 1, ..., s \\ y_1 =& y_0 + \sum_{j=1}^{s} b_j k_j \end{aligned} \tag{2.8}$$

where $\alpha_{ij}, \gamma_{ij}, b_j$ are the determining coefficients, and $J = f'(y_0)$.

2.2.2 Backward differential formulas (BDF)

The generalized multi-step method for integrating $\dot{y} = f(y)$ is given in Equation 2.9.

$$\alpha_k y_{m+k} + \alpha_{k-1} y_{m+k-1} + \cdots + \alpha_0 y_m = h(\beta_k \dot{y}_{m+k} + \cdots + \beta_0 \dot{y}_m) \tag{2.9}$$

BDF methods are a variant of the multi-step approach, using a fully implicit step, combined with a k-th order polynomial through k+1 points. This polynomial is differentiated and the resulting relation must be satisfied at time k and k+1.

The general BDF formula, using a k-th order polynomial, has the following form:

$$\sum_{j=1}^{k} \frac{1}{j} \Gamma^{j} y_{n+1} = h f_{n+1} \tag{2.10}$$

The Γ operator is defined in Equation B.23.

It can be seen from Equation 2.10 that the integration method is not a self starter, because it needs the state values at k-1 points prior to the starting point. This can be solved by using a variable order BDF algorithm. A detailed description of BDF methods is given by Hairer and Wanner (1996).

2.3 DAE index

According to Hairer and Wanner (1996), the general DAE can be classified by a differential or a perturbation index. Only the differential index will be reviewed in this section. The literature mentions several other indexes for DAE equations.

Consider the general DAE:

$$\mathbf{f}(\dot{\mathbf{s}}, \mathbf{s}, t) = 0, \tag{2.11}$$

where $\dot{\mathbf{s}} = d\mathbf{s}/dt$. Then the differentiation/differential index, v, is defined as the minimal number of times v, the equations has to be differentiated in order to determine $\dot{\mathbf{s}}$ as a continuous function of $\mathbf{s}$ and t. This is the definition given by Martinson and Barton (2000).

The equations considered in this thesis are the index one type unless otherwise stated. The index one problem is the simplest case of differential algebraic equations. The higher the index of a DAE problem, the more challenging the numerical integration becomes. A system is considered *high index* if $v \geq 2$.

A fully determined system of ODE is of differential index-0. For a differential index-1 system the algebraic equations typically appear in explicit form. A system with differential index-2 or higher will have implicit algebraic equations, which must be satisfied through the integration. These are 'hidden constraints'. The characteristics of differential index-1 and index-2 systems will be briefly discussed below.

The differential index is often considered using the *derivative array equation*, see Campbell and Gear (1995). Before defining the derivative array equation, some notation is defined in Equations 2.12 and 2.13.

$$\mathbf{S}_{[i]} = \begin{bmatrix} \frac{d}{dt}\mathbf{s} \\ \left(\frac{d}{dt}\right)^2 \mathbf{s} \\ \vdots \\ \left(\frac{d}{dt}\right)^i \mathbf{s} \end{bmatrix} \tag{2.12}$$

$$\left(\frac{d}{dt}\right)^i \mathbf{f}(\dot{\mathbf{s}}, \mathbf{s}, t) = \mathbf{f}_{[i]}(\dot{\mathbf{s}}, \mathbf{s}, t) \tag{2.13}$$

The k^{th} derivative array equation, $\mathbf{F}_{[k]}$, can then be defined, Equation 2.14.

$$\mathbf{F}_{[k]}(\mathbf{S}_{[k+1]}, \mathbf{s}, t) = \begin{bmatrix} \mathbf{f}_{[0]}(\dot{\mathbf{S}}_{[1]}, \mathbf{s}, t) \\ \mathbf{f}_{[1]}(\dot{\mathbf{S}}_{[2]}, \mathbf{s}, t) \\ \vdots \\ \mathbf{f}_{[k]}(\dot{\mathbf{S}}_{[k+1]}, \mathbf{s}, t) \end{bmatrix} \tag{2.14}$$

The differentiation index is then the smallest k such that $\mathbf{F}_{[k]}$ uniquely determine $\dot{\mathbf{s}}$ as a continuous function of $\mathbf{s}$ and t.

2.3.1 Index-1 system

Consider the general differential algebraic equation set given in Equation 2.15.

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{f}(\mathbf{y}, \mathbf{z}) \\ \mathbf{0} &= \mathbf{g}(\mathbf{y}, \mathbf{z}) \end{aligned} \tag{2.15}$$

Time differentiating the algebraic equations in Equation 2.15 we get Equation 2.16.

$$\begin{aligned} \mathbf{0} &= \mathbf{g}_{\mathbf{z}}(\mathbf{y}, \mathbf{z})\dot{\mathbf{z}} + \mathbf{g}_{\mathbf{y}}(\mathbf{y}, \mathbf{z})\dot{\mathbf{y}} \\ &\Updownarrow \quad (\mathbf{g}_{\mathbf{z}} - \text{non-singular}) \\ \dot{\mathbf{z}} &= -\mathbf{g}_{\mathbf{z}}^{-1}(\mathbf{y}, \mathbf{z})\mathbf{g}_{\mathbf{y}}(\mathbf{y}, \mathbf{z})\mathbf{f}(\mathbf{y}, \mathbf{z}) \end{aligned} \tag{2.16}$$

From the definition of the differential index it is seen that the system of differential algebraic equations given in Equation 2.15 is of index one if $\mathbf{g}_{\mathbf{z}}$ is non-singular.

2.3.2 Special index-1 system

Differential index-1 systems with one or more implicit/hidden constraints are sometimes called 'special index one'. An example of a linear special index-1 system is given by Martinson (2000). The system is shown in Equation 2.17.

$$\begin{aligned} \dot{y}_1 + \dot{z}_2 &= f_1(t) \\ y_1 + 3y_2 &= f_2(t) \end{aligned} \tag{2.17}$$

For the initial conditions to be consistent for this equation, they must satisfy Equation 2.18.

$$\dot{y}_1 + 3\dot{y}_2 = \dot{f}_2(t) \tag{2.18}$$

2.3.3 Index-2 system

Consider the general differential algebraic equation set given in Equation 2.19.

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{f}(\mathbf{y}, \mathbf{z}) \\ \mathbf{0} &= \mathbf{g}(\mathbf{y}) \end{aligned} \tag{2.19}$$

Time differentiating the algebraic equations in Equation 2.19 we obtain Equation 2.20, a 'hidden constraint'.

$$\mathbf{0} = \mathbf{g_y f} \tag{2.20}$$

Differentiating once more gives Equation 2.21.

$$\begin{aligned} \mathbf{0} &= \mathbf{g_{yy} f^2} + \mathbf{g_y f_y f} + \mathbf{g_y f_z \dot{z}} \\ &\Updownarrow \quad (\mathbf{g_y f_z} - \text{non-singular}) \\ \dot{\mathbf{z}} &= -(\mathbf{g_y f_z})^{-1}(\mathbf{g_{yy} f^2} + \mathbf{g_y f_y f}) \end{aligned} \tag{2.21}$$

That is; the system has differential index-2 if $\mathbf{g_y f_z}$ is non-singular.

The system of differential algebraic equations given in Equation 2.19 represents the larger class of problems of the type given in Equation 2.15, where $\mathbf{g_z}$ is singular. This is shown by Hairer and Wanner (1996).

2.3.4 Index in VLE systems

The index of vapour-liquid equilibrium (VLE) systems is discussed by Ponton and Gawthrop (1991), and Pantelides et al. (1988). Ponton and Gawthrop (1991) show that the general formulation, shown in Equation 2.22 is index-1. Moe, Riksheim, and Hertzberg (1996), Ponton and Gawthrop (1991) and Pantelides et al. (1988) stress that specifications to the equation sets often introduce an index problem. This increased index is usually because the differential variables emerging from the material and energy balances are dependent. They are dependent through the VLE relationship.

Dynamic modelling of chemical process units often introduces specifications to the pressure to reduce the stiffness of the system. This leads to an index-2 DAE system. Moe, Riksheim, and Hertzberg (1996) give guideline about how these problems can be solved by first reducing the index.

A typical VLE system is shown in Figure 2.1. The fixed volume container is assumed to have a homogeneous pressure, P, and temperature, T. Energy, E, and every component species are conserved. The energy can be enthalpy or internal energy. The inlet flow, W_I, is an algebraic relation in properties upstream the container that is typically dependent on the variables shown in Figure 2.1. The outlet flows, W_V and W_L, are described in a similar manner.

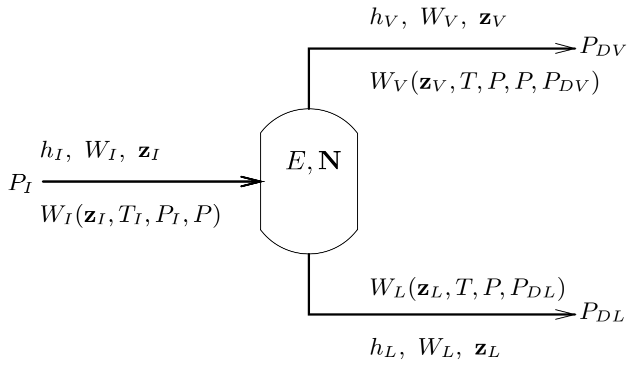

$h_V,\ W_V,\ \mathbf{z}_V$
P_{DV}
$W_V(\mathbf{z}_V, T, P, P, P_{DV})$
$h_I,\ W_I,\ \mathbf{z}_I$
P_I
$W_I(\mathbf{z}_I, T_I, P_I, P)$
$E, \mathbf{N}$
$W_L(\mathbf{z}_L, T, P, P_{DL})$
P_{DL}
$h_L,\ W_L,\ \mathbf{z}_L$

Figure 2.1: VLE system

If the VLE system is described using the overall energy and component holdups such as the differential variables defined by Equation 2.22. The system will be index-1 if algebraic relations give the pressure-driven flow.

$$\begin{aligned}\frac{dE}{dt} &= W_I h_I - W_V h_V - W_L h_L \\ \frac{d\mathbf{N}}{dt} &= W_I \mathbf{z}_I - W_V \mathbf{z}_V - W_L \mathbf{z}_L\end{aligned} \tag{2.22}$$

More specifically, it will be a stiff semi-explicit differential index-1 DAE system. *Semi-explicit* means equations of the form shown in Equation 2.23, where the differential variables are defined explicitly, and the algebraic equations are formulated implicitly.

$$\begin{aligned}\dot{\mathbf{y}} &= \mathbf{f}(\mathbf{y}, \mathbf{z}) \\ \mathbf{0} &= \mathbf{g}(\mathbf{y}, \mathbf{z})\end{aligned} \tag{2.23}$$

2.4 Initialisation of DAE's

Initial conditions are the values of $\dot{\mathbf{s}}$ and $\mathbf{s}$ at time t_0. The initial conditions are denoted $\dot{\mathbf{s}}_0$ and $\mathbf{s}_0$. To be consistent, the initial conditions must satisfy

$$\mathbf{f}(\dot{\mathbf{s}}_0, \mathbf{s}_0, t_0) = 0 \tag{2.24}$$

The initial conditions must also satisfy any implicit constraints the DAE set may have.

Variables $\mathbf{s}_0$ or their derivatives $\dot{\mathbf{s}}_0$ which can be assigned arbitrary values and still allow the original system to be solved, are called *dynamic degrees of freedom.*

For an explicit ordinary differential equation of the form:

$$\dot{\mathbf{s}} = \mathbf{f}(\mathbf{s}, \mathbf{t}) \tag{2.25}$$

the initial conditions normally refer to a set of values for $\mathbf{s}$ at t = 0. That is; the dynamic degrees of freedom in an ODE system are equal to the number of dynamic variables.

For a DAE set of equations, the initialisation problem is more complicated. Having n differential equations and m algebraic equations the number of unknowns is 2n+m. In order to provide consistent initial conditions, n additional specifications are required. In the case where no implicit constraints are present in the equations, there are $(2n+m)!/(n+m)!n!$ possible sets of n initial values. Initial values can be given as differential and/or algebraic variable values and/or the values of the differentials. The equation set of n+m equations can then, in principle, be solved to produce consistent initial values for all the variables in the system.

Having a high differential index, or having implicit constraints will reduce the dynamic degrees of freedom for the DAE system. The number of initial variable values is reduced by the number of implicit constraints.

One of the difficulties of the initialisation is the identification of the hidden constraints. Another difficulty is the possible necessity of reducing the index in order to be able to integrate. Pantelides et al. (1988) provide an algorithm for finding hidden constraints. Brown, Hindmarsh, and Petzold (1998) outline the algorithm for calculating consistent initial values in DASPK, formerly DASSL. Bachmann et al. (1990) give a general algorithm for index reduction, and Moe, Riksheim, and Hertzberg (1996) show how the index can be reduced for special types of chemical process models.

2.5 Large-Scale systems

Large-scale systems are differential algebraic equation sets with 1000 to 100 000 (or more) dynamic variables. To efficiently integrate stiff DAE systems of this magnitude requires the following among other thing:

☞ A fast function evaluation

☞ A fast Jacobian evaluation

☞ A fast linear solver

☞ Intelligent integration

If these qualities are not avaliable, an implementation solving large-scale DAE problems might be too slow to be practical. The computational demand often depends on the correlation methods used for the AEs. Using rigorous thermodynamics will slow down the function evaluation considerably. If the function evaluates slowly, and cannot be speeded up by further simplifications, the only possible outcome is to evaluate this on a parallel computer. Parallel computing will not be addressed in this thesis.

The computation of the Jacobian, or partial differentials, must be performed by one of the five methods described in Section 2.6. Other methods for integrating DAE/ODE systems are shown in Section 2.2. The ability to block decompose a large system and solve subsystems affects the overall performance of a simulator. The principle of block decomposition is given in Section 2.7. It is important to have the ability to efficiently solve the linear equations, emerging in every iteration of the non-linear Newton-Rapshon algorithm. Different approaches for solving asymmetric linear equations are shown in Section 2.8. The ease with which one can configure and initialise a system as large as this is also of importance. At least this is a concern for commercial interests.

2.6 Computational differentiation

Computational differentiation is the calculation of derivatives on a computer.

Tolsma and Barton (1998) review several methods for obtaining numerical derivatives. Their paper reviews: hand-coding, finite difference approximations, reverse Polish notation evaluation, symbolic differentiation, and automatic differentiation (AD). After systematically going through the various methods, giving pros and cons, they conclude that automatic differentiation has significant advantages over all the other approaches.

The hand-coding, finite difference approximations, reverse Polish notation evaluation and the symbolic differentiation all work with the equations. The automatic differentiation approach is designed to differentiate computer programs.

All methods, except the finite difference approach, are exact, and free of truncation error. However, all such methods are affected by roundoff error. These methods are reviewed in this section.

2.6.1 Hand coding

This involves the analytical differentiation of the equations used. The differentials are then implemented in subroutines, producing exact differentials. This work can be time consuming and error-prone.

2.6.2 Finite difference

This approach involves variable perturbations and function evaluations. This method is described in Cheney and Kincaid (1999). In order to produce a Jacobian all the variables must be perturbed, and the function set describing the DAEs must be evaluated at every perturbation. If an equation set has n variables, the cost of evaluating the Jacobian is n+1 times the cost of a function evaluation.

This method is associated with truncation and roundoff error, and will be less accurate than all the other methods discussed. The optimal perturbation, minimising the overall error, varies with the value of the variables, and is therefore difficult to find/guess. The advantage is that this method is simple to implement.

2.6.3 Reverse Polish Notation

The Polish mathematician Jan Lukasiewicz invented the Polish Notation in the 1920s. Lukasiewicz showed that by writing operators in front of their operands, instead of between them, brackets were made unnecessary. Later Charles L. Hamblin proposed a scheme in which the operators follow the operands (postfix operators), resulting in the Reverse Polish Notation (RPN). The advantage is that the operators appear in the order required for computation. Hewlett-Packard introduced their first 'calculator' using RPN in 1968.

Using RPN, the example '(1+2) x 3' becomes '3 2 1 + x'. The expressions are evaluated with the use of a stack. That is; a *first in*, *last out* data structure. Everyone having worked with stacks knows the terms *pushing* and *popping*, for adding and removing elements from the stack. Starting from the left and moving right in the RPN expression, the constants and variables are pushed onto the stack. When encountering a binary operation, the two top elements are popped for evaluation with the operator. The result is then pushed back in the stack. If a unary operator or function is encountered only one element is popped, and the result is pushed to the stack. After finishing the RPN expression, only one element remains in the stack that is the result of the expression.

By handling a differential stack in parallel with the main evaluation stack, the differentials are calculated. The procedure for calculating them is the same as above, but the laws of differentiation must be applied. For details on RPN, see Ponton (1982) and Koup et al. (1981).

The time required for calculating these differentials is similar for finite difference and symbolic differentiation.

2.6.4 Symbolic differentiation

In symbolic differentiation the equations will be represented by a rooted directed binary tree, Cormen et al. (2001). A rooted tree consists of a set of vertices and a set of edges. Each edge links a parent node to one of the parent's children. A special root node has no parent. Every other vertex has exactly one parent. It is possible to reach any vertices by following a unique path of edges from the root.

The interior vertices represent an intrinsic function (ln, sin, ...) or an operator (+,-,/,*). The leaves of the tree hold either numbers or variables. The equations will be evaluated by recursively traversing this tree structure.

When having the equations stored in this tree structure, they can be differentiated, by applying the rules of differentiation recursively to the tree. For an introduction to symbolic differentiation, see 'www.SymbolicNet.org'.

An example of a computer program using symbolic representation as well as symbolic differentiation is Maple. Depending on the implementation of the symbolic differentiation the evaluation of the differentials may be costly. However, it will be exact.

2.6.5 Automatic differentiation

Automatic differentiation (AD) is a chain-rule-based technique for evaluating the differentials with respect to the input variables of functions defined by a high-level language computer program. AD relies on the fact that all computer programs use a finite set of intrinsic functions (ln, sin, etc.) and unary/binary operators (+,-,/,*). The coded functions representing the equations are a composition of these elementary functions. Knowing the partial derivatives of these elementary functions, the overall derivatives can be computed using the chain rule. That is; the process of automatic differentiation.

AD has two basic modes, the *forward mode* and the *reverse mode.* Werma (2000) gives a brief and simple introduction to AD. To illustrate the two modes, a vector valued function $f : \Re^n \rightarrow \Re^m$ is defined in Equation 2.26, as is done by

Forth et al. (2004).

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \tag{2.26}$$

The computation of $\mathbf{y}$ in a computer will then involve the intermediate variables $\mathbf{w} \in \Re^p,\ p \gg m + n$. (Here the '$\gg$' refers to the general case.)

Then variables are grouped as follows:

- ☞ *Independent:* $\mathbf{x} \in \Re^n$. These variable values must be supplied.
- ☞ *Intermediate:* $\mathbf{w} \in \Re^p$. These variables are calculated from $\mathbf{x}$, directly or through other elements of $\mathbf{w}$, using intrinsic functions and operators. The $\mathbf{w}$ is built following the operations from $\mathbf{x}$ to $\mathbf{y}$, and w_k is therefore a function of $w_j,\ j \leq k$ or elements of $\mathbf{x}$.
- ☞ *Dependent:* $\mathbf{y} \in \Re^m$. Variables to be calculated. The differentials $\partial\mathbf{y}/\partial\mathbf{x}$ are required.

Considering the calculation of w_k through calculated elements of $\mathbf{w}$. The unary operations are represented by Equation 2.27, and the binary operations are represented by Equation 2.28.

$$w_k = f^k_{intrinsic}(w_i),\ \ i < k \tag{2.27}$$

$$w_k = f^k_{intrinsic}(w_i, w_j),\ \ i < k,\ \ j < k \tag{2.28}$$

Forward mode

The forward mode of AD creates a new code that, for each of the variables in the code, calculates the numerical values of the variable and its derivatives with respect to the independent variables. That is; using the chain rule-based computation of Equation 2.29 for every intermediate variable up to $\mathbf{y}$, it will add up to $\partial\mathbf{y}/\partial\mathbf{x}$.

$$\frac{\partial w_k}{\partial x_l} = \left(\frac{\partial f^k_{intrinsic}}{\partial w_i}\right)\frac{\partial w_i}{\partial x_l} + \left(\frac{\partial f^k_{intrinsic}}{\partial w_j}\right)\frac{\partial w_j}{\partial x_l},\ \ 1 \leq l \leq n \tag{2.29}$$

The forward mode is sometimes referred to as direct or tangent linear mode.

Reverse mode

Reverse mode AD produces a code that passes forward through the original code storing the partials/sensitivities required for a reverse pass. Here the entire computation trace must be stored which can be demanding in terms of memory. For the binary function/operation in Equation 2.28 the two partials shown in Equation 2.30 will be calculated and stored for the reverse pass.

$$\frac{\partial y_l}{\partial w_i} = \left(\frac{\partial f^k_{intrinsic}}{\partial w_i}\right)\frac{\partial y_l}{\partial w_k},\qquad \frac{\partial y_l}{\partial w_j} = \left(\frac{\partial f^k_{intrinsic}}{\partial w_j}\right)\frac{\partial y_l}{\partial w_k},\ \ 1 \leq l \leq n \tag{2.30}$$

A thorough introduction to AD may be found in Griewank (2000). The reverse mode is also called backward, adjoint or cotangent linear mode.

AD - applications

There are several AD tools for codes written in Fortran, C/C++ and Matlab. A useful source for information on AD and AD tools is 'www.autodiff.org'.

According to Forth et al. (2004), these tools implement AD in one of two ways, *source transformation* (precompilation) or *operator overloading.* Source transformation involves sophisticated compiler techniques and the generation of new code. The new code will calculate the necessary intermediate differentials/sensitivities as well as values for the dependent variables. Examples of precompiling AD software are the Fortran tools ADIFOR, Bischof et al. (1998), and TAMC, Giering and Kaminski (1998), and the C/C++ tool ADIC, Bischof, Rho, and Mauer (1997). The overloading approach utilises the overloading functionality of object-oriented programming languages. Defining new types with storage containers for the auxiliary values needed in the differential calculations and extending the intrinsic functions and operations to the new variables will produce the needed differentials. An example of an AD package using operator overloading is the ADOL-C, Griewank, Juedes, and Utke (1996)

For large-scale non-linear problems a layer of sparsity exploitation above the AD is required. For efficient computation the problem structure must also be exploited.

The computational cost of a Jacobian matrix evaluation using AD techniques is less than any other of the other computational derivative methods discussed in this section.

Current AD software is capable of handling a wide variety of source code. According to Tolsma, Clabaugh, and Barton (2002), AD can handle implicit equations solved with iterative schemes. That is; EOS is handled by AD software.

2.7 Block decomposition

Barton (2000) gives a clear presentation of the concept of *block decomposition.* In order to illustrate the essence of block decomposition, the definition of the incidence (or occurrence) matrix must be known. Considering the vector valued function $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, the incidence matrix, $\mathbf{I}^f$, will show the occurrence of a variable, x_j, in equation $f_i(\mathbf{x}) = 0$. If x_j, is present in f_i, $\mathbf{I}^f_{i,j} = \times$, otherwise $\mathbf{I}^f_{i,j}$ is empty. That is; a nonzero element of the Jacobian, $\mathbf{J}_{i,j}$, matrix will correspond to $\mathbf{I}^f_{i,j} = \times$.

The objective of the block decomposition is to transform the equation set making the incidence matrix a block lower triangular with minimal blocks. This is managed through a permutation of the variables and the equations. The new variables, $\mathbf{y}$, and the new function matrix, $\mathbf{g}(\mathbf{y})$, are introduced. These are both the respective linear combinations of the original variables and functions. These transformations are shown in Equation 2.31.

$$
\mathbf{I}^f : \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{array}
\begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\
\left[\begin{array}{ccccc}
\times & & & \times & \\
 & \times & \times & \times & \times \\
\times & \times & & \times & \times \\
\times & & & \times & \\
\times & & \times & & \times
\end{array}\right] \end{array}
$$

$$
\Updownarrow \tag{2.31}
$$

$$
\mathbf{I}^g : \begin{array}{c} \\ f_1 \\ f_4 \\ f_3 \\ f_5 \\ f_2 \end{array}
\begin{array}{c} \begin{array}{ccccc} x_1 & x_4 & x_2 & x_3 & x_5 \end{array} \\
\left[\begin{array}{cc|c|cc}
\times & \times & & & \\
\times & \times & & & \\
\hline
\times & \times & \times & & \\
\hline
\times & & & \times & \times \\
 & \times & \times & \times & \times
\end{array}\right] \end{array}
$$

The advantage of the block lower triangular form is the solution process by sequential solution of sub-problems. Starting with an upper block diagonal sub-matrix, the overall system of equations can be solved in series taking the diagonal block by block downwards in the matrix. Block decomposition has the following advantages:

- ☞ The solution of a large system of equations is broken down to smaller systems. The solution becomes computationally faster and requires less computer memory.
- ☞ Having solved the first sub-problem, a partial solution can exist when solving the remaining sub-problems. This partial solution will improve the robustness of the locally convergent algorithms applied to each sub-problem.
- ☞ Linear blocks or blocks containing one unknown variable can be solved easily.
- ☞ Differential information for the off-diagonal blocks are not required.

An algorithm for obtaining the minimal block decomposition is given by Duff and Reid (1978b) and Duff and Reid (1978a). These both use a directed graph algorithm and the algorithm of Tarjan (1972).

2.8 Linear solvers

The DAE systems in chemical process plants usually have a sparse unstructured Jacobian matrix. This can be exploited in the solution of the linear system, Equation 2.32.

$$\mathbf{Ax} = \mathbf{b} \tag{2.32}$$

Spares matrix techniques begin with the idea that zero elements need not be stored. The array elements are therefore stored in some compressed form, not storing zeros. Saad (2003) gives some detail about the storage scheme. Every compressed form will give an overhead in the tracking of array element positions. Using sparse matrix techniques also give a much faster solution of a linear system and demands fewer operations for solving the same system. The solvers for the sparse system can be divided in two groups:

- ☞ direct solvers
- ☞ iterative solvers

Most direct methods for sparse linear systems perform an LU (GAUSS) factorisation of the original matrix and try to reduce cost by minimising fill-ins. Fill-ins are nonzero elements introduced during the elimination process in positions which were initially zeros. The MA48, Duff and Reid (1996), is an example of a direct sparse linear solver. The MA48 is frequently used in equation-oriented modelling environments. The dense matrix LU factorisation is an O(n^3) operation, so much can be gained by exploiting the sparsity.

Most iterative solvers are error, $e_k = x_* - x_k$, or residual, $r_k = b - Ax_k$, projection methods. x_k is the approximate solution in iteration k, and x_* is the solution. All these methods need initial values to start, and the better the initial value; the faster they produce the solution.

Methods with projection onto Krylov subspaces have become very popular. Both the Full Orthogonalization Method (FOM) and Generalized Minimum Residual Method (GMRES) are popular, especially the GMRES method. These methods rely heavily on preconditioners to converge efficiently. For preconditioners, see Saad (2003).

2.9 Discontinuities

As stated in Section 2.2 discontinuities, also called events in the following, can occur in both time and state. According to Mao and Petzold (2002) a time event is defined to be *a priori* known, and a state event is defined to be implicitly given when a state satisfies some condition. Since the time events are known before starting the simulation, the solver is easily stopped and restarted at a discontinuity.

Examples of time events are predefined steps in boundary conditions, change in valve position, change in compressor rpm, and change of controller parameters. Boundary conditions can be flow rate, temperature, pressure, composition etc.

State events are more complicated. The logical expression related to a state event is referred to as a state condition. The mode changes whenever a state condition is satisfied. The state condition will typically be in the following form:

$$\alpha(t, \mathbf{x}) > \beta(t, \mathbf{x}) \tag{2.33}$$

The code implementation of this would involve an **IF** statement. The event is defined as the earliest time at which one of the currently pending state conditions becomes true. That is; when the state function becomes zero.

$$g(t, \mathbf{x}) = \beta(t, \mathbf{x}) - \alpha(t, \mathbf{x}) \tag{2.34}$$

Examples of state events are phase changes, and liquid flow in a column described by the Francis' weir formula. When the phase changes, the property functions of the density and the enthalpy will change considerably. The column downflow of liquid, described using Francis' weir formula, will have a discontinuity at zero flow. Also the differentials may exhibit non-continuous behaviour. This will lead to problems for the solver routine. When a state change occurs the integration should be restarted.

Other examples of state events in dynamic process simulation, are the surge control of a compressor and saturation of a valve. That is; if a recycle flow valve for a compressor is switched on or off. Both the input and output signal of a controller may saturate, resulting in a state event. A safety valve may also be trigged to open, which will change the system.

If they are not intelligently handled, the events will slow a solver with error control considerably after passing a discontinuity. When passing, the integration error becomes large, and the integration slows down making many unsuccessful function evaluations, before finding an acceptable step size. A short-lived state switching on and then off can be missed out entirely, if the current step includes its entire life span.

A possible way to remove the problem is by adding an additional equation for the discontinuity function, $z(t) = g(t, \mathbf{x}) = \beta(t, \mathbf{x}) - \alpha(t, \mathbf{x})$, to the DAE set. Alternatively the state condition can be added as a differential equation, $\dot{z}(t) = \dot{\beta}(t, \mathbf{x}) - \dot{\alpha}(t, \mathbf{x})$, to an ODE or DAE set. This is done by Carver (1978) and Wagner (1998). If the discontinuities are detected from these functions, included in the DAE/ODE system, and the function evaluation is forced to follow the state condition variable, $z(t)$, the solver sees a smooth vector field during the integration step. That is; the code uses the state event variable, $z(t)$, for checking if the function should switch to another mode required by the Equation 2.33.

It is important to detect the state events in the correct order of time. The instance of an event might change the system completely, affecting the future state events. When detecting state events with root-finding in discontinuity functions, one faces the problem of false mode switching.

Mao and Petzold (2002) give a review of several approaches for handling discontinuities. They also describe the inclusion of discontinuity handling to DASPK3.0, forming the new package DASPKE. They append a discontinuity function to a DAE system, locating state events both by looking at the discontinuity function value and its derivative at the two ends of a step (detecting even numbered zero crossings).

Approaches not using the discontinuity function are Gear and Østerby (1984) and Pantelides (1988). Gear and Østerby (1984) examine the behaviour of the local truncation error and estimate the order of the discontinuity to find a time step, which keeps the error under control while stepping over the discontinuity. Pantelides (1988) use the state conditions directly. *Locking* the equation system to the current mode during a integration step, and checking if state conditions are violated after finishing the integration step. The time of the state event is located using interpolation.

Barton (1992) and Park and Barton (1996) address the reinitialisation to a new mode. Park and Barton (1996) introduce the terminology *discontinuity sticking*, meaning a problem associated with changes in values of algebraic variables and discontinuity variables after reinitialisation in the new mode. *Locking* the equation system to the current mode during an integration step and locating the state events using some interpolation scheme, typically the interpolation formulas provided by the integrator, will give the dynamic states at the time of the state event. Reinitialising with these dynamic variables getting a consistent set of initial values will produce 'new' values for the algebraic variables. Evaluating the discontinuity functions with these values might not predict the new mode, but the same mode as before the 'state event'. This can lead to repeated detection of the same state event, and therefore the term *discontinuity sticking.*

2.10 Equation-oriented integration

The equation-oriented integration approach is characterised by the following facts. The entire equation system is solved simultaneously with one integration routine. All dynamic, and possibly all algebraic states are handled by the integrating routine. The entire structure of the equations is available for analysis.

The equation-oriented modelling environments have their own declarative modelling languages, defining variables and the equations linking them together. These high level languages are object-oriented with inheritance mechanisms. Using this modelling language, the unit models can be built, and later aggregated to describe entire flowsheet processes.

Having all the equations and variables available for the solver, possibly in a symbolic environment, the entire equation set can be analysed and solved simultaneously. This analysis can detect, and solve, possible differential index problems, and possibly block decompose the system for more efficient integration.

These formulations lead to a very large set of equations, that needs to be integrated simultaneously. This requires an differential index-1 DAE integrator with sparse handling of the linear equation solver.

Four different equation-oriented modelling environments are reviewed in Section 2.11.

2.11 Modelling and simulation environments

Several modelling software packages were presented in Section 1.5. All of them are able to solve differential index-1 differential algebraic equations of chemical process models. A short review of the capabilities of ASCEND, gPROMS, ABACUSS and DIVA will be given here. The information is based on information available on the Internet and articles.

ASCEND

ASCEND currently version IV is both a large-scale object-oriented mathematical modelling environment and a strongly typed mathematical modelling language. Although ASCEND has primarily been developed by chemical engineers, it is domain independent. ASCEND started out as a steady state solver, and the name ASCEND can be traced back to 1978. ASCEND has evolved into a general model language for implementing and integrating large-scale differential algebraic equations.

The need for a sophisticated model language is recognised in all equation-oriented simulators. The language uses concepts from semantic data modelling and object-oriented programming, with structured representation of encapsulated sub-models and a hierarchy allowing for inheritance. The languages facilitate reuse and modification of existing models. The modelling language is parsed and compiled to generate understandable code for the solver. The preprocessing of the modelling language will help the user to write well-posed models and debug and detect errors in the implementation.

The software is able to analyse the equations for structural and numerical dependencies. Structural dependencies for DAEs will trigger a differentiation to overcome index problems in the model. The numerical dependencies are used to detect model singularities. The incidence matrix is available, showing the block decomposition of the system.

The integrator used in ASCEND IV is LSODE, Hindmarsh (1980). LSODE is an initial value solver for ordinary differential equations. To cope with a large sparse linear system with up to 250 000 non-linear equations ASCEND uses partitioning and reordering algorithms that exploit the hierarchical structures inherent in the structured modelling language. The solution of the linear equation systems is performed by direct methods. The details are found in Abbott (1996).

ASCEND IV is able to use different solvers for different problems. Solver algorithms for pure non-linear algebraic equations and optimisation solvers are present, but beyond the scope of this thesis. Application of ASCEND for non-linear optimisation was reported already in 1983, Locke and Westerberg (1983).

ASCEND IV is available for download at: 'http://www-2.cs.cmu.edu/ ascend/'.

DIVA

The architecture of the equation-oriented modelling environment DIVA comprises four layers.

☞ *DIVA Simulation Kernel.* It contains a library of generic process unit models that can be aggregated to plant models by specifying flowsheets. The flowsheets can then be integrated and/or optimised. The linear-implicit DAEs are solved by one of the BDF integrators DDASAC, Caracotsios and Stewart (1985), or DAESOL. The integrators use a direct sparse solver for large sparse linear systems, (MA48, Duff and Reid (1996)). DDASAC is an extension and revision of DASSL Petzold (1983).

  The Jacobian is generated using an automatic routine for finite difference, or the FORTRAN automatic differentiation tool ADIFOR, Bischof et al. (1998). The kernel handles state events, by iteratively locating the event occurence time, and reinitialisation.

☞ *Code Generator.* The Code Generator automatically generates FORTRAN subroutines representing the DAE simulation models. The code can then be added to DIVA, and be a part of a flowsheeting project. The input files to the code generator have their own language. DIVA has a compact formulation of chemical process models and generation of efficient FORTRAN code. The input files for the code generator can be written directly by the modeller, by SyPProT or by ProMoT.

☞ *Symbolic PreProcessing Tool SyPProT.* SyPProT, Köhler, Gerstlauer, and Zeitz (2001), performs several tasks using symbolic representation of the equations and variables. It performs an index-analysis and possibly index-reduction, Mattsson and Söderlind (1993), into DAEs with a differential index-0 or index-1. The equation set is transformed into linear-implicit DAEs.

SyPProT contains functionality for transforming one-dimensional PDEs into DAEs through the method of lines (MOL). The first principles PDE models for spatial distributed parameter systems or to IPDE for population systems are transformed using sophisticated methods to get reliable results in an acceptable computation time. That is; finite-difference schemes or finite-volume schemes possibly with dynamic regridding.

Finally it produces an input-file for the code generator.

☞ *Process Modeling Tool - ProMoT.* ProMoT, Trankle et al. (2000) and Trankle et al. (1997), is a graphical model editor for DIVA for building flowsheet models or new modules. The graphical editor use MDL, the object-oriented model definition language. Models can also be supplied directly by the user, written in MDL, to ProMoT.

The modelling and simulation environment DIVA focuses on chemical processes and plants. The strengths of DIVA lie in the development and implementation of the new models, through the highly supported building process in ProMoT, and the automated features of SyPProT. Examples of the use of DIVA are shown by Mangold et al. (2000).

gPROMS

gPROMS is a type of commercial software that is freely available for academic use. It can communicate with a wide range of other software packages, and it is easy to communicate with gPROMS using general communicating protocols. The modelling tool allows flowsheet simulation with a combination of lumped and distributed parameter units, Pantelides and Britt (1995).

Like all the software packages described in this section, gPROMS has an object-oriented modelling language with possible inheritance. A feature of the modelling language is the ability to incorporate sequences (operating procedures), common in all process plants, handling automated start-up and shut-down. The models are flowsheet models that are created in a hierarchic fashion with arbitrary depth. Symmetric and reversible, reversible and asymmetric, and irreversible discontinuities are handled.

An example of a distributed parameter, is the spatial position. The mathematical description of these distributed systems will produce PDAE or IPDAE, Oh and Pantelides (1996). These equations are transformed to DAEs through the MOL approach. Allowing for more advanced, irregular geometries, gPROMS has a link to Fluent, a commercial CFD simulator. The combined functionality is described by Bezzo, Macchietto, and Pantelides (2003).

In contrast to pure deterministic process modelling, some stochastic process modelling capabilities are included in gPROMS.

The flowsheets are built within the graphical flowsheeting editor (ModelBuilder), defining the topology of composite unit models.

gPROMS contains several solvers for DAE equations and linear and non-linear equations. DASOLV is an integrator for large, sparse systems of DAEs. DASOLV uses the BDF method. A direct solver is used for solving sparse linear equations (MA48 Duff and Reid (1996)).

gPROMS is capable of symbolic analysis of the equations. The analysis contains among other things, a check for well-posedness, index check and possibly index reduction.

As gPROMS is a commercial software it supports a vide variety of automated functionality. Being commercial also means less programming bugs than the other environments described in this section.

ABACUSS

ABACUSS II has state-of-the-art functionality. On the home page of ABACUSS II it is stated: 'ABACUSS II is the next generation open modelling environment and simulator'. The home page is at 'http://yoric.mit.edu/abacuss2/abacuss2.html' where the ABACUSS II binaries can be downloaded. It is also emphasised that the flexibility of the ABACUSS II implementation makes it easier to embed within another applications than other equation-oriented simulation tools available. The current design of ABACUSS II is based on years of experience with gPROMS and ABACUSS, Feehery and Barton (1996a).

ABACUSS II is integrated with DAEPACK, Tolsma and Barton (2000) and Tolsma and Barton (2004). DAEPACK is a set of software components for performing symbolic and numeric computations on general FORTRAN-90 models. Fortran-90 code representing a DAE model can be supplied as input and is translated to the internal symbolic representation. The entire DAE model can then be a combination of the supplied FORTRAN code, and the models written in the ABACUSS II modelling language. ABACUSS II translates the input files and passes the information to DAEPACK. The model language of ABACUSS II can be characterised with the same general terms as the ASCEND modelling language.

The integrator routine in DAEPACK is derived from DASSL, Petzold (1983), and uses a large-scale direct linear solver, Duff and Reid (1996). The integrator handles DAE index-1 systems. The symbolic components distinguish DAEPACK from other libraries for numerical calculation. These symbolic components can be used to automatically generate the information required for carrying out general numerical calculations efficiently, robustly, and correctly. That means that sparsity patterns are located, analytical derivative matrices are generated using AD technology and automatic location of discontinuities/state events, Park and Barton (1996). This allows for a discontinuity-locked model, that improves the integration.

Further the code checks if the model is structurally well posed, it checks for the index, and checks if the model has hidden constraints. High index systems are modified for solution using the method of dummy derivatives, Feehery and Barton (1996b), Mattsson and Söderlind (1993). The code handles the large sparse systems by decomposition to block lower triangular form (provided it is not irreducible), and solve the linear system as a sequence of smaller subsystems, Tolsma and Barton (1999).

The ability to include legacy models, available in FORTRAN-90 source code is one of the great advantages of the ABACUSS II/DAEPACK system. DAEPACK is designed to work with a very general FORTRAN-90 source that may contain sophisticated solution strategies and not simply sequences of assignments. For example, the model may compute molar volumes internally using the SRK EOS and a Newton-type iterative scheme, Tolsma, Clabaugh, and Barton (2002).

2.12 Modular integration

In contrast to the equation-oriented approach, the process model is partitioned in sub-domains or models comprising subsets of equations in the modular approach. Each subset can be solved with the same or different numerical integration algorithms (multi-method technique). If the same algorithm is utilised, different time steps can be used in the various sub-models, so-called multirating techniques. All the equations can also be solved simultaneously using the *simultaneous-modular* approach.

A procedural implementation and execution of the unit models is common for the modular simulation environments. The process equipment naturally gives the modulation, and every process unit is a module in the simulator. If the procedures associated with a unit module produce differentials for the state variables, all equations can be solved simultaneously. If that is not available, the differentials can be produced with a finite difference approximation. Otherwise the modules must integrate themselves in a pure sequential-modular approach.

In a pure sequential-modular simulator the modules are integrated with fixed input (and output), and give output properties. These are used to update the input to the connected modules. This method requires iteration in the outer integration, to converge and match the inputs and outputs of the connected modules.

According to Brosilow et al. (1985) sequential-modular integration has the following benefits over simultaneous integration of the entire equation set.

- The simulator can be more efficient because[1]:
  - Each sub-system uses an integration algorithm that is best suited to that sub-system.
  - The integration of the sub-systems can be solved in parallel.
  - The simulator only has modular error control
- The modular software is easier to maintain, because of the logical partitioning and independence of the modules

Another advantage of this integration method is the possibility that the handling of discontinuities can be done locally, and this can simplify the integration. It is also easy to front and connect modules created by others, without knowing the details of the module internal description.

For further details and information on the difference between the simultaneous modular and the sequential modular methods, see Hillestad and Hertzberg (1986), Laganier et al. (1993), Patterson and Tozsa (1980), Hlavacek (1977), and Brosilow et al. (1985). The sequential modular approach is abandoned in dynamic simulation, but has a strong position in steady state simulation.

A network solver, as described by Endrestøl et al. (1989), is believed to be the industrial standard of dynamic real-time simulators. An advantage of network integration method is the splitting of the integration into fast and slow dynamics. That is; the fast pressure dynamics can be solved using a fully implicit integration algorithm, and the slower compositional and energy dynamics can be solved using less computationally demanding explicit integration. The drawback of these approaches is the lack of error control, when splitting the integration.

The stability of the integrations is also difficult to predict because of the decoupling effect.

2.13 Thermodynamics

When describing the phase equilibrium and the physical properties of the hydrocarbon mixtures in the LNG plant, several methods have to be considered. In the list below, there are three main approaches to correlate equilibrium and properties.

[1]The term efficient, means the least computationally demanding integration

- Rigorous, equations of state (EOS)
- Simplified descriptions
- Table based

The thermodynamic method must solve the algebraic equilibrium equation with several sets of specifications. The most common are the PT, HP, SP, UV and SV. Because of the close relation between the UV- and SV-flash in constant volume systems, only one of these flashes is needed.

The enthalpy, entropy, Gibbs free energy and density must be described by thermodynamic relations. Usually the physical properties are expressed as functions of the temperature, pressure and composition. The partial differentials of the physical properties are often required, and should therefore be available.

Table-based thermodynamics consist of a look-up table (LUT) and a multidimensional interpolation scheme. Mapping LUT prior to simulation, the equilibrium and property calculations are explicit. The time consumption is predictable and small relative to the iterative rigorous thermodynamics. Therefore this is the standard approach for real-time operator training simulators.

The EOS is a large group of methods, which can be divided into three main groups.

- Cubic EOS:
  Soave-Redlich-Kwong, (SRK), and Peng-Robinson (PR)
- Principle of corresponding state
- Virial EOS

The most famous, simplest and most used methods in practice are the SRK, Soave (1972), and PR, Peng and Robinson (1976), EOS. These equations are called cubic because the pressure is a cubic function of the specific volume. They generally give a good description of the equilibrium in non-ionic mixtures.

In general the EOS approach gives the best accuracy and is useful over a large region in temperature and pressure. The methods are applicable for mixtures of components, and single components. But they are the most demanding in terms of computing time for the three main methods for describing the equilibrium.

There are several methods of simplified thermodynamics. The most obvious is to use simplified models like the ideal gas law, or other simple algebraic relations. One popular approach in dynamic simulation is local thermodynamic models, see Chimowitz and Lee (1985), Anderko, Coon, and Goldfarb (1994), and Hillestad et al. (1989). The EOS is simplified in small regions, to speed up the simulation.

The table-based thermodynamics can be experimental data, or the tabulation of a rigorous model. The possibility of including experimental data without correlating it, is useful in some cases.

There are two main considerations when choosing thermodynamic description. First the accuracy of the thermodynamic predictions. If the accuracy of the equilibrium and physical properties is poor, the output of the simulator will not be realistic. It is, therefore, necessary to define how accurate the thermodynamics needs to perform.

The second consideration is the demand for fast computation. If the simulator is required to perform better than real time, it may not be possible to use time-consuming rigorous methods.

Chapter 3

Modelling Framework

SEPTIC and TPLib were chosen to be the modelling framework. SEPTIC is described in Section 3.1 while TPLib is described in Section 3.2. The decision to use SEPTIC and TPLib resulted in implementing most of the code necessary to describe the process equipment and to solve their equations. An alternative approach would have been to use one of the tools described in Section 1.5. This approach had resulted in less code implementation and possibly produced a fast simulator with a qualitatively good dynamic description of the plant. SEPTIC and TPLib were believed to give a better foundation for future academic or industrial work within the field of dynamic simulation, getting hands-on experience with the implementation of solution methods for integration, non-linear and linear equation solutions. This choice also opens the possibility to compare the industrial approach using a network solver vs. the more fundamentally correct approach, conserving internal energy in constant volume process equipment.

The network solver is introduced in Section 3.3.

The modelling programming environment has been C++ and FORTRAN. The implementation has progressed in parallel on both the Win32 and the Linux platform. Visual C++ v6.0 (Microsoft) and Compaq Digital Visual FORTRAN v6.6 (Hewlett-Packard) have been the compilers of choice on the Win32 platform. The GNU compiler collection (gcc) v. 3.2.2 and the NAGWare FORTRAN 95 compiler Release 4.2(511) from The Numerical Algorithms Group Ltd. (NAG Ltd.) has been used on the Linux platform.

The reason for the mixed coding is simple. The preferred code for implementation is C++, and the modelling is performed within the main program SEPTIC. Mathematical function libraries available for free download on the Internet are usually written according to FORTRAN 77 standard. These codes need to be compiled and sometimes modified to suit the needs of the programmer. Also the thermodynamics library, TPLib, is implemented in FORTRAN.

When using mixed code there are some compiler conventions that must be handled. The most important is the mapping of memory. C/C++ maps matrix memory row-wise, while FORTRAN maps the memory column-wise. This difference was solved implementing a matrix template in C++ that emulated FORTRAN mapping by mapping the memory column-wise.

The integration of DAEs describing process equipment generally requires Jacobian information. The finite difference and analytical differential were considered to produce the Jacobian matrix. Automatic differentiation was not considered at the time, but could have been used after considerable preprocessing of the source code. The finite difference was used for a small plant section, but was found too slow when using rigorous thermodynamics. The Jacobian was therefore implemented by hand coding.

An analytical relation for thermodynamics differentials was derived to produce an analytical Jacobian matrix. The thermodynamics in general and the differentials are shown in Section 3.2.1.

3.1 Statoil Estimation and Prediction Tool for Identification and Control - SEPTIC

SEPTIC is Statoil owned software primarily used for simulation and MPC control. In this work SEPTIC is used for configuration through keyword-based ASCII-files and plotting state variables during the simulation.

The keyword configuration defines the units/models of the process to be simulated. The units are given an initial state, and sources and static dummy units are included to give boundary conditions for the simulation. The boundary condition is either fixed flow or fixed pressure. All boundary flows are also given constant composition and constant physical properties.

In SEPTIC the units/models, sub-models, are placed in the following three categories:

- Pressure models
- Flow models
- Controller models

The Pressure models are fixed volume units that conserve mass and energy. The Pressure models will therefore produce a pressure through the thermodynamic correlations, hence the name “Pressure models”. The Flow units/models have an algebraic equation describing the flow through the unit as a function of the pressure difference, and other physical properties. The Flow models will therefore produce a flow value, hence the name.

The function of the Controller units/models are obvious. The controllers (PID) are implemented as continuous controllers. See Appendix 6 for further details.

The layout of the configuration file is as follows. First the Pressure models are defined, and given labels. Next the Flow models are defined and given labels. The Flow units are connected to the Pressure models, by referring to the labels of the Pressure models. The directions of the flow are determined by the Flow models themselves, by looking at pressures in the connected Pressure models.

It is only possible to make connections between Pressure models and Flow models. One Pressure model can have an arbitrary number of Flow models connected. The data flow is managed by the main program, and the sub-models are given input data. The sub-models solve their own algebraic equations, and update properties and differentials.

The Controller models are given input/output/reference by referring to the Pressure, Flow and other Controller model labels. The variable name of the measured/output/reference is also given. The controller unit then has a reference variable for all its communication with the other units.

From the above description it is understood that the layout of the plant and the connection between the models are arbitrary, and do not need any form of hard coding.

The configuration file also set flags determining what integration method to use. The integration routine input and sampling time of the integration are also given. The process states are plotted only when the integration is sampled. Time events can only take place at a sample time.

The Graphical User Interface (GUI) has the appearance of a typical window application. An example screen shot is given in Figure 3.1.

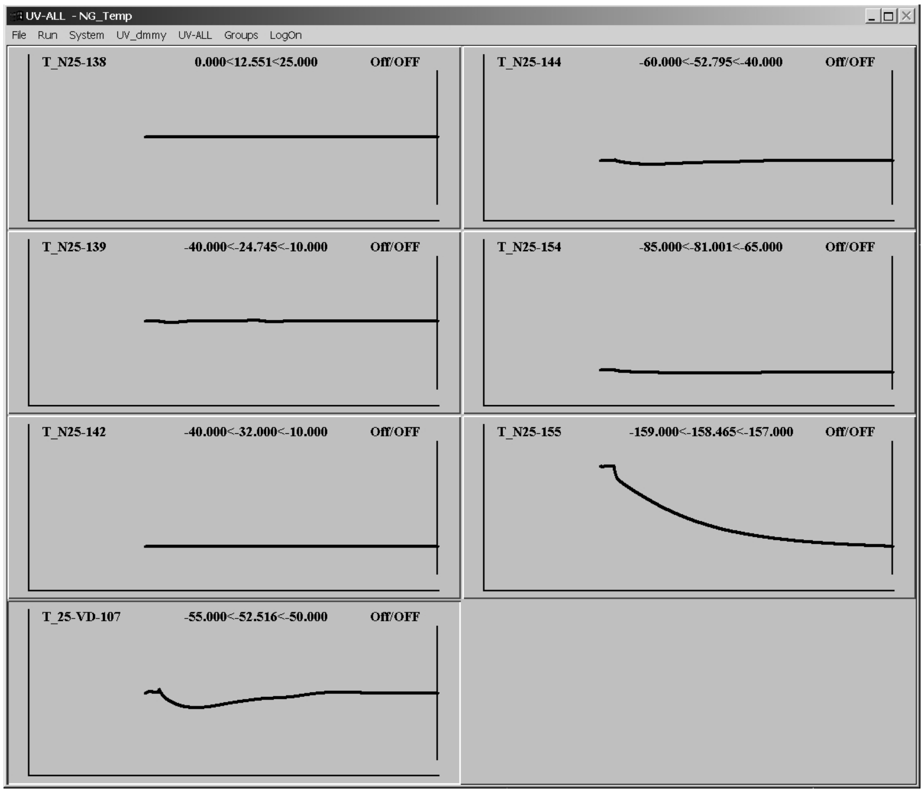

UV-ALL - NG_Temp
File Run System UV_dmmy UV-ALL Groups LogOn
T_N25-138 0.000<12.551<25.000 Off/OFF
T_N25-144 -60.000<-52.795<-40.000 Off/OFF
T_N25-139 -40.000<-24.745<-10.000 Off/OFF
T_N25-154 -85.000<-81.001<-65.000 Off/OFF
T_N25-142 -40.000<-32.000<-10.000 Off/OFF
T_N25-155 -159.000<-158.465<-157.000 Off/OFF
T_25-VD-107 -55.000<-52.516<-50.000 Off/OFF

Figure 3.1: Example of the SEPTIC interface

The GUI does 2D-plotting of states vs. time and vs. spatial position. The plots are defined on the configuration file or they can be defined interactively while running the application. All defined plots are available from the pull-down menus. The simulation can be started/stopped and stepped from the pull-down menus. It is also possible to introduce step changes in boundary conditions and states, both through the configuration file and the GUI.

Spatial plots, rather than time plots are preferred to describe the internal of the heat exchanger. Temperature, vapor fraction, density, etc. are then plotted from inlet to outlet for the heat exchanger. A example of this is shown in Figure 3.2.

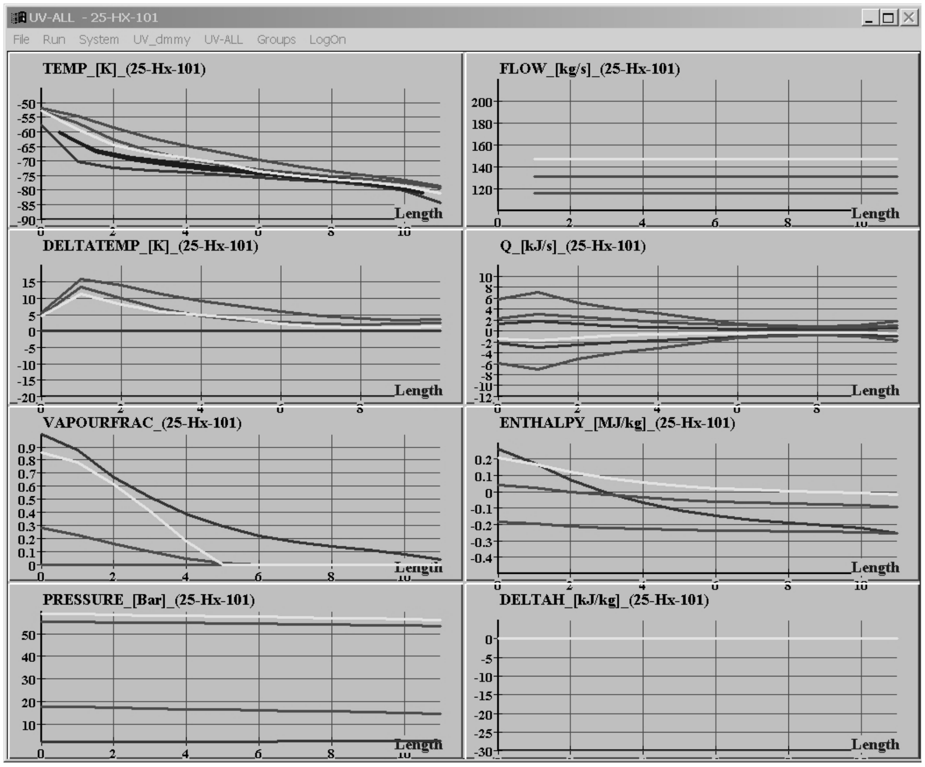

UV-ALL - 25-HX-101
File Run System UV_dmmy UV-ALL Groups LogOn
TEMP_[K]_(25-Hx-101)
FLOW_[kg/s]_(25-Hx-101)
DELTATEMP_[K]_(25-Hx-101)
Q_[kJ/s]_(25-Hx-101)
VAPOURFRAC_(25-Hx-101)
ENTHALPY_[MJ/kg]_(25-Hx-101)
PRESSURE_[Bar]_(25-Hx-101)
DELTAH_[kJ/kg]_(25-Hx-101)
Length

Figure 3.2: Example of the heat exchanger plots in SEPTIC

3.2 Thermodynamics

TPLib is a NTNU developed thermodynamic and physical property function library for single and two-phase mixtures. Both the SRK and the PR methods are implemented. (The package contains implementations of more thermodynamic methods, but they will not be considered in this thesis). The function interface for both methods are identical and switching between the two methods can be done on initiation. The default method and the method used in this work is the SRK EOS.

TPLib contains functions for some general flashes and the physical properties, enthalpy and density. Flash routines available in TPLib:

- TP - flash at constant temperature and pressure
- HP - flash at constant enthalpy and pressure
- SP - flash at constant entropy and pressure

The only flash missing, and needed by this work, is a flash where internal energy and volume, UV, is specified. The UV-flash is therefore implemented, using the TPLib Gibbs functions and physical property functions for enthalpy and density. It can be argued that an UP-flash (specified internal energy and pressure) should have been implemented. The UP-flash is not implemented.

The TPLib single phase physical property function calls provide analytical partial differentials of temperature, pressure and composition.

The TPLib is compiled as a dynamically linkable library, and is interfaced from SEPTIC.

3.2.1 Thermodynamic implementation

This work needed an UV-flash, and therefore an algorithm to solve the UV-flash problem is implemented. The implementation details are shown in Appendix A. Considering the fact that the flash is going to be used in a dynamic simulator, where good initial values always are available, a Newton-Raphson (NR) approach is chosen. Dimensionless Gibbs energy and dimensionless state functions for specific internal energy and specific volume describe the equilibrium.

The NR algorithm has local q-quadratic convergence, see definition in Equation B.19, but does not guarantee convergence from an arbitrary starting point. To ensure fast and global convergence the NR search must be combined with a line search along the descent NR search direction. The line search can only accept a step if the norm of the function value decreases. The line search of choice uses cubic backtracking, as described by Dennis Jr. and Schnabel (1996).

Another complicating fact is the constraints on the compositional variables, and the possibility of the phase changing. The constraints on the compositional variables can be handled simply resetting the initial NR search length. The step is reset in a manner that allows the composition to approach the constraint from the feasible region.

Possible phase change is handled by dropping or adding a phase. The dropping or adding of a phase is done when the algorithm experiences difficulties in converging, typically when searching for a solution near a phase boundary.

If it is not able to produce a solution, the solution strategy is changed to a nested loop iteration. This iteration is formulated in a manner that guarantees convergence, see Appendix A and Michelsen and Mollerup (1998).

When writing out the full Jacobian for the ODEs, the need for partial differentials of states and physical properties along equilibrium arises. The equilibrium system is a set of NC, NC+1 or NC+2 nonlinear implicit AEs, depending on the flash algorithm considered. NC is short for "Number of Components".

The partial differentials are needed with respect to the dynamic states, that is internal energy, U, and compositional holdup. To produce these differentials, the function describing the equilibrium is linearized in the dynamic state variables. The linearized equilibrium is then solved to give partial differentials for the internal equilibrium variables. The partial differentials of the various physical properties can then be found as linear combinations of these internal equilibrium partial differentials.

The principal equations for the linearization are given below, but for the detailed description, see Appendix A.3. The equilibrium is described in the following form:

$$\mathbf{F}(\mathbf{S}) = \mathbf{F}(\mathbf{X}(\mathbf{S}), \hat{\mathbf{S}}(\mathbf{S})) = \begin{bmatrix} g_1(\mathbf{X}(\mathbf{S}), \hat{\mathbf{S}}(\mathbf{S})) \\ \vdots \\ g_{NC}(\mathbf{X}(\mathbf{S}), \hat{\mathbf{S}}(\mathbf{S})) \\ c_1(\mathbf{X}(\mathbf{S}), \hat{\mathbf{S}}(\mathbf{S})) \\ c_2(\mathbf{X}(\mathbf{S}), \hat{\mathbf{S}}(\mathbf{S})) \end{bmatrix},$$

$$\mathbf{X} = \begin{bmatrix} N_{V,1} \\ \vdots \\ N_{V,NC} \\ lnT \\ lnP \end{bmatrix}, \quad \hat{\mathbf{S}} = \begin{bmatrix} z_1 \\ \vdots \\ z_{NC} \\ s_1 \\ s_2 \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} N^*_{z_1} \\ \vdots \\ N^*_{z_{NC}} \\ s_1 \\ s_2 \end{bmatrix}, \tag{3.1}$$

$$\mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_{NC} \end{bmatrix}, \quad \mathbf{N^*_z} = \begin{bmatrix} N^*_{z_1} \\ \vdots \\ N^*_{z_{NC}} \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$

c_i, are functions making sure the general flash specifications, s_1 and s_2, are satisfied. $N_{V,i}$ is the vapor mole number for component i and z_i is the overall composition. N_{z_i} are the overall mole numbers. The asterisk notation means normalized mole numbers. The definition is given in Equation 3.2. g_i are the dimensionless Gibbs free energy functions. The $\mathbf{S}$ vector is parameter specifications to the flash system.

$$\sum_{j=1}^{NC} N^*_{z_j} = 1, \text{ that is; } N^*_{z_j} = \frac{N_{z_j}}{\sum_{j=1}^{NC} N_{z_j}} \tag{3.2}$$

The asterisk superscript indicates that the molar content is a scaled value, and the vector elements sum to one. It is important to notice, that the scaling factor is treated as a constant. Otherwise, this would be the definition of composition, z_i.

The direct linearization of $\mathbf{F}$ around $\mathbf{X_0}, \mathbf{\hat{S}_0} = \mathbf{X}(\mathbf{S_0}), \mathbf{\hat{S}}(\mathbf{S_0})$ would yield Equation 3.3, using the chain rule of differentiation.

$$\begin{aligned}\mathbf{F}(\mathbf{S}) \approx& \mathbf{G}(\mathbf{S}) = \mathbf{F}(\mathbf{S_0}) + \nabla_\mathbf{S}\mathbf{F}^T_{\mathbf{S_0}}\Delta\mathbf{S} \\ \nabla_\mathbf{S}\mathbf{F}^T_{\mathbf{S_0}} =& \nabla_\mathbf{X}\mathbf{F}^T_{\mathbf{S_0}}\nabla_\mathbf{S}\mathbf{X}^T + \nabla_{\mathbf{\hat{S}}}\mathbf{F}^T_{\mathbf{S_0}}\nabla_\mathbf{S}\mathbf{\hat{S}}^T\end{aligned} \tag{3.3}$$

Where $\nabla_\mathbf{X}\mathbf{F}^T$ is recognized as the Jacobian matrix used in the internal solution of the flash problem. $\nabla_\mathbf{S}\mathbf{X}^T$ is the required differential solution matrix. The entries of both $\nabla_\mathbf{X}\mathbf{F}^T$ and $\nabla_{\mathbf{\hat{S}}}\mathbf{F}^T$ is shown in Section A.3.1. The $\nabla_\mathbf{S}\mathbf{\hat{S}}^T$ are given below in Equation 3.4.

$$\nabla_\mathbf{S}\mathbf{\hat{S}}^T = \begin{bmatrix} \nabla_{\mathbf{N^*_z}}\mathbf{z}^T & \mathbf{0} \\ \nabla_{\mathbf{N^*_z}}\mathbf{s}^T & \mathbf{I_2} \end{bmatrix} \tag{3.4}$$

It is seen from Equation 3.3, that if the first order Taylor expansion is performed in an equilibrium point, that $\nabla_\mathbf{S}\mathbf{F}^T_{\mathbf{S_0}}$ should be a zero matrix. This is shown in Equation 3.5 and Equation 3.6

$$\mathbf{G}(\mathbf{S}) = \mathbf{F}(\mathbf{S_0}) + \nabla_\mathbf{S}\mathbf{F}^T_{\mathbf{S_0}}\Delta\mathbf{S} = \nabla_\mathbf{S}\mathbf{F}^T_{\mathbf{S_0}}\Delta\mathbf{S} \tag{3.5}$$

Requiring the linear function to be zero, for all changes in $\mathbf{S}$.

$$\mathbf{G}(\mathbf{S}) = \nabla_\mathbf{S}\mathbf{F}^T_{\mathbf{S_0}}\Delta\mathbf{S} = \mathbf{0} \Rightarrow \nabla_\mathbf{S}\mathbf{F}^T_{\mathbf{S_0}} = \mathbf{0} \tag{3.6}$$

This gives the following relation for $\nabla_\mathbf{S}\mathbf{X}^T$:

$$\nabla_\mathbf{S}\mathbf{X}^T = -(\nabla_\mathbf{X}\mathbf{F}^T_{\mathbf{S_0}})^{-1}\nabla_{\mathbf{\hat{S}}}\mathbf{F}^T_{\mathbf{S_0}}\nabla_\mathbf{S}\mathbf{\hat{S}}^T \tag{3.7}$$

The computation of $\nabla_\mathbf{S}\mathbf{X}^T$ is possibly cheap, because the LU decomposition of $\nabla_\mathbf{X}\mathbf{F}^T_{\mathbf{S_0}}$ is available from the last NR iteration in the flash solver. Building the $\nabla_\mathbf{S}\mathbf{\hat{S}}^T$ requires no physical property data, and is therefore cheap. Building the $\nabla_{\mathbf{\hat{S}}}\mathbf{F}^T_{\mathbf{S_0}}$ might require the evaluation of one or two physical property functions in the equilibrium point. Most of the matrix input is known from the evaluation of the Gibbs functions in the flash algorithm. The physical property function calls are usually required anyway, and therefore they give no additional computation cost.

Knowing $\nabla_\mathbf{S}\mathbf{X}^T$, the two-phase, tp, differentials can be written out. The two-phase physical property will have the general form of Equation 3.8.

$$\psi_{tp}(\mathbf{S}) = \psi_{tp}(W(\mathbf{S}), \psi_V(\mathbf{X}(\mathbf{S})), \psi_L(\mathbf{X}(\mathbf{S}), \mathbf{\hat{S}}(\mathbf{S})) \tag{3.8}$$

The differentials therefore take the form of Equation 3.9.

$$\begin{aligned}\nabla_\mathbf{S}\psi_{tp} =& \left(\frac{\partial\psi_{tp}}{\partial w}\right)\nabla^T_\mathbf{X} w\nabla_\mathbf{S}\mathbf{X}^T + \left(\frac{\partial\psi_{tp}}{\partial\psi_V}\right)\nabla^T_\mathbf{X}\psi_V\nabla_\mathbf{S}\mathbf{X}^T \\ &+ \left(\frac{\partial\psi_{tp}}{\partial\psi_L}\right)\nabla^T_\mathbf{X}\psi_L\nabla_\mathbf{S}\mathbf{X}^T + \left(\frac{\partial\psi_{tp}}{\partial\psi_L}\right)\nabla^T_{\mathbf{\hat{S}}}\psi_L\nabla_\mathbf{S}\mathbf{\hat{S}}^T\end{aligned} \tag{3.9}$$

Summary - Thermodynamics

A rigorous approach was chosen to describe the equilibrium and physical properties. The SRK equation of state was used. The integration routines required Jacobian information. This information can be obtained either by numerical perturbation or by analytically writing out the differentials. To get an acceptable simulation time an analytical approach was chosen.

The equilibrium system is implicitly defined, and in order to produce partial differentials with respect to the dynamic states, the equilibrium was linearized. The linearized equilibrium is equated to zero, and the resulting equation system is solved for partial differentials of the internal flash variable vector, with respect to the dynamic state vector. It is shown that with an optimal implementation the evaluation of these partial differentials only required the back substitution of the flash system Jacobian matrix. In the present version of the simulator, the Jacobian matrix for the flash system is recalculated and LU decomposed before executing the back substitution. Finally, to produce the physical property differentials, the single phase functions are evaluated. Some of these property functions are also evaluated during the evaluation of the flash Jacobian matrix. Therefore there is some potential to reduce the simulation time.

The function library TPLib supplied routines for calculating fugacities, enthalpy, entropy and density. These functions could be flagged to return analytical partial differentials. They were used to implement the UV flash and the linearization for the UV and the PH flash system. The PH, PS and PT flash were not implemented, instead the flash functions originally in TPLib was used. To lower the simulation time further, these functions should be reimplemented with an algorithm similar to the algorithm used for the UV flash. Linearization for the PS and PT system should also be implemented.

The UV flash implementation was tailormade for dynamic simulation. The algorithm exploited the good starting values in an NR iteration. The solver of the nonlinear UV system was set up to enter a nested loop solver, see Section A.2.2, if not converging after doing 200 iterations. During the simulations shown in Section 7, the solver never entered the nested loop solver.

The equilibrium equation set in some areas is highly nonlinear, and the nonlinearity will restrict the step length of the simulator. Nevertheless, the linearization produces building blocks for the dynamic state variable Jacobian matrix. After making the physical property differentials available, the differentials of the unit models could be written out. The simulation results, see Section 7.2.6, show that the linearization of the thermodynamic equations produce a system Jacobian matrix of a quality good enough to be used with the freeware solvers DVODPK, DASPK and LSODES.

3.3 The network solver

The motivation for the network solver is the need to control the fast pressure dynamics. This can be done if the pressure, which is an algebraic state, is treated as a dynamic state and integrated implicitly. The means of obtaining the equations that are needed are shown below.

The general set of dynamic variables for a control volume is given in Equation 3.10. The dynamic equations for this system are one energy and NC component conservation equation.

$$\mathbf{S_{gen}} = \begin{bmatrix} U \\ \mathbf{N} \end{bmatrix} \tag{3.10}$$

The new set of dynamic variables is given in Equation 3.11. Here the common simplification to use enthalpy, H, instead of internal energy, U, is introduced. This simplification will result in a simpler equation set, but will give a poorer description at high pressures.

$$\mathbf{S^*_{gen}} = \begin{bmatrix} H \\ \mathbf{N} \end{bmatrix} \tag{3.11}$$

The differential equations for the control volumes of the system can be written out as follows:

$$\frac{dH}{dt} = \sum Wh \tag{3.12}$$

$$\frac{d\mathbf{N}}{dt} = \sum W\mathbf{z} \tag{3.13}$$

The summation is over all flows in and out of the control volume. Outlet flows are negative.

The constant volume of the system gives the following algebraic constraint:

$$N - V\rho(T, P, \mathbf{z}) = 0 \tag{3.14}$$

where N, the overall holdup, is given by:

$$N - \sum_{i=1}^{NC} N_i = 0 \tag{3.15}$$

Combining Equations 3.14 and 3.15 and differentiating:

$$V\frac{d\rho}{dt} - \sum_{i=1}^{NC} \frac{dN_i}{dt} = 0 \tag{3.16}$$

Combining Equations 3.16 and 3.13, remembering that the composition vector $\mathbf{z}$ sum to unity:

$$\frac{d\rho}{dt} = \frac{1}{V}\sum W \tag{3.17}$$

If the total differential for the density is written out, Equation 3.17 becomes 3.18.

$$\left(\frac{\partial\rho}{\partial P}\right)_{h,\mathbf{z}}\left(\frac{dP}{dt}\right)+\left(\frac{\partial\rho}{\partial h}\right)_{P,\mathbf{z}}\left(\frac{dh}{dt}\right)+(\nabla_{\mathbf{z}}\rho)^T\left(\frac{d\mathbf{z}}{dt}\right)=\frac{1}{V}\sum W \tag{3.18}$$

Splitting the integration, first integrating the pressure implicitly, then integrating enthalpy and composition gives the network solver. The pressure integration is executed assuming $(\partial\rho/\partial P)_{h,\mathbf{z}}$ constant, and the flow to be pressure dependent functions. The integration is therefore only partly implicit in pressure, and is executed at constant enthalpy and composition. The control volume pressure is described using Equation 3.19.

$$\left(\frac{\partial\rho}{\partial P}\right)_{h,\mathbf{z}}\left(\frac{dP}{dt}\right)=\frac{1}{V}\sum W \tag{3.19}$$

The second integration must then be performed at constant pressure and constant flow. Since, in general these dynamics are slow compared to the pressure flow dynamics, the integration is explicit. It is obvious that the integration of Equation 3.13 will not produce an overall hold-up satisfying Equation 3.14.

The network solver approach is often used in training simulators in the petroleum industry, where large models are required to run faster than real time. Endrestøl et al. (1989) describe the development and use of a network solver model.

3.3.1 Implementation of the network solver

The pressure states in the nodes and the flash tanks need to be solved with some implicitness to have stability for large time steps. To avoid perturbations, and slow simulation, the differentials for the system are generated analytically. The differentials are generated by the unit models (node and flash), and a pointer interface from the ODE solver collects the data, and maps the information into the needed matrices. The needed differentials are derived from a simple example, and then generalized.

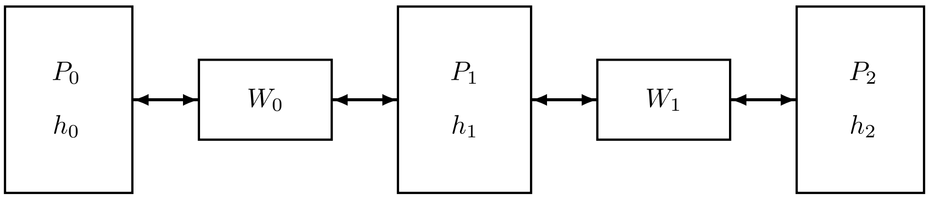


Figure 3.3: Reference system

The flow descriptions are as follows (refer to Figure 3.3 for simplicity):

$$W_0 = W_0(P_0, P_1) \tag{3.20}$$

The total differentials for the flow then become:

$$dW_0 = \left(\frac{\partial W_0}{\partial P_0}\right)_{P_1} dP_0 + \left(\frac{\partial W_0}{\partial P_1}\right)_{P_0} dP_1 \tag{3.21}$$

Linearized flow becomes:

$$W_0 = W_{0,0} + \left(\frac{\partial W_0}{\partial P_0}\right)_{P_1} dP_0 + \left(\frac{\partial W_0}{\partial P_1}\right)_{P_0} dP_1 \tag{3.22}$$

The extra 0 subscript indicates initial conditions.

The mass balance for the node with the subscript 1, is as follows. (Use subscript only on the flow and later the pressure since the rest is obvious):

$$V\left(\frac{d\rho}{dt}\right) = W_0 + W_1 \tag{3.23}$$

Make some definitions for readability, (NB! These are strictly not simple time constants):

$$Def: \qquad \tau_P = V\left(\frac{\partial \rho}{\partial P}\right)_h \tag{3.24}$$

Using Equation 3.24 in combination with Equation 3.23 the following equation emerges:

$$\tau_{P_1}\left(\frac{dP_1}{dt}\right) = W_0 + W_1 \tag{3.25}$$

Inserting the linearisation of W_0 and W_1 then gives:

$$\begin{aligned}\tau_{P_1}\left(\frac{dP_1}{dt}\right) =& W_{0,0} + \left(\frac{\partial W_0}{\partial P_0}\right)_{P_1}\left(\frac{dP_0}{dt}\right)\Delta t + \left(\frac{\partial W_0}{\partial P_1}\right)_{P_0}\left(\frac{dP_1}{dt}\right)\Delta t \\ &+ W_{1,0} + \left(\frac{\partial W_1}{\partial P_1}\right)_{P_0}\left(\frac{dP_1}{dt}\right)\Delta t + \left(\frac{\partial W_1}{\partial P_2}\right)_{P_1}\left(\frac{dP_2}{dt}\right)\Delta t\end{aligned} \tag{3.26}$$

Collecting terms:

$$\begin{aligned} RHS: &\quad W_{0,0} + W_{1,0} \\ LHS: &\quad \left(\tau_{P_1} - \left(\frac{\partial W_0}{\partial P_1}\right)_{P_0}\Delta t - \left(\frac{\partial W_1}{\partial P_1}\right)_{P_0}\Delta t\right)\left(\frac{dP_1}{dt}\right) \\ &- \left(\frac{\partial W_0}{\partial P_0}\right)_{P_1}\Delta t\left(\frac{dP_0}{dt}\right) - \left(\frac{\partial W_1}{\partial P_2}\right)_{P_1}\Delta t\left(\frac{dP_2}{dt}\right)\end{aligned} \tag{3.27}$$

Generalising:

$$\begin{aligned}&\left(\tau_{P_j} - \Delta t\sum_k\left(\frac{\partial W_k}{\partial P_j}\right)_{P_{i,i\neq j}}\right)\left(\frac{dP_j}{dt}\right) \\ &\qquad -\Delta t\sum_k\left(\frac{\partial W_k}{\partial P_{ik}}\right)_{\dots}\left(\frac{dP_{ik}}{dt}\right) = \sum W_{k,0}\end{aligned} \tag{3.28}$$

Dividing with τ_{P_j}

$$\begin{aligned}\left(1 - \frac{\Delta t}{\tau_{P_j}}\sum_k \left(\frac{\partial W_k}{\partial P_j}\right)_{P_{i,i\neq j}}\right)\left(\frac{dP_j}{dt}\right) \\ -\frac{\Delta t}{\tau_{P_j}}\sum_k \left(\frac{\partial W_k}{\partial P_{ik}}\right)_{\dots}\left(\frac{dP_{ik}}{dt}\right) = \frac{1}{\tau_{P_j}}\sum W_{k,0}\end{aligned} \tag{3.29}$$

The Jacobian for the pressure differentials is given in Equation 3.30.

$$\mathbf{J_P} = \nabla\left(\frac{d\mathbf{P}}{dt}\right)^T \tag{3.30}$$

Introducing the pressure Jacobian in Equation 3.29 the equation takes the form of Equation 3.31.

$$\left(\mathbf{I} - \Delta t\mathbf{J_P}\right)\left(\frac{d\mathbf{P}}{dt}\right) = \left(\frac{d\mathbf{P}}{dt}\right)_0 \tag{3.31}$$

Using Equation 3.31 directly to predict the pressure states at $t_0 + \Delta t$, is equivalent to the 1-stage Rosenbrock method or a first iteration in a fully implicit Euler formulation.

The fully implicit Euler scheme has the following form:

$$\mathbf{P_{k+1}} = \mathbf{P_k} + \Delta t\left(\frac{d\mathbf{P_{k+1}}}{dt}\right) \tag{3.32}$$

This implicit equation is solved as shown in Section B.2. The network solver can progress using both a 1-stage Rosenbrock approach, and a fully implicit Euler approach.

The entries of this (partly) analytical Jacobian, $\mathbf{J_P}$, are given together with the detailed unit modelling in Chapters 4 and 5.

If flow or pressure is controlled by manipulating a flow, these controllers are included and solved as part of the pressure-flow system. The reason for doing this is the relatively small integral times and high proportional terms used in these controllers. Including them will give better performance of the solver.

The integral terms of these controllers are therefore included in the pressure-flow system and integrated implicitly. In this thesis, the Jacobian entries depending on these integral terms are calculated using perturbations.

3.4 Integration

This section will describe the two methods of integration used for solving the entire equation set simultaneously. The integration progresses by substitution of the algebraic equations into the ODEs. The AEs, mainly thermodynamic relations, are solved directly or using an iterative approach. Generally the iterative solvers are using a Newton approach with a line search. But some of the thermodynamics equations have tailor-made solution scheme.

Solving by substitution of the AEs into the ODEs result in a nested iteration on every time step. To avoid this, the implicit AEs could have been solved simultaneously with the ODEs. This would give a faster integration. Having almost all the required differentials available this approach where also possible to implement. The reason for choosing the approach involving the substitution of the AEs is its simplicity.

The full system Jacobian is analytically calculated. The different unit models calculate the Jacobian entries locally. The equations describing the partial differentials in the Jacobian are given in the unit modelling section of Section 4 and 5.

Simplified integration

The 1-stage Rosenbrock can be used for both the overall equation system, the network solver and for solving the heat exchanger as a sub-model in the network solver. Using the network solver the sub-models, other than the heat exchanger, are integrated using a simple forward Euler.

The general s-stage Rosenbrock method was given in Equation 2.8. Setting s=1 gives Equation 3.33.

$$\begin{aligned} k_1 &= hf(y_0) + hJ\gamma_{11}k_1 \\ y_1 &= y_0 + b_1 k_1 \end{aligned} \tag{3.33}$$

Writing out the equation gives Equation 3.34

$$y_1 = y_0 + b_1(I - hJ\gamma_{11})^{-1} hf(y_0) \tag{3.34}$$

According to Hairer and Wanner (1996) the determining coefficients γ_{ij} and b_j both should have the value 1 for the 1-stage Rosenbrock method.

$$y_1 = y_0 + (I - hJ)^{-1} hf(y_0) \tag{3.35}$$

As can be seen in Appendix B the 1-stage Rosenbrock method is equivalent to using a first order Taylor expansion for the derivative, that is; $f(y_1) \approx f(y_0) + J(y_1 - y_0)$.

It is further shown in Appendix B that the integration formula Equation 3.35 is A-stable.

Integration of the full equation set

To solve the overall equation system, an integrator must be chosen. The integrator must fulfill the following criteria:

- A-stability
- Sparse handling of the algebraic equation set solved in the Newton-Raphson iterations
- Need to utilize the cheap Jacobian information

Since the algebraic equations are substituted into the ODEs, the solver is not required to handle algebraic equations.

For simple and fast integration without error control the same method as above is used, a 1-stage Rosenbrock method.

From the review in Section 2 it is concluded that a BDF method should be used. Searching the web for free integrators, the following three solvers were encountered: LSODES, DVODPK and DASPK.

LSODES, Hindmarsh and Sherman (1987) and Hindmarsh (1983), and DVODPK, Hindmarsh, Brown, and Byrne (2002), can solve first order ODE systems with either an implicit Adams method, or a BDF method. LSODES uses a sparse LU linear solver while DVODPK uses a Krylov linear solver.

The DASPK, Petzold et al. (2000), Petzold (1983), Brown, Hindmarsh, and Petzold (1994) and Brown, Hindmarsh, and Petzold (1998), solver uses the BDF method of orders one through five to solve a system of DAEs. The linear system solver is either a dense, banded or a preconditioned Krylov iterative method. The Krylov method is the Generalized Minimum Residual (GMRES) method, in either complete or incomplete form, and with scaling and preconditioning. The method is implemented in an algorithm called SPIGMR (Scaled Preconditioned Incomplete GMRES).

DVODPK also uses the SPIGMR algorithm.

Chapter 4

Modelling of Pressure Units

This section will give an introduction to the pressure unit modelling, before stating the equations used in the implementation. Section 4.1 includes some definitions which are added to ease the indexing in the equations.

the following Pressure unit category models are required to describe a general LNG system.

- Tank
  - Node (Volume representation)
  - Flash (Separation)
- Column

In general the tank units and the stages of the column are modelled as control volumes (CVs) with multiple inlets and outlets. The outlets can be homogeneous flow (Node) or one phase flow (Flash, Column). The general conservation laws for the CVs are given in Equations 4.1 and 4.2. The details are given later in this chapter.

The mass conservation of the different components is given in Equation 4.1.

$$\left(\frac{dN_{z_i}}{dt}\right) = \sum_{pi \in IN_s} W_{pi} z_{pi,i} - \sum_{po \in OUT_s} W_{po} z_i \tag{4.1}$$

The energy conservation is given in Equation 4.2.

$$\left(\frac{dU}{dt}\right) = \sum_{pi \in IN_s} W_{pi} h_{pi} - \sum_{po \in OUT_s} W_{po} h \tag{4.2}$$

Here, W is the flow rate into/out of the CV. The index pi refers to inlet ports, and the index set IN_s is the set of all inlet ports. In the same way, po is the outlet ports, and OUT_s is the set of all outlet ports. $z_{pi,i}$ is the overall composition in inlet port stream pi of component i. z_i is the overall internal node composition. The same rules apply to the specific enthalpy h. That is; the nodes are described as mixing tanks.

Two parallel implementations exist, one where enthalpy, H, is conserved, and one where internal energy, U, is conserved. When enthalpy is conserved, the left hand side entry in Equation 4.2, simply is replaced by dH/dt. The volume, V, of the CV is always treated as constant.

Using the relation $U = H - PV$ in combination with Equation 4.2, and at the same time applying the constant volume constraint produces Equation 4.3.

$$\left(\frac{dH}{dt}\right) = \sum_{pi \in IN_s} W_{pi} h_{pi} - \sum_{po \in OUT_s} W_{po} h + V \left(\frac{dP}{dt}\right) \tag{4.3}$$

The implementation conserving enthalpy will neglect the contribution from the last term in Equation 4.3. This simplification is acceptable for liquids where the molar volume, volume/mole, is small. For gases this simplification can be crude. Both approaches will produce the same steady state.

This implies that the internal energy approach is fundamentally correct, while the conservation of enthalpy only is an approximation of the internal energy approach. That is; the internal energy conservation is qualitatively better than the enthalpy conservation.

When using the network approach, the compositional conservation in the nodes, can be switched off (default). Then the node copies the inlet composition, and this composition is copied downstream at the next integration step. The composition is therefore delayed by one time step. The network solver also uses the pressure as a dynamic variable, and a special integration routine for the flow pressure network. This was explained in more detail in Section 3.3.1.

It was the intention of this work to include a column model. Several simplified approaches were implemented, but none performed on the high-pressure columns in the LNG plant. With this knowledge the columns should have been implemented as a set of separations units, determining the liquid flow with a weir equation. The lack of the column model is not a simplification, but a consequence of the time limit on this work.

The Pressure units all have a mode flag, with the possible values STATIC or DYNAMIC. If the mode flag is STATIC, the Pressure unit will do nothing, but supply a constant pressure, temperature, composition and enthalpy for the connecting flow units. If the mode is DYNAMIC, the conserved properties of the unit will be added to the set of ODE states, and described dynamically. The STATIC option is used only for boundary conditions.

The nodes represent pipes and volumes in the process, and are treated as mixing volumes. Fronts traveling through the pipes are therefore approximated with first order dynamics. The outflow is assumed to be a homogeneous. The flash tanks, are mixing tanks and phase splitters. Equilibrium thermodynamics are used in this model and in all other units.

4.1 Definitions

The units are distinguished in flow and pressure models, that are respectively; WM and PM. The pressure models can only be connected to the flow models, and the flow models are only connected to the pressure models.

All configured flow model units, **WM**, all configured pressure model units, **PM**, all configured controller model units and their connections define the process flowsheet.

This section introduce some definitions, that will make the equation writing, and the comprehension of the equations easier.

4.1.1 Ports to pressure models

In the general case a pressure model has more than one connection point, this connection point is called a port. An example is given in Figure 4.1. The notation will refer to this port, and some sets will be defined to simplify the summation indexes in the equations.

The situation where there is only one port will be distinguished from a situation with multiple ports.

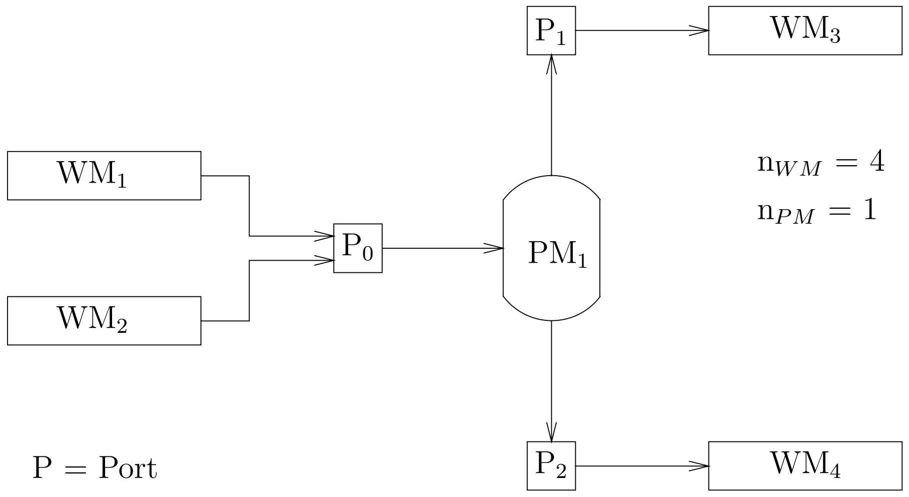


Figure 4.1: Example of a pressure model, PM_1, with three ports

Start by defining the sets of flow and pressure models for a given configuration, and their index sets. The number of all flow models are n_{WM}, and the flow models are numbered from 1 to n_{WM}. For the pressure models the number of units are n_{PM}, and they are numbered from 1 to n_{PM}. This defines the following index sets:

$$IS_{WM} = \{s : s \in \mathcal{N} \mid s \neq 0, s \leq n_{WM}\} \tag{4.4}$$

$$IS_{PM} = \{s : s \in \mathcal{N} \mid s \neq 0, s \leq n_{PM}\} \tag{4.5}$$

Where, $\mathcal{N}$ is the set of natural numbers, defined in Equation 4.6.

$$\mathcal{N} = \{0, 1, 2, ...\} \tag{4.6}$$

All flow models, **WM**, are then given by Equation 4.7. All pressure models, **PM**, are then given by Equation 4.8.

$$\mathbf{WM} = \{\text{All flow models}\} = \bigcup_{j \in IS_{WM}} WM_j \tag{4.7}$$

$$\mathbf{PM} = \{\text{All pressure models}\} = \bigcup_{s \in IS_{PM}} PM_s \tag{4.8}$$

The set of all flow models, WM$_j$, connected to port i of pressure model, PM$_s$ is defined through Equation 4.9.

$$S_{WM,PM_{s,i}} = \{WM_j : WM_j \in \mathbf{WM} \mid WM_j \;\; \text{connected to} \;\; PM_{s,i}\} \tag{4.9}$$

The set of flow models, $IN_{s,i}$, flowing into pressure model PM_s through port i, is then given by Equation 4.10.

$$\begin{aligned} IN_{s,i} =& \{WM_j : WM_j \in S_{WM,PM_{s,i}} \mid \\ & W_{WM_j} \;\; \text{flowing into} \;\; PM_{s,i}\} \end{aligned} \tag{4.10}$$

Here the flow through flow model WM_j is W_{WM_j}. When only one port is associated with the pressure model, the port index, i, is dropped. The set of flow models, $OUT_{s,i}$, with flow out of the pressure model port i, is then given by Equation 4.11.

$$OUT_{s,i} = S_{WM,PM_{s,i}} - IN_{s,i} \tag{4.11}$$

The zero-flowing flow models are part of the OUT set. They will not be important in this thesis.

For the example above, given in Figure 4.1, the different sets for the port subscripted with zero are as follows (positive flow is in the direction of the arrow):

$$\begin{aligned} S_{WM,PM_{s=1,i=0}} &= \{WM_1, WM_2\} \\ IN_{1,0} &= S_{WM,PM_{1,0}} \\ OUT_{1,0} &= \emptyset \end{aligned}$$

4.1.2 Composition

The naming rules for the composition and the holdup variables are given in Table 4.1.

Table 4.1: Naming of composition and holdup variables

Variable	Description	Unit
z_i	Overall composition of component i	kmol/kmol
$\mathbf{z}$	Vector of overall composition	kmol/kmol
y_i	Composition of component i in vapor phase	kmol/kmol
$\mathbf{y}$	Vector of vapor composition	kmol/kmol
x_i	Composition of component i in liquid phase	kmol/kmol
$\mathbf{x}$	Vector of liquid composition	kmol/kmol
$N_i = N_{z_i}$	Overall holdup of component i	kmol
$\mathbf{N} = \mathbf{N_z}$	Overall holdup vector	kmol
$N_{V,i}$	Holdup of component i in the vapor phase	kmol
$\mathbf{N_V}$	Holdup vector in vapor phase	kmol
$N_{L,i}$	Holdup of component i in vapor phase	kmol
$\mathbf{N_L}$	Holdup vector in liquid phase	kmol

To make the partial differentials easier to write, some set notation for the composition index is defined. We have by definition, NC components. The set of all components, S_{NC}, are then given Equation 4.12.

$$S_{NC} = \{1, 2, ..., NC\} \tag{4.12}$$

The following situation is quite common:

$$\left(\frac{\partial z_i}{\partial N_{z_j}}\right)_{N_{z_k}}, k \in S_{NC}, k \neq j$$

To cope with this some new notation is introduced:

$$S_{NC,j} = S_{NC} - j \tag{4.13}$$

Making the above example:

$$\left(\frac{\partial z_i}{\partial N_{z_j}}\right)_{N_{z_k}}, k \in S_{NC,j}$$

When the index is used explicitly, like in the following example another definition is needed.

$$g = g(N_i), \quad i \in S_{NC}, \quad \begin{bmatrix} \left(\frac{\partial g}{\partial N_1}\right)_{N_j} & \left(\frac{\partial g}{\partial N_2}\right)_{N_j} & \cdots & \left(\frac{\partial g}{\partial N_{NC}}\right)_{N_j} \end{bmatrix}, j \in ?$$

The index set $S_{NC,p}$ is therefore introduced in Equation 4.14.

$$S_{NC,p} = S_{NC} - p, \text{ for } p = 1, p = 2, ..., p = NC \tag{4.14}$$

The example above then look like the following:

$$g = g(N_i), \quad i \in S_{NC}, \quad \begin{bmatrix} \left(\frac{\partial g}{\partial N_1}\right)_{N_j} & \left(\frac{\partial g}{\partial N_2}\right)_{N_j} & \cdots & \left(\frac{\partial g}{\partial N_{NC}}\right)_{N_j} \end{bmatrix}, j \in S_{NC,p}$$

Another shortcut notation will be used for the divergence of vector $\mathbf{N}$. Notice the subscripting in Equation 4.15.

$$\nabla_{\mathbf{N}} = \begin{bmatrix} \left(\frac{\partial}{\partial N_1}\right)_{N_j,others} & \left(\frac{\partial}{\partial N_2}\right)_{N_j,others} & \cdots & \left(\frac{\partial}{\partial N_{NC}}\right)_{N_j,others} \end{bmatrix}^T, \quad j \in S_{NC,p} \tag{4.15}$$

In the general case, the functions will depend on other variables as well as the composition. In that case the operator will not have a subscript, and will mean partial differentials in the entire variable vector. The typical situation, "others" will mean temperature and pressure.

The overall holdup is defined in Equation 4.16.

$$N_{tot} = N = N_z = \sum_{j=1}^{NC} N_{z_j} \tag{4.16}$$

The overall holdup in the liquid and vapor phases is written in the same manner, but they use subscripts L and V, respectively.

4.2 Tank and pipes - Internal energy

4.2.1 Pipe - Node

The nodes represent the volume in the piping, and should therefore be described as a delay function. This is not practical due to the infinite gradients it generates. The nodes are therefore described as mixing tanks. They will dampen the compositional dynamics. The Node only has one port, and the port indexing for the pressure model will be dropped.

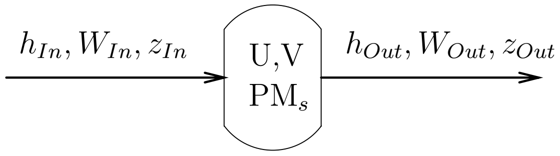


Figure 4.2: Node

Mass conservation

The mass conservation of the different components, is in the form of Equation 4.17.

$$\left(\frac{dN_{z_i}}{dt}\right) = \sum_{pi \in IN_s} W_{pi} z_{pi,i} - \sum_{po \in OUT_s} W_{po} z_i \tag{4.17}$$

Energy conservation

When the node variables are integrated without the analytical Jacobian, the conservation equation has the form of Equation 4.18.

$$\left(\frac{dU}{dt}\right) = \sum_{pi \in IN_s} W_{pi} h_{pi} - \sum_{po \in OUT_s} W_{po} h \tag{4.18}$$

Using the specific internal energy as a dynamic variable simplifies the differentiation for the Jacobian elements, and is therefore chosen. Equations 4.18 and 4.19 combine to Equation 4.20.

$$u = \frac{U}{N_{tot}} \tag{4.19}$$

$$\left(\frac{du}{dt}\right) = \frac{1}{N_{tot}}\left(\sum_{pi \in IN_s} W_{pi}(h_{pi} - u) - \sum_{po \in OUT_s} W_{po}(h - u)\right) \tag{4.20}$$

Analytical derivatives

First we need to write out the equations for the needed differentials. The thermodynamic relations are given first. The specific internal energy is given by Equation 4.21. The density is given by Equation 4.22.

$$u = h - Pv, \quad du = dh - vdP - Pdv \tag{4.21}$$

$$\rho(T, P, N_z) = \frac{1}{V}\sum_{j=1}^{NC} N_{z_j}, \quad v = \frac{1}{\rho} = \frac{V}{N_{tot}} \tag{4.22}$$

The differential for the internal energy, with respect to state holdup variables, is given in Equation 4.23.

$$\underbrace{\left(\frac{\partial u}{\partial N_{z_i}}\right)_{u,N_{z_j}}}_{=0} = \left(\frac{\partial h}{\partial N_{z_i}}\right)_{u,N_{z_j}} - v\left(\frac{\partial P}{\partial N_{z_i}}\right)_{u,N_{z_j}} + \frac{vP}{N_{tot}}, \quad j \in S_{NC,i} \tag{4.23}$$

The first RHS term is written out in Equation 4.24. The subscript is simplified.

$$\left(\frac{\partial h}{\partial N_{z_i}}\right)_u = \left(\frac{\partial h}{\partial T}\right)\left(\frac{\partial T}{\partial N_{z_i}}\right) + \left(\frac{\partial h}{\partial P}\right)\left(\frac{\partial P}{\partial N_{z_i}}\right) + \left(\frac{\partial h}{\partial N_{z_i}}\right) \tag{4.24}$$

The differential for the density, with respect to state holdup variables, is given in Equation 4.23.

$$\left(\frac{\partial \rho}{\partial N_{z_i}}\right)_u = \frac{1}{V} = \left(\frac{\partial \rho}{\partial T}\right)\left(\frac{\partial T}{\partial N_{z_i}}\right) + \left(\frac{\partial \rho}{\partial P}\right)\left(\frac{\partial P}{\partial N_{z_i}}\right) + \left(\frac{\partial \rho}{\partial N_{z_i}}\right) \tag{4.25}$$

Solving Equations 4.23 and 4.25 the change in T and P is given at constant u.

For the specific internal energy Equation 4.26 is valid.

$$\left(\frac{\partial u}{\partial u}\right) = \left(\frac{\partial h}{\partial u}\right) - v\left(\frac{\partial P}{\partial u}\right) = 1 \tag{4.26}$$

Where:

$$\left(\frac{\partial h}{\partial u}\right) = \left(\frac{\partial h}{\partial T}\right)\left(\frac{\partial T}{\partial u}\right) + \left(\frac{\partial h}{\partial P}\right)\left(\frac{\partial P}{\partial u}\right) \tag{4.27}$$

Differentiate the density with respect to internal energy in Equation 4.28.

$$\left(\frac{\partial \rho}{\partial u}\right) = \left(\frac{\partial \rho}{\partial T}\right)\left(\frac{\partial T}{\partial u}\right) + \left(\frac{\partial \rho}{\partial P}\right)\left(\frac{\partial P}{\partial u}\right) = 0 \tag{4.28}$$

Solving Equations 4.26 and 4.28 the change in T and P is given at constant composition.

Using Equations 4.23 through 4.28, all the enthalpy differentials can be written out. That is;

$$\left[\begin{array}{cccc} \left(\frac{\partial h}{\partial u}\right) & \left(\frac{\partial h}{\partial N_{z_1}}\right) & \cdots & \left(\frac{\partial h}{\partial N_{z_{NC}}}\right) \end{array}\right]$$

The density differentials are given by Equation 4.29.

$$\left[\begin{array}{cccc} \left(\frac{\partial \rho}{\partial u}\right) & \left(\frac{\partial \rho}{\partial N_{z_1}}\right) & \cdots & \left(\frac{\partial \rho}{\partial N_{z_{NC}}}\right) \end{array}\right] = \left[\begin{array}{cccc} 0 & \frac{1}{V} & \cdots & \frac{1}{V} \end{array}\right] \tag{4.29}$$

A 2x2 matrix system with NC+2 right hand sides need to be solved. The system abstraction is given in Equation 4.30.

$$\mathbf{MX} = \mathbf{B} \tag{4.30}$$

The M matrix is given by Equation 4.31.

$$\mathbf{M} = \left[\begin{array}{cc} \left(\frac{\partial h}{\partial T}\right)_{P,N_{z_i}} & \left(\frac{\partial h}{\partial P}\right)_{T,N_{z_i}} - v \\ \left(\frac{\partial \rho}{\partial T}\right)_{P,N_{z_i}} & \left(\frac{\partial \rho}{\partial P}\right)_{T,N_{z_i}} \end{array}\right] \tag{4.31}$$

The B matrix is given by Equation 4.32.

$$\mathbf{B} = \left[\begin{array}{cccc} 1 & -\frac{vP}{N_{tot}} - \left(\frac{\partial h}{\partial N_{z_1}}\right)_{P,T,N_{z_j}} & \cdots & -\frac{vP}{N_{tot}} - \left(\frac{\partial h}{\partial N_{z_{NC}}}\right)_{P,T,N_{z_j}} \\ 0 & \frac{1}{V} - \left(\frac{\partial \rho}{\partial N_{z_1}}\right)_{P,T,N_{z_j}} & \cdots & \frac{1}{V} - \left(\frac{\partial \rho}{\partial N_{z_{NC}}}\right)_{P,T,N_{z_j}} \end{array}\right] \tag{4.32}$$

The unknown X matrix is given by Equation 4.33.

$$\mathbf{X} = \left[\begin{array}{cccc} \left(\frac{\partial T}{\partial u}\right)_{N_{z_j}} & \left(\frac{\partial T}{\partial N_{z_1}}\right)_{u,N_{z_j}} & \cdots & \left(\frac{\partial T}{\partial N_{z_{NC}}}\right)_{u,N_{z_j}} \\ \left(\frac{\partial P}{\partial u}\right)_{N_{z_j}} & \left(\frac{\partial P}{\partial N_{z_1}}\right)_{u,N_{z_j}} & \cdots & \left(\frac{\partial P}{\partial N_{z_{NC}}}\right)_{u,N_{z_j}} \end{array}\right] \tag{4.33}$$

The conservation equation will be written out for the example in Figure 4.3, but the relations will in the general case be a sum over the respective IN and OUT sets.

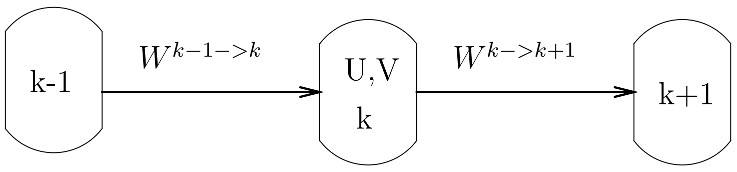


Figure 4.3: Node index reference for Node derivatives

- The composition

The relations between the mole numbers and the composition vector.

$$z_i = \frac{N_{z_i}}{\sum\limits_{j=1}^{NC} N_{z_j}} \tag{4.34}$$

$$\left(\frac{\partial z_i}{\partial N_{z_j}}\right)_{N_{z_l}} = \frac{N_{tot} - \left(\frac{\partial N_{z_i}}{\partial N_{z_j}}\right)_{N_{z_l}}}{N_{tot}^2}, \quad l \in S_{NC,j} \tag{4.35}$$

Restate the equation:

$$\left(\frac{dN_{z_i,k}}{dt}\right) = W^{k-1->k} z_{i,k-1} - W^{k->k+1} z_{i,k} \tag{4.36}$$

$$\begin{aligned}\left(\frac{\partial\left(\frac{dN_{z_i,k}}{dt}\right)}{\partial N_{z_i,k}}\right) &= z_{i,k-1}\left(\frac{\partial W^{k-1->k}}{\partial N_{z_i,k}}\right) - W^{k->k+1}\frac{1-z_{i,k}}{N_{tot,k}} \\ &- z_{i,k}\left(\frac{\partial W^{k->k+1}}{\partial N_{z_i,k}}\right)\end{aligned} \tag{4.37}$$

$$\begin{aligned}\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial N^k_{z_j}}\right) &= z_i^{k-1}\left(\frac{\partial W^{k-1->k}}{\partial N^k_{z_j}}\right) - \frac{W^{k->k+1} z_i^k}{N_z^k} \\ &- z_i^k\left(\frac{\partial W^{k->k+1}}{\partial N^k_{z_j}}\right)\end{aligned} \tag{4.38}$$

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial u^k}\right) = z_i^{k-1}\left(\frac{\partial W^{k-1->k}}{\partial u^k}\right) - z_i^k\left(\frac{\partial W^{k->k+1}}{\partial u^k}\right) \tag{4.39}$$

The k-1 node:

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial N^{k-1}_{z_i}}\right) = z_i^{k-1}\left(\frac{\partial W^{k-1->k}}{\partial N^{k-1}_{z_i}}\right) + W^{k-1->k}\left(\frac{\partial z_i^{k-1}}{\partial N^{k-1}_{z_i}}\right) \tag{4.40}$$

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial N^{k-1}_{z_j}}\right) = z_i^{k-1}\left(\frac{\partial W^{k-1->k}}{\partial N^{k-1}_{z_j}}\right) + W^{k-1->k}\left(\frac{\partial z_i^{k-1}}{\partial N^{k-1}_{z_j}}\right) \tag{4.41}$$

$$\left(\frac{\partial\left(\frac{dN_{z_i}^k}{dt}\right)}{\partial u^{k-1}}\right) = z_i^{k-1}\left(\frac{\partial W^{k-1->k}}{\partial u^{k-1}}\right) + W^{k-1->k}\left(\frac{\partial z_i^{k-1}}{\partial u^{k-1}}\right) \tag{4.42}$$

The k+1 node:

$$\left(\frac{\partial\left(\frac{dN_{z_i}^k}{dt}\right)}{\partial N_{z_i}^{k+1}}\right) = -z_i^k\left(\frac{\partial W^{k->k+1}}{\partial N_{z_i}^{k+1}}\right) \tag{4.43}$$

$$\left(\frac{\partial\left(\frac{dN_{z_i}^k}{dt}\right)}{\partial u^{k+1}}\right) = -z_i^k\left(\frac{\partial W^{k->k+1}}{\partial u^{k+1}}\right) \tag{4.44}$$

- The internal energy

$$\begin{aligned}\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial u^k}\right) = \frac{1}{N_z^k}\Bigg(&\left(h^{k-1}-u^k\right)\left(\frac{\partial W^{k-1->k}}{\partial u^k}\right)\\ &-\left(h^k-u^k\right)\left(\frac{\partial W^{k->k+1}}{\partial u^k}\right)\\ &-W^{k->k+1}\left(\frac{\partial h^k}{\partial u^k}\right) - W^{k-1->k} + W^{k->k+1}\Bigg)\end{aligned} \tag{4.45}$$

$$\begin{aligned}\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial N_{z,j}^k}\right) = \frac{1}{N_z^k}\Bigg(&-\left(\frac{du^k}{dt}\right) + \left(h^{k-1}-u^k\right)\left(\frac{\partial W^{k-1->k}}{\partial N_{z,j}^k}\right)\\ &-\left(h^k-u^k\right)\left(\frac{\partial W^{k->k+1}}{\partial N_{z,j}^k}\right) - W^{k->k+1}\left(\frac{\partial h^k}{\partial N_{z,j}^k}\right)\Bigg)\end{aligned} \tag{4.46}$$

The k-1 node:

$$\begin{aligned}\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial u^{k-1}}\right) = \frac{1}{N_z^k}\Bigg(&\left(h^{k-1}-u^k\right)\left(\frac{\partial W^{k-1->k}}{\partial u^{k-1}}\right)\\ &+W^{k-1->k}\left(\frac{\partial h^{k-1}}{\partial u^{k-1}}\right)\Bigg)\end{aligned} \tag{4.47}$$

$$\begin{aligned}\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial N_{z,j}^{k-1}}\right) = \frac{1}{N_z^k}\Bigg(&\left(h^{k-1}-u^k\right)\left(\frac{\partial W^{k-1->k}}{\partial N_{z,j}^{k-1}}\right)\\ &+W^{k-1->k}\left(\frac{\partial h^{k-1}}{\partial N_{z,j}^k}\right)\Bigg)\end{aligned} \tag{4.48}$$

The k+1 node:

$$\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial u^{k+1}}\right) = -\frac{h^k - u^k}{N_z^k}\left(\frac{\partial W^{k->k+1}}{\partial u^{k+1}}\right) \tag{4.49}$$

$$\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial N_{z,j}^{k+1}}\right) = -\frac{h^k - u^k}{N_z^k}\left(\frac{\partial W^{k->k+1}}{\partial N_{z,j}^{k+1}}\right) \tag{4.50}$$

4.2.2 Separation tank - Flash

Extending the node with one outlet.

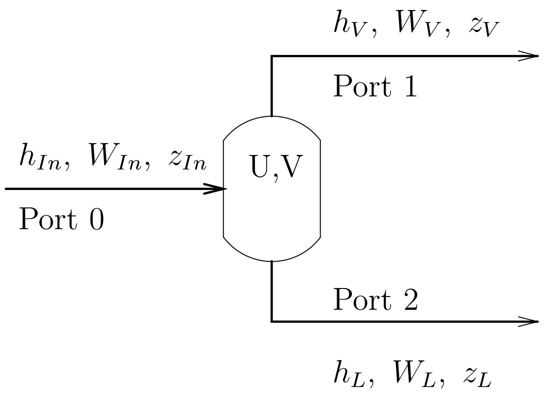


Figure 4.4: Separation tank

For simplification the pressure node index is dropped on the flow model sets.

Mass conservation

$$\left(\frac{dN_{z_i}}{dt}\right) = \sum_{pi \in IN_0} W_{pi} z_{pi,i} - \sum_{po \in OUT_1} W_{po} z_{V,i} - \sum_{po \in OUT_2} W_{po} z_{L,i} \tag{4.51}$$

Energy conservation

The energy conservation equation in its simplest form.

$$\left(\frac{dU}{dt}\right) = \sum_{pi \in IN_0} W_{pi} h_{pi} - \sum_{po \in OUT_1} W_{po} h_V - \sum_{po \in OUT_2} W_{po} h_L \tag{4.52}$$

Using the specific internal energy, gives the following differentials.

$$\begin{aligned}\left(\frac{du}{dt}\right) = \frac{1}{\sum_j N_{z_j}} \Bigg(&\sum_{pi \in IN_0} W_{pi}(h_{pi} - u) - \sum_{po \in OUT_1} W_{po}(h_V - u) \\ &- \sum_{po \in OUT_2} W_{po}(h_L - u) \Bigg)\end{aligned} \tag{4.53}$$

Analytical derivatives

The single phase density differentials in internal energy is given in Equations 4.54 and 4.55.

$$\begin{aligned}\left(\frac{\partial \rho_V}{\partial u}\right)_{N_{z,i}} = &\left(\frac{\partial \rho_V}{\partial T}\right)_{P,N_{z,i}} \left(\frac{\partial T}{\partial u}\right)_{N_{z,i}} + \left(\frac{\partial \rho_V}{\partial P}\right)_{T,N_{z,i}} \left(\frac{\partial P}{\partial u}\right)_{N_{z,i}} \\ &+ \sum_j \left(\frac{\partial \rho_V}{\partial N_{V,j}}\right)_{T,P,N_{V,k}} \left(\frac{\partial N_{V,j}}{\partial u}\right)_{N_{z,i}}\end{aligned} \tag{4.54}$$

$$\begin{aligned}\left(\frac{\partial \rho_L}{\partial u}\right)_{N_{z,i}} = &\left(\frac{\partial \rho_L}{\partial T}\right)_{P,N_{z,i}} \left(\frac{\partial T}{\partial u}\right)_{N_{z,i}} + \left(\frac{\partial \rho_L}{\partial P}\right)_{T,N_{z,i}} \left(\frac{\partial P}{\partial u}\right)_{N_{z,i}} \\ &+ \sum_j \left(\frac{\partial \rho_L}{\partial N_{L,j}}\right)_{T,P,N_{L,k}} \left(\frac{\partial N_{L,j}}{\partial u}\right)_{N_{z,i}}\end{aligned} \tag{4.55}$$

The differentials of the single phase mole numbers must also be supplied. For the homogeneous flow (node), these were known based on the fact that the total composition sums to one. Here the transfer between the phases must be considered.

The differentials can be generated in the following way:

$$\begin{aligned}\left(\frac{\partial N_{V,i}}{\partial u}\right)_{N_{z,k}} = &\left(\frac{\partial N_{V,i}}{\partial T}\right)_{P,N_{z,k}} \left(\frac{\partial T}{\partial u}\right)_{N_{z,j}} \\ &+ \left(\frac{\partial N_{V,i}}{\partial P}\right)_{T,N_{z,k}} \left(\frac{\partial P}{\partial u}\right)_{N_{z,j}}\end{aligned} \tag{4.56}$$

$$\begin{aligned}\left(\frac{\partial N_{V,i}}{\partial N_{z,j}}\right)_{u,N_{z,k}} = &\left(\frac{\partial N_{V,i}}{\partial T}\right)_{P,N_{z,l}} \left(\frac{\partial T}{\partial N_{z,j}}\right)_{u,N_{z,k}} \\ &+ \left(\frac{\partial N_{V,i}}{\partial P}\right)_{T,N_{z,l}} \left(\frac{\partial P}{\partial N_{z,j}}\right)_{u,N_{z,k}} \\ &+ \left(\frac{\partial N_{V,i}}{\partial N_{z,j}}\right)_{T,P,N_{z,l}}\end{aligned} \tag{4.57}$$

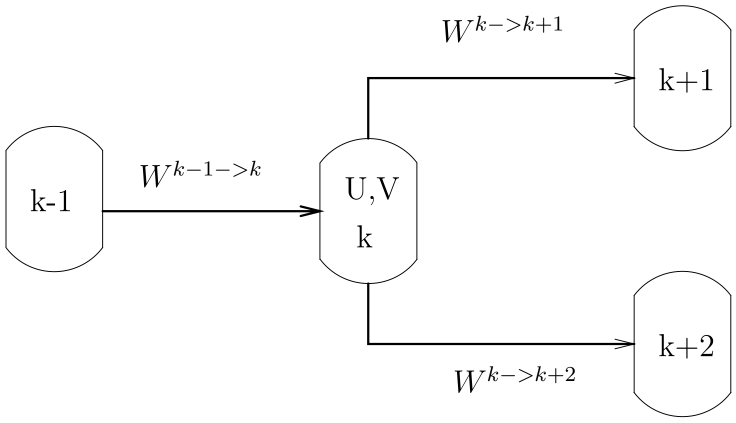


Figure 4.5: Separation tank reference system

Restate the conservation equations with the new indexing. The use of the flow model sets are dropped, because they will obscure the information communicated.

$$\left(\frac{dN_{z_i}^k}{dt}\right) = W^{k-1->k} z_i^{k-1} - W_{k->k+1} z_{V,i}^k - W_{k->k+1} z_{L,i}^k \tag{4.58}$$

$$\begin{aligned}\left(\frac{du^k}{dt}\right) = \frac{1}{\sum\limits_j N_{z_j}^k} \Bigg(& W^{k-1->k} h^{k-1} - W^{k->k+1} h_V^k - W^{k->k+2} h_L^k \\ & - W^{k-1->k} u^k + W^{k->k+1} u^k + W^{k->k+2} u^k \Bigg)\end{aligned} \tag{4.59}$$

Generally the vapor phase pressure is the dynamic state, and the liquid phase pressure is an algebraic relation to the vapor pressure. The algebraic relation used for the bottom liquid pressure is given in Equation 4.60.

$$P_L^k = P_V^k + C \frac{\rho_L^k N_L^k}{N_z^k} \tag{4.60}$$

Here N_L is the liquid holdup. C is a constant.

$$P_V^k = P^k \tag{4.61}$$

- The composition

The k node:

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial u^k}\right) = z_i^{k-1}\left(\frac{\partial W^{k-1->k}}{\partial u^k}\right) - z^k_{V,i}\left(\frac{\partial W_{k->k+1}}{\partial u^k}\right) - W_{k->k+1}\left(\frac{\partial z^k_{V,i}}{\partial u^k}\right) - z^k_{L,i}\left(\frac{\partial W_{k->k+2}}{\partial u^k}\right) - W_{k->k+2}\left(\frac{\partial z^k_{L,i}}{\partial u^k}\right) \tag{4.62}$$

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial N^k_{z_i}}\right) = z_i^{k-1}\left(\frac{\partial W^{k-1->k}}{\partial N^k_{z_i}}\right) - z^k_{V,i}\left(\frac{\partial W_{k->k+1}}{\partial N^k_{z_i}}\right) - W_{k->k+1}\left(\frac{\partial z^k_{V,i}}{\partial N^k_{z_i}}\right) - z^k_{L,i}\left(\frac{\partial W_{k->k+2}}{\partial N^k_{z_i}}\right) - W_{k->k+2}\left(\frac{\partial z^k_{L,i}}{\partial N^k_{z_i}}\right) \tag{4.63}$$

The k-1 node:

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial u^{k-1}}\right) = z_i^{k-1}\left(\frac{\partial W^{k-1->k}}{\partial u^{k-1}}\right) + W^{k-1->k}\left(\frac{\partial z_i^{k-1}}{\partial u^{k-1}}\right) \tag{4.64}$$

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial N^{k-1}_{z_i}}\right) = z_i^{k-1}\left(\frac{\partial W^{k-1->k}}{\partial N^{k-1}_{z_i}}\right) + W^{k-1->k}\left(\frac{\partial z_i^{k-1}}{\partial N^{k-1}_{z_i}}\right) \tag{4.65}$$

The k+1 node:

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial u^{k+1}}\right) = -z^k_{V,i}\left(\frac{\partial W^{k->k+1}}{\partial u^{k+1}}\right) \tag{4.66}$$

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial N^{k+1}_{z_i}}\right) = -z^k_{V,i}\left(\frac{\partial W^{k->k+1}}{\partial N^{k+1}_{z_i}}\right) \tag{4.67}$$

The k+2 node:

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial u^{k+2}}\right) = -z^k_{L,i}\left(\frac{\partial W^{k->k+2}}{\partial u^{k+2}}\right) \tag{4.68}$$

$$\left(\frac{\partial\left(\frac{dN^k_{z_i}}{dt}\right)}{\partial N^{k+1}_{z_i}}\right) = -z^k_{L,i}\left(\frac{\partial W^{k->k+1}}{\partial N^{k+2}_{z_i}}\right) \tag{4.69}$$

It is seen that the composition differentials can depend on u also. This is through phase splitting.

- The composition

The k node:

$$\begin{aligned}\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial u^k}\right) = \frac{1}{N^k_z}\Bigg[&(h^{k-1}-u^k)\left(\frac{\partial W^{k-1->k}}{\partial u^k}\right)\\ &- W^{k->k+1}\left(\frac{\partial h^k_V}{\partial u^k}\right) - (h^k_V - u^k)\left(\frac{\partial W^{k->k+1}}{\partial u^k}\right)\\ &- W^{k->k+2}\left(\frac{\partial h^k_L}{\partial u^k}\right) - (h^k_L - u^k)\left(\frac{\partial W^{k->k+2}}{\partial u^k}\right)\\ &- W^{k-1->k} + W^{k->k+1} + W^{k->k+2}\Bigg]\end{aligned} \tag{4.70}$$

$$\begin{aligned}\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial N^k_{z,i}}\right) = \frac{1}{N^k_z}\Bigg[&-\left(\frac{du^k}{dt}\right) + (h^{k-1}-u^k)\left(\frac{\partial W^{k-1->k}}{\partial N^k_{z,i}}\right)\\ &- W^{k->k+1}\left(\frac{\partial h^k_V}{\partial N^k_{z,i}}\right) - (h^k_V - u^k)\left(\frac{\partial W^{k->k+1}}{\partial N^k_{z,i}}\right)\\ &- W^{k->k+2}\left(\frac{\partial h^k_L}{\partial N^k_{z,i}}\right) - (h^k_L - u^k)\left(\frac{\partial W^{k->k+2}}{\partial N^k_{z,i}}\right)\Bigg]\end{aligned} \tag{4.71}$$

The k-1 node:

$$\begin{aligned}\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial u^{k-1}}\right) = \frac{1}{N^k_z}\Bigg[&(h^{k-1}-u^k)\left(\frac{\partial W^{k-1->k}}{\partial u^{k-1}}\right)\\ &+ W^{k-1->k}\left(\frac{\partial h^{k-1}}{\partial u^{k-1}}\right) - (h^k_V - u^k)\left(\frac{\partial W^{k->k+1}}{\partial u^{k-1}}\right)\\ &- (h^k_L - u^k)\left(\frac{\partial W^{k->k+2}}{\partial u^{k-1}}\right)\Bigg]\end{aligned} \tag{4.72}$$

$$\begin{aligned}\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial N_{z,i}^{k-1}}\right) = \frac{1}{N_z^k}\Bigg[&(h^{k-1}-u^k)\left(\frac{\partial W^{k-1->k}}{\partial N_{z,i}^{k-1}}\right) \\ &+ W^{k-1->k}\left(\frac{\partial h^{k-1}}{\partial u^{k-1}}\right) - (h_V^k - u^k)\left(\frac{\partial W^{k->k+1}}{\partial N_{z,i}^{k-1}}\right) \\ &- (h_L^k - u^k)\left(\frac{\partial W^{k->k+2}}{\partial N_{z,i}^{k-1}}\right)\Bigg]\end{aligned} \tag{4.73}$$

The k+1 node:

$$\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial u^{k+1}}\right) = \frac{u^k - h_V^k}{N_z^k}\left(\frac{\partial W^{k->k+1}}{\partial u^{k+1}}\right) \tag{4.74}$$

$$\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial N_{z,i}^{k+1}}\right) = \frac{u^k - h_V^k}{N_z^k}\left(\frac{\partial W^{k->k+1}}{\partial N_{z,i}^{k+1}}\right) \tag{4.75}$$

The k+2 node:

$$\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial u^{k+2}}\right) = \frac{u^k - h_L^k}{N_z^k}\left(\frac{\partial W^{k->k+2}}{\partial u^{k+2}}\right) \tag{4.76}$$

$$\left(\frac{\partial\left(\frac{du^k}{dt}\right)}{\partial N_{z,i}^{k+2}}\right) = \frac{u^k - h_L^k}{N_z^k}\left(\frac{\partial W^{k->k+2}}{\partial N_{z,i}^{k+2}}\right) \tag{4.77}$$

New requirements:

$$\begin{aligned}\left(\frac{\partial N_{V,i}^k}{\partial N_{z,i}^k}\right)_{N_{z,j},u,v} &= ? \quad \left(\frac{\partial N_{L,i}^k}{\partial N_{z,i}^k}\right)_{N_{z,j},u,v} = ? \\ \left(\frac{\partial N_{V,i}^k}{\partial u^k}\right)_{N_{z,j},u,v} &= ? \quad \left(\frac{\partial N_{L,i}^k}{\partial u^k}\right)_{N_{z,j},u,v} = ?\end{aligned} \tag{4.78}$$

$$\begin{aligned}\left(\frac{\partial N_{V,i}^k}{\partial N_{z,i}^k}\right)_{N_{z,j},u,v} = &\left(\frac{\partial N_{V,i}^k}{\partial T}\right)_{P,N_{z,j}^k}\left(\frac{\partial T}{\partial N_{z,i}^k}\right)_{u,v,N_{z,j}^k} \\ &+ \left(\frac{\partial N_{V,i}^k}{\partial P}\right)_{T,N_{z,j}^k}\left(\frac{\partial P}{\partial N_{z,i}^k}\right)_{u,v,N_{z,j}^k} \\ &+ \left(\frac{\partial N_{V,i}^k}{\partial N_{z,i}^k}\right)_{T,P,N_{z,j}^k}\end{aligned} \tag{4.79}$$

$$\left(\frac{\partial N^k_{L,i}}{\partial N^k_{z,i}}\right)_{N_{z,j},u,v} = 1 - \left(\frac{\partial N^k_{V,i}}{\partial N^k_{z,i}}\right)_{N_{z,j},u,v} \tag{4.80}$$

$$\begin{aligned}\left(\frac{\partial N^k_{V,i}}{\partial N^k_{z,l}}\right)_{N_{z,j},u,v} &= \left(\frac{\partial N^k_{V,i}}{\partial T}\right)_{P,N^k_{z,j}} \left(\frac{\partial T}{\partial N^k_{z,i}}\right)_{u,v,N^k_{z,j}} \\ &+ \left(\frac{\partial N^k_{V,i}}{\partial P}\right)_{T,N^k_{z,j}} \left(\frac{\partial P}{\partial N^k_{z,l}}\right)_{u,v,N^k_{z,j}} \\ &+ \left(\frac{\partial N^k_{V,i}}{\partial N^k_{z,l}}\right)_{T,P,N^k_{z,j}}\end{aligned} \tag{4.81}$$

$$\left(\frac{\partial N^k_{L,i}}{\partial N^k_{z,l}}\right)_{N_{z,j},u,v} = - \left(\frac{\partial N^k_{V,i}}{\partial N^k_{z,l}}\right)_{N_{z,j},u,v} \tag{4.82}$$

$$\begin{aligned}\left(\frac{\partial N^k_{V,i}}{\partial u^k}\right)_{v,N_{z,j}} &= \left(\frac{\partial N^k_{V,i}}{\partial T}\right)_{P,N^k_{z,j}} \left(\frac{\partial T}{\partial u^k}\right)_{v,N^k_{z,j}} \\ &+ \left(\frac{\partial N^k_{V,i}}{\partial P}\right)_{T,N^k_{z,j}} \left(\frac{\partial P}{\partial u^k}\right)_{v,N^k_{z,j}}\end{aligned} \tag{4.83}$$

$$\left(\frac{\partial N^k_{L,i}}{\partial u^k}\right)_{v,N_{z,j}} = - \left(\frac{\partial N^k_{V,i}}{\partial u^k}\right)_{v,N_{z,j}} \tag{4.84}$$

The enthalpy and density relations needed can be written out from this.

The liquid pressure need special consideration. See Equation 4.60.

$$\left(\frac{\partial P^k_L}{\partial u^k}\right) = \left(\frac{\partial P^k}{\partial u^k}\right) + \frac{C\rho_L}{N^k_z}\sum_j \left(\frac{\partial N^k_{L,j}}{\partial u^k}\right) + \frac{CN^k_L}{N^k_z}\left(\frac{\partial \rho_L}{\partial u^k}\right) \tag{4.85}$$

$$\begin{aligned}\left(\frac{\partial P^k_L}{\partial N^k_{z,i}}\right) &= \left(\frac{\partial P^k}{\partial N^k_{z,i}}\right) + \frac{C\rho_L}{N^k_z}\sum_j \left(\frac{\partial N^k_{L,j}}{\partial N^k_{z,i}}\right) + \frac{CN^k_L}{N^k_z}\left(\frac{\partial \rho_L}{\partial N^k_{z,i}}\right) \\ &- \frac{C\rho_L N^k_L}{(N^k_z)^2}\end{aligned} \tag{4.86}$$

The density and enthalpy leaving the node are evaluated at the vapor pressure. The pressure generated by the liquid is only used as a flow driver in the connecting flow module.

4.3 Tank and pipes - Enthalpy

4.3.1 Pipe - Node

There are two types of nodes using enthalpy as the conserved energy form. One conserves components, and the other one only copies the inlet composition to the outlet. As for the internal energy nodes, the node only have one port, and the port indexing for the pressure model will be dropped.

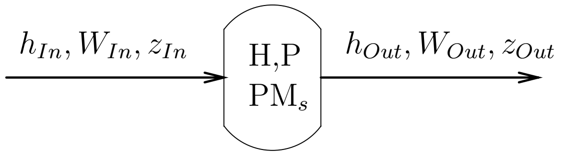


Figure 4.6: Node conserving enthalpy

Mass conservation

The mass conservation of the different components, is in the form of Equation 4.87.

$$\left(\frac{dN_{z_i}}{dt}\right) = \sum_{pi \in IN_s} W_{pi} z_{pi,i} - \sum_{po \in OUT_s} W_{po} z_i \tag{4.87}$$

Overall mass conservation

The overall mass conservation, is given in Equation 4.88.

$$\left(\frac{d\rho V}{dt}\right) = V\left(\frac{d\rho}{dt}\right) = \sum_{pi \in IN_s} W_{pi} - \sum_{po \in OUT_s} W_{po} \tag{4.88}$$

The density differential in Equation 4.88 is approximated to give an explicit pressure differential.

$$V\left(\frac{d\rho}{dt}\right) \approx V\left(\frac{\partial \rho}{\partial P}\right)_{h,\mathbf{N_z}} \left(\frac{dP}{dt}\right) = \tau_P \left(\frac{dP}{dt}\right) \tag{4.89}$$

Equations 4.89 and 4.88 are combined, to give a relation for the pressure differential.

$$\tau_P \left(\frac{dP}{dt}\right) = \sum_{pi \in IN_s} W_{pi} - \sum_{po \in OUT_s} W_{po} \tag{4.90}$$

The partial differential for the single phase density, at constant enthalpy and constant composition is derived below. Given constant composition, the density and enthalpy functions are given by Equation 4.91.

$$\rho = \rho(T, P), \qquad h = h(T, P) \tag{4.91}$$

The total differentials then become:

$$d\rho = \left(\frac{\partial \rho}{\partial T}\right)_P dT + \left(\frac{\partial \rho}{\partial P}\right)_T dP \tag{4.92}$$

$$dh = \left(\frac{\partial h}{\partial T}\right)_P dT + \left(\frac{\partial h}{\partial P}\right)_T dP \tag{4.93}$$

Setting the enthalpy change to zero:

$$dh = \left(\frac{\partial h}{\partial T}\right)_P dT + \left(\frac{\partial h}{\partial P}\right)_T dP = 0 \Rightarrow \left(\frac{\partial T}{\partial P}\right)_h = -\frac{\left(\frac{\partial h}{\partial P}\right)_T}{\left(\frac{\partial h}{\partial T}\right)_P} \tag{4.94}$$

Combining Equations 4.92 and 4.94:

$$\left(\frac{\partial \rho}{\partial P}\right)_h = \left(\frac{\partial \rho}{\partial P}\right)_T - \left(\frac{\partial \rho}{\partial T}\right)_P \frac{\left(\frac{\partial h}{\partial P}\right)_T}{\left(\frac{\partial h}{\partial T}\right)_P} \tag{4.95}$$

Energy conservation

When the node variables are integrated without the analytical Jacobian, the conservation equation has the following form.

$$\left(\frac{dH}{dt}\right) = \sum_{pi \in IN_s} W_{pi} h_{pi} - \sum_{po \in OUT_s} W_{po} h \tag{4.96}$$

4.3.2 Separation tank - Flash

Extending the node with one outlet.

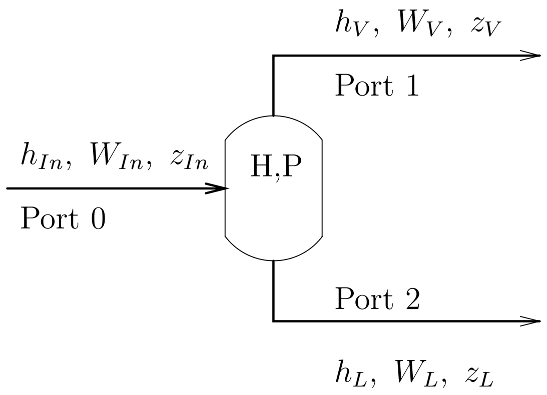


Figure 4.7: Separation tank conserving enthalpy

For simplification the pressure node index is dropped on the flow model sets.

Mass conservation

$$\left(\frac{dN_{z_i}}{dt}\right) = \sum_{pi \in IN_0} W_{pi} z_{pi,i} - \sum_{po \in OUT_1} W_{po} z_{V,i} - \sum_{po \in OUT_2} W_{po} z_{L,i} \tag{4.97}$$

The overall mass balance is then given by Equation 4.98

$$\left(\frac{d\rho V}{dt}\right) = V\left(\frac{d\rho}{dt}\right) = \sum_{pi \in IN_0} W_{pi} - \sum_{po \in OUT_1} W_{po} - \sum_{po \in OUT_2} W_{po} \tag{4.98}$$

The pressure differential can be obtained in same manner as in Section 4.3.1.

$$\tau_P \left(\frac{dP}{dt}\right) = \sum_{pi \in IN_0} W_{pi} - \sum_{po \in OUT_1} W_{po} - \sum_{po \in OUT_2} W_{po} \tag{4.99}$$

Energy conservation

The energy conservation equation in its simplest form.

$$\left(\frac{dH}{dt}\right) = \sum_{pi \in IN_0} W_{pi} h_{pi} - \sum_{po \in OUT_1} W_{po} h_V - \sum_{po \in OUT_2} W_{po} h_L \tag{4.100}$$

Chapter 5

Modelling of Flow Units

The following flow units are implemented to describe the LNG system:

- Valve
- Heat Exchanger
- Compressor
- Pump
- Liquid Expander

This introduction gives a short descriptions of the flow units. The model details are described later in this chapter.

The valve is treated isenthalpic and copies the inlet composition to the outlet. The valve is therefore a single AE, describing flow as a function of inlet density and pressure difference. The flow equation is given in Equation 5.1.

$$W = Cv\,hv\sqrt{\rho_{In}\Delta_{In,Out}P} \tag{5.1}$$

Here, Cv is the valve constant, *hv* is the percentual valve opening. Currently only linear valves are supported. The valve equations are given Section 5.1.

The heat exchanger is the most important unit in the LNG system. The general multistream heat exchanger model is described below in Section 5.2. Simplified heat exchangers are used to describe the water coolers. In the water coolers, the control of the outlet temperature is assumed to be fast, and the temperature at the outlet is therefore set to be constant.

The compressor flow is modelled by curve data. The model requires input points for the curve, which are interpolated by splines. The curve is interpolated using the fan laws. A constant polytropical efficiency is assigned to the compressor.

No motor/driver effects are modelled for the compressor. The compressor equations are given in Section 5.3.

The pump flow is modelled by Equation 5.2. If no parameter data is present, the program make a guess of appropriate parameters. The curve is scaled by the fan laws. The inlet composition is copied to the outlet, and the inlet enthalpy is corrected (increased), applying a constant efficiency parameter, e, for the pump. The algebraic enthalpy relation is given in Equation 5.3. No motor is attached to the pump, and the pump is therefore simulated with constant speed. The detailed pump equations are given in Section 5.4.

$$q = q_{Max}\sqrt{1 - \frac{H}{H_{Max}}} \tag{5.2}$$

Here, q is the volume flow through the pump, and H is the head. The pump parameters are q_{Max} and H_{Max}, which are the maximal possible volume flow through the pump and the maximal possible head for the pump, respectively.

$$h_{Out} = h_{In} + \frac{\Delta_{Out,In}P}{e\rho_{In}}, \qquad 0 < e < 1 \tag{5.3}$$

The liquid expander, Section 5.5, has valve flow dynamics, see Equation 5.1, and the outlet enthalpy is determined applying constant isentropic efficiency, γ. The outlet enthalpy is therefore described in the following way:

$$h_{Out} = \gamma \hat{h}_{Out}|_{S_{In},P_{Out}} + (1-\gamma)h_{In} \tag{5.4}$$

$\hat{h}_{Out}|_{S_{In},P_{Out}}$ is the the enthalpy evaluated in $(\hat{T}_{Out}, P_{Out}, \mathbf{z}_{In})$, where $\hat{T} = \hat{T}(S_{In}, P_{Out}, \mathbf{z}_{In})$. S_{In} is the entropy in the inlet node.

Common for all unit models is that they are simple, only conserving mass and energy, and utilising some fixed parameter values. The flow description is also simple, only using a valve equation (Bernoulli). The compressor and pump models are more sophisticated using curve data in the flow description. Equilibrium is always assumed for all unit models.

The simple unit models can easily be changed/improved within the existing framework. For example, adding curves or algebraic relations for the efficiency parameters or adding correlations for the heat transfer, for both the HP and UV formulation, will only imply changes to the one unit model class, and will be a moderate implementation task. All properties needed are interfaced and the Jacobian matrix structure is not changed.

Adding new dynamic states within the unit models will change the Jacobian matrix structure, and might therefore be a larger implementation task.

A summary of the model simplifications is now given, together with a brief discussion to indicate if the simplifications/assumptions are justified. The discussion is focused on the the heat exchanger model, and will give some information regarding model improvement/extension.

5.1 Valve

The valve is treated isenthalpically. The composition at the inlet is copied to the outlet. The flow through the valve is calculated from an algebraic relation, which is a quasi-stationary momentum equation.

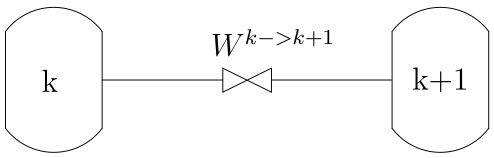


Figure 5.1: Valve - Flow from node k to node k+1

5.1.1 Algebraic flow relation

The flow equation in mass units is as follows:

$$W_{Mass} = Cv\,hv\sqrt{\rho_{Mass,k}\Delta_{k,k+1}P} \tag{5.5}$$

The indexes have references in Figure 5.1. Described in mole units:

$$W = \frac{Cv\,hv\sqrt{\rho_k Mw_k \Delta_{k,k+1}P}}{Mw_k} \tag{5.6}$$

5.1.2 Pressure - Flow - Linearization

The linearization in the valve flow function is simple when only the explicit relations to the pressure is considered. No restrictions are placed upon the flow directions.

$$\left(\frac{\partial W_{Mass}}{\partial P_k}\right)_{P_{k+1}} = \frac{Cv\,hv\sqrt{\rho_{Mass,k}}}{2\sqrt{\mathbf{abs}\left[\Delta_{k,k+1}P\right]}}\mathbf{sign}\left[\Delta_{k,k+1}P\right] \tag{5.7}$$

In the case where the flow through a valve is controlled by a PID controller the flow equation can take an implicit form. (That is; the closed loop flow must be written out using the controller parameters). The implicit form appears when the flow itself is measured. This could be avoided by introducing actuator dynamics.

For a controller with proportional and integral action, the flow equation is derived in Equations 5.8 to 5.10. The relation describing the PI controller is given in Equation 6.5.

$$\begin{aligned} W_{Mass} &= Cv\,hv(W_{Mass})\sqrt{\rho_{Mass,k}\Delta_{k,k+1}P} \\ &= hv(W_{Mass})f(\rho_{Mass,k}, P_k, P_{k+1}) \end{aligned} \tag{5.8}$$

$$hv(W_{Mass}) = Kp(r - W_{Mass}) + e_I \tag{5.9}$$

Inserting Equation 5.9 into Equation 5.8, and performing some algebraic manipulations, Equation 5.10 describes the closed loop mass flow.

$$W_{Mass} = \frac{(Kphvr + e_I)f}{1 + Kphvf} \tag{5.10}$$

5.1.3 Analytical derivatives

The derivatives for the flow is written out with respect to the differential variables in the connecting nodes.

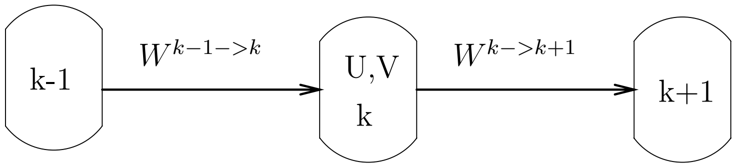


Figure 5.2: Node index reference for flow derivatives

This gives, in reference to Figure 5.2 and Equation 5.6:

$$W^{k->k+1}(N_{z_j,k}, N_{z_j,k+1}, u_k, u_{k+1}) = \frac{Cv\,hv\sqrt{\rho_k Mw_k(P_k - P_{k+1})}}{Mw_k} \tag{5.11}$$

The flow need to generate the following differentials for both nodes connected.

$$\begin{bmatrix} \left(\frac{\partial W}{\partial u}\right)_{N_{z_i}, i \in S_{NC}} & \left(\frac{\partial W}{\partial N_{z_1}}\right)_{u, N_{z_j}, j \in S_{NC,1}} & \cdots \left(\frac{\partial W}{\partial N_{z_{NC}}}\right)_{u, N_{z_j}, j \in S_{NC,NC}} \end{bmatrix} \tag{5.12}$$

The needed differentials are written out below, in Equations 5.13 through 5.21.

$$\left(\frac{\partial W}{\partial u}\right)_{N_{z,i}} = \left(\frac{\partial W}{\partial T}\right)_{P,N_{z_i}} \left(\frac{\partial T}{\partial u}\right)_{N_{z_i}} + \left(\frac{\partial W}{\partial P}\right)_{T,N_{z_i}} \left(\frac{\partial P}{\partial u}\right)_{N_{z_i}}, \quad i \in S_{NC} \tag{5.13}$$

$$\begin{aligned}\left(\frac{\partial W}{\partial N_{z_l}}\right)_u &= \left(\frac{\partial W}{\partial T}\right)_{P,N_{z_i}}\left(\frac{\partial T}{\partial N_{z_l}}\right)_{u,N_{z_j}} \\ &+ \left(\frac{\partial W}{\partial P}\right)_{T,N_{z_i}}\left(\frac{\partial P}{\partial N_{z_l}}\right)_{u,N_{z_j}} \\ &+ \left(\frac{\partial W}{\partial N_{z_l}}\right)_{P,T,N_{z_j}}, \quad i \in S_{NC} \quad j \in S_{NC,l}\end{aligned} \tag{5.14}$$

For the valve equation this gives.

$$\left(\frac{\partial W^{k->k+1}}{\partial T}\right)_{P_k,N_{z_i}} = \frac{Cv\,hv\sqrt{(P_k-P_{k+1})}}{2\sqrt{\rho_k Mw_k}}\left(\frac{\partial \rho_k}{\partial T}\right)_{P,N_{z_i}}, \quad i \in S_{NC} \tag{5.15}$$

$$\begin{aligned}\left(\frac{\partial W^{k->k+1}}{\partial P_k}\right)_{T,N_{z_i}} &= \frac{Cv\,hv\sqrt{(P_k-P_{k+1})}}{2\sqrt{\rho_k Mw_k}}\left(\frac{\partial \rho_k}{\partial P_k}\right)_{T,N_{z_i}} \\ &+ \frac{Cv\,hv\sqrt{\rho_k Mw_k}}{2\sqrt{(P_k-P_{k+1})}}, \quad i \in S_{NC}\end{aligned} \tag{5.16}$$

$$\begin{aligned}\left(\frac{\partial W^{k->k+1}}{\partial N_{z_l}}\right)_{P,T,N_{z_i}} &= \frac{Cv\,hv\sqrt{(P_k-P_{k+1})}}{2Mw_k\sqrt{\rho_k Mw_k}}\left(Mw_k\left(\frac{\partial \rho_k}{\partial N_{z_l}}\right)_{P,T,N_{z_i}}\right. \\ &\left. + \rho_k\left(\frac{\partial Mw_k}{\partial N_{z_l}}\right)_{P,T,N_{z_i}}\right) \\ &+ \frac{W^{k->k+1}}{Mw_k^2}\left(\frac{\partial Mw_k}{\partial N_{z_l}}\right)_{P,T,N_{z_i}}, \quad i \in S_{NC,l}\end{aligned} \tag{5.17}$$

$$\left(\frac{\partial W^{k->k+1}}{\partial P_{k+1}}\right)_{P_k,T,N_{z_i}} = -\frac{Cv\,hv\sqrt{\rho_k Mw_k}}{2Mw_k\sqrt{(P_k-P_{k+1})}}, \quad i \in S_{NC} \tag{5.18}$$

The differentials written out with respect to the state variables:

$$\begin{aligned}\left(\frac{\partial W^{k->k+1}}{\partial u_k}\right)_{N_{z_i,k}} &= \frac{Cv\,hv\sqrt{(P_k-P_{k+1})}}{2\sqrt{\rho_k Mw_k}}\left(\frac{\partial \rho_k}{\partial u_k}\right)_{N_{z_i,k}} \\ &+ \frac{Cv\,hv\sqrt{\rho_k}}{2\sqrt{(P_k-P_{k+1})Mw_k}}\left(\frac{\partial P_k}{\partial u_k}\right)_{N_{z_i,k}} \\ &- \frac{W^{k->k+1}}{2Mw_k}\left(\frac{\partial Mw_k}{\partial u_k}\right)_{N_{z_i,k}}, \quad i \in S_{NC}\end{aligned} \tag{5.19}$$

$$\begin{aligned}\left(\frac{\partial W^{k->k+1}}{\partial u_{k+1}}\right)_{N_{z_i,k+1}} =& \frac{Cv\,hv\sqrt{\rho_k}}{2\sqrt{(P_k-P_{k+1})Mw_k}}\left(\frac{\partial P_{k+1}}{\partial u_{k+1}}\right)_{N_{z_i,k+1}}, \\ & i \in S_{NC}\end{aligned} \tag{5.20}$$

The compositional differentials are identical in form, and are therefore not written out.

When the mass flow is in a closed loop form, as given in Equation 5.10, the additional differential shown in Equation 5.21 must be used.

$$\left(\frac{\partial W_{Mass}}{\partial f}\right) = \frac{(Kphvr + e_I)}{(1 + Kphvf)^2} \tag{5.21}$$

The overall flow differentials will therefore be a combination of Equation 5.21 and the above differentials.

5.2 Heat exchanger

The heat exchanger is the most important and most difficult unit model for in the LNG plant. This section summarizes the equations used to describe the heat exchanger.

Figure 5.3 shows how the system is discretized. The streams are staggered in pressure and mass flow. The wall temperature is displaced from the stream temperature by half a CV. The number of wall states is then N+1. The number of stream CVs are N+2, but only N+1 have dynamic energy states. The inlet CV is only a dummy. Equation 5.22 defines the set for the CV index, with and without the inlet CV index.

$$S_{N+1} = 1, .., N+1 \tag{5.22}$$

$$S_{N+1,1} = 2, .., N+1 \tag{5.23}$$

The energy equation is upstream or central difference, CD. This is set through an α parameter. $(1.0 - \alpha)$ scales the CD scheme, and α scales the upstream scheme.

The different properties are interpolated to the staggered points. This is either upstream or linear "interpolation". This interpolation is weighted by different α parameters.

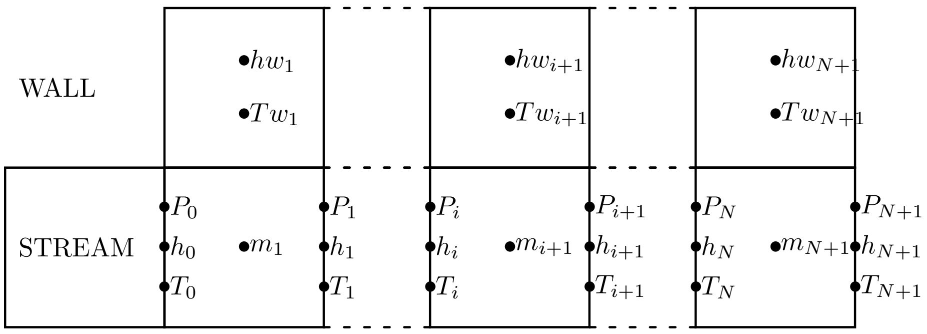


Figure 5.3: Stream and wall discretisation

Define the interpolation parameters used:

- α Enthalpy interpolation
- α_{AP1} Flow interpolation to CV boundarys
- α_{AP2} Property interpolation to CV midpoint
- α_{AT} Temperature interpolation in the streams

Setting the alpha parameters to one will give the simplest interpolation, pure upstream. That is; only the closest upstream value is used. This gives the general interpolation formula for the arbitrary parameter γ.

$$\gamma_I = (1.0 - \alpha)\frac{\gamma_{i-1} + \gamma_i}{2} + \alpha\gamma_{i-1} \tag{5.24}$$

5.2.1 Assumptions and simplifications

The major simplification is on the flow and pressure. The mass flow is assumed constant through the heat exchanger, and the pressure is assumed to be linear between inlet and outlet.

The composition dynamics, are considered to be infinitely fast, and the inlet composition, of an specific stream, is used through the entire heat exchanger. That is; the stream in the heat exchanger has the same compositional dynamics as the node at the inlet.

Dispersion or diffusion in the flow direction are therefore not needed. The conduction in the flow direction is also neglected. Heat transfer through radiation is neglected.

The only heat transfer considered is the heat transfer between a stream and its wall(s).

Conduction in the wall material is neglected.

The enthalpy is used as the conserved property. Not the internal energy.

After making these assumptions, the equations describing the heat exchanger can be written out. The conservation equations will be written out with reference to Figures 5.4 and 5.5.

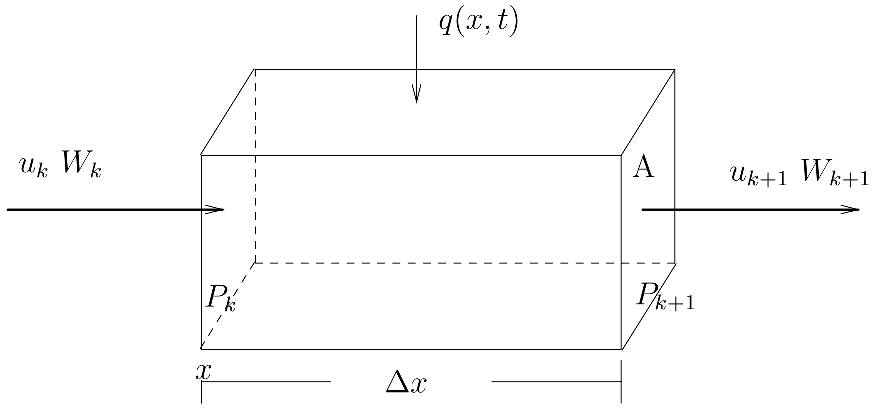


Figure 5.4: Reference system for the stream conservation equations

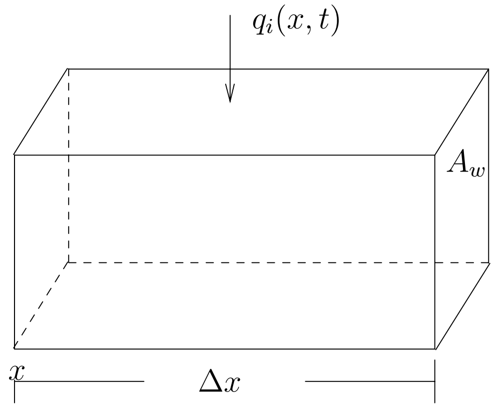


Figure 5.5: Reference system for the wall conservation equations

5.2.2 Algebraic flow relation

The flow description is simple. The paper submitted to a related work Hammer et al. (2003) shows that this simple relation is sufficient to predict the major dynamics.

Stating the simple flow pressure system under the assumption $P_{in} > P_{out}$:

$$W = k\sqrt{P_{in} - P_{out}} \tag{5.25}$$

The fact that this is a streamwise relation is implicitly implied, and will not be indexed in any way. The linear pressure profile is described with the following abstraction:

$$P_i = P_{in} + i\frac{(P_{out} - P_{in})}{(N+1)}, \ i \in S_{N+1} \tag{5.26}$$

The differentials, of the pressures, are needed in the following sections, and are stated here.

$$\left.\begin{aligned} \left(\frac{\partial P_i}{\partial P_{in}}\right)_{P_{out}} &= 1 - \tfrac{i}{(N+1)} \\ \left(\frac{\partial P_i}{\partial P_{out}}\right)_{P_{in}} &= \tfrac{i}{(N+1)} \end{aligned}\right\} i \in S_{N+1} \tag{5.27}$$

5.2.3 Energy conservation

The stream energy conservation equation

As seen from Figure 1.2, every tube stream is a compilation of several small diameter tubes. The volume of all the tubes are lumped together. The tubes have constant diameter, and have approximately the same length. The volume can therefore be assumed linear between the top and bottom of the heat exchanger. The same applies to the shell stream, the volume is described as linear, and the flow cross section is assumed constant. The shell stream has contact surface to all the tube stream walls, and therefore it exchanges heat with all these walls. The outer wall is assumed well insulated, and no heat is exchanged with the outer wall.

After lumping the volumes of the streams, one common spatial variable is introduced for the streams. All the physical property functions are therefore a function of this spatial variable and time. The only spatial variable is the height of the heat exchanger. The stream properties considered can therefore be thought of as cross section averages.

The equations will be derived using Figure 5.4 as reference. First the mass balance is written out. It is needed in the derivation of the energy equation. Using the principle of micro balances, the mass balance is written out in Equations 5.28 and 5.29.

$$\begin{aligned} \frac{\Delta(\rho A \Delta x)}{\Delta t} &= W_k - W_{k+1} \\ A\frac{\Delta\rho}{\Delta t} &= -\frac{(W_{k+1} - W_k)}{\Delta x} \end{aligned} \tag{5.28}$$

Dividing through with Δx, and using the fact that the area normal to the flow direction is constant. When the Δx and Δt are made infinitely small the mass balance is established.

$$\begin{gathered} \lim_{\Delta x \to 0,\ \Delta t \to 0} \left[A\frac{\Delta\rho}{\Delta t} = -\frac{(W_{k+1} - W_k)}{\Delta x} \right] \\ \Downarrow \\ A\left(\frac{\partial\rho}{\partial t}\right)_x = -\left(\frac{\partial W}{\partial x}\right)_t \end{gathered} \tag{5.29}$$

The same approach is applied with the energy. Recognizing that $h = u + Pv$, the energy conservation equation is written out in Equations 5.30 through 5.33.

$$\frac{\Delta(u\rho A\Delta x)}{\Delta t} = W_k u_k - W_{k+1}u_{k+1} + W_k P_k v_k - W_{k+1}P_{k+1}v_{k+1} + \int_x^{x+\Delta x} q\, dx$$

$$A\frac{\Delta(u\rho)}{\Delta t} = -\frac{(W_{k+1}h_{k+1} - W_k h_k)}{\Delta x} + \frac{1}{\Delta x}\int_x^{x+\Delta x} q\, dx \tag{5.30}$$

$$\lim_{\Delta x\to 0,\ \Delta t\to 0}\left[A\frac{\Delta(u\rho)}{\Delta t} = -\frac{(W_{k+1}h_{k+1} - W_k h_k)}{\Delta x} + \frac{1}{\Delta x}\int_x^{x+\Delta x} q\, dx\right]$$
$$\Downarrow \tag{5.31}$$
$$A\left(\frac{\partial(u\rho)}{\partial t}\right)_x = -\left(\frac{\partial hW}{\partial x}\right)_t + q(x,t)$$

Combining Equation 5.29 and Equation 5.31 we get the following, without forcing any new restrictions on the equations.

$$A\rho\left(\frac{\partial u}{\partial t}\right)_x = -W\left(\frac{\partial h}{\partial x}\right)_t + q(x,t) \tag{5.32}$$

Introducing the approximation $u \approx h$ then give

$$A\rho\left(\frac{\partial h}{\partial t}\right)_x = -W\left(\frac{\partial h}{\partial x}\right)_t + q(x,t) \tag{5.33}$$

Using the MOL, the PDE is transformed to an ODE, were time is the only continuous variable. This is shown in Equation 5.34, which is the final energy equation. The approximation to the spatial differential will be applied in an upstream way. That is; in the purely explicit case, it can described by the stencil shown in Figure 5.6.

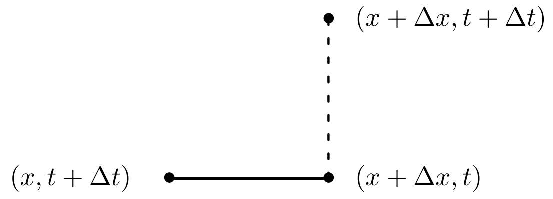


Figure 5.6: Explicit upstream stencil

$$V\rho\left(\frac{dh_{k+1}}{dt}\right) = -W\Delta_{k+1,k}h + Q, \qquad Q = \int_{x}^{x+\Delta x} q(x,t)\ dx \tag{5.34}$$

When we combine the mass equation with the original energy equation, the result is Equation 5.34. This equation requires mass conservation. When the mass flow is treated as constant, and the density is in the form $\rho(T(h), P, \mathbf{z})$, the mass conservation equation is violated.

The wall energy conservation equation

The wall volume and mass is lumped together and both are treated as linear from the top to bottom of the heat exchanger. The heat transfer is only considered to and from the tube side and shell stream. Conduction in the wall is neglected.

The equations will be derived using Figure 5.5 as reference. Equations 5.35 and 5.36 show the derivation of the wall energy conservation equation.

$$\begin{aligned}\frac{\Delta\left(hw\rho_w A_w \Delta x\right)}{\Delta t} &= \sum_i \int_x^{x+\Delta x} q_i\ dx \\ A_w\rho_w\frac{\Delta hw}{\Delta t} &= \sum_i \frac{1}{\Delta x}\int_x^{x+\Delta x} q_i\ dx\end{aligned} \tag{5.35}$$

$$\begin{aligned}&\lim_{\Delta x\to 0,\ \Delta t\to 0}\left[A_w\rho_w\frac{\Delta hw}{\Delta t} = \sum_i \frac{1}{\Delta x}\int_x^{x+\Delta x} q\ dx\right] \\ &\Downarrow \\ &A_w\rho_w\left(\frac{\partial hw}{\partial t}\right)_x = \sum_i q_i\end{aligned} \tag{5.36}$$

The wall energy equation, Equation 5.37, exhibits no spatial effects, and is therefore in ODE form.

$$M_w\left(\frac{\partial hw}{\partial t}\right)_x = \sum_i q_i \tag{5.37}$$

The enthalpy of the wall will be treated in the simplified way, linear in temperature, shown in Equation 5.38.

$$hw = C_{V,w}Tw \tag{5.38}$$

Heat transfer

Nothing is said so far about the actual heat transfer function, $q(x,t)$. To simplify the notation, the integrated heat transfer function is defined in Equation 5.39.

$$Q(x,\Delta x,t) = \int_{x}^{x+\Delta x} q(x,t)dx = \int_{x}^{x+\Delta x} u(x,t)l(x,t)(T(x,t) - Tw(x,t))dx \quad (5.39)$$

The perimeter $l(x,t)$ is constant, l. The heat transfer area is then described as $A_l(\Delta x) = l\Delta x$. Introducing the defining Equation 5.40 for $U(x,\Delta x,t)$ the heat transfer function then get the form of Equation 5.41.

$$\begin{aligned} U(x,\Delta x,t) &\int_{x}^{x+\Delta x} (T(x,t) - Tw(x,t))dx \\ &= \int_{x}^{x+\Delta x} u(x,t)(T(x,t) - Tw(x,t))dx \end{aligned} \quad (5.40)$$

$$Q(x,\Delta x,t) = U(x,\Delta x,t)A_l(\Delta x)\frac{1}{\Delta x}\int_{x}^{x+\Delta x} (T(x,t) - Tw(x,t))dx \quad (5.41)$$

The mean temperature difference over the interval x and $x + \Delta x$ is usually approximated with a constant value for both temperatures. The selection of the value is typically done with an interpolation operator, $L_{x,\Delta x}$. The approximating equation for the heat transfer then becomes 5.42.

$$Q(x,\Delta x,t) \approx U(x,\Delta x,t)A_l(\Delta x)L_{x,\Delta x}(T(x,t) - Tw(x,t)) \quad (5.42)$$

Experimental work, at SINTEF Energy Research and NTNU, Department of Energy and Process Engineering (formerly Department of Refrigeration and Air Conditioning), with coil heat exchanger test sections has produced correlations for $U(x,\Delta x,t)$. These correlations, made for steady flow, do not have the smoothness properties in function values, and differentials to be used in this work. The heat transfer coefficient, U, is therefore set constant, which is a crude assumption. (Every stream gets its own heat transfer coefficient.)

A important discontinuity is found in the heat exchanger model. More specifically in the heat transfer model. The heat transfer model uses the temperature of the stream to describe the driving potential for heat transfer. From a thermodynamic perspective the temperature must be described by Equation 5.43.

$$T = T(h,P,\mathbf{N_z}) \quad (5.43)$$

As pointed out from Wagner (1998), this function has a discontinuous differential function. The discontinuities are located on the phase boundaries. These discontinuities will complicate the integration.

Several articles, Wagner (1998), Verwer, Blom, and Sanz-Serna (1989) and Verwer and Trompert (1993), describe how it is possible to avoid the problem. No action is taken in this work to remove the discontinuities.

The heat transfer will be given by Equation 5.44.

$$Q = q\Delta x = UA(Tw - T_s) \tag{5.44}$$

Summary of the heat exchanger simplifications

Table 5.1 shows a summary of the simplifications/assumptions made when modeling the heat exchanger.

The effect of the neglected physics is difficult to evaluate, and some simplifications are made to make the model solvable. The discussion might therefore seem qualitative.

The use of constant heat transfer coefficients are not valid. The ratio of the heat transfer coefficient in liquid phase and in gas phase is in the order of 10 for a typical LNG heat exchanger stream. At the same time the heat transfer from a stream depends greatly on the flow rate. The constant heat transfer coefficients are used for simplicity.

Table 5.1: Heat exchanger assumptions

Ass. no.	Assumption	Valid [Y/N/U][a]
1	Constant heat transfer coefficient	NO
2	No axial conduction in the wall material	UNKNOWN
3	No heat transfer in wall supporting material	UNKNOWN
4	No heat transfer through radiation	UNKNOWN
5	No diffusion/dispersion in axial direction	UNKNOWN
6	Reducing to one spatial dimension	UNKNOWN
7	MOL PDE to ODE transformation, fixed grid	UNKNOWN
8	Enthalpy conservation instead of internal energy	NO
9	Infinite composition dynamics	UNKNOWN
10	Equilibrium, same T and P in both phases	UNKNOWN
11	Const mass flow and linear pressure drop	UNKNOWN
12	Flow described using valve equation	NO
13	Influence of the gravity is neglected	NO

[a]Short for [YES/NO/UNKNOWN]

The neglected axial conduction in the wall material might seem crude. Taking into account a winding angle of about 10 degrees, the assumption seems less crude, but might still give a noticeable contribution to the model dynamics, and even the stationary states. Therefore this assumption should be verified.

Using the enthalpy as conserved state, instead of internal energy, will introduce an error. At least for the system generally conserving internal energy the energy state should be changed to internal energy. The reason for not having the feature to use internal energy in the heat exchanger is simple. The heat exchanger model was first implemented with enthalpy as conserved state, and there has not been time available to extend the model.

The infinite composition dynamics and the constant flow assumption is partly justified through the low space time, τ (see Equation B.24), in the heat exchangers. For the tube streams, $\tau \approx 8 - 10$ seconds, but for the shell streams the space time can be several times larger, and the simplification becomes more questionable.

The assumption of equilibrium is common for both static and dynamic process simulation. The assumption of equilibrium is often wrong in dynamic simulation, but lacking measurement data and correlations, the only choice is to assume equilibrium. The effect of this assumption is difficult to foresee for the various heat exchangers.

The neglected gravity, and the description of the flow using a valve equation is not good. Especially at low pressures and low flow rates these assumptions will fail. At least the two-phase shell flow is poorly described using these model simplifications. This is shown by Hammer et al. (2003).

Some simplifications are enforced simply to get a model. All the below heat exchanger simplifications fall into this category. The heat transfer through electromagnetic radiation is difficult to describe in the geometry of the coil heat exchanger. Considering the low temperatures and the relative small temperature differences through the heat exchanger, the simplification of removing the contribution from radiation is justified.

The reduction to a single spatial dimension is necessary, in order to get a simple and relatively fast model. This means that an average over the two remaining dimension is used. The first order approximation of the spatial partial differential in the energy conservation Equation 5.33, requires small discretization. A fixed grid with 11 dynamic states is used in all heat exchangers. The error is proportional to the discretization size and the curvature of the enthalpy, see Cheney and Kincaid (1999). That is; the error can be large if a stream changes phase between two discretization points, and if the heat transfer coefficient is changing. The axial averaging of the heat transfer coefficient may also introduce a large error.

Heat transfer in wall supporting material would imply direct heat transfer between the wall materials. This will only give a contribution to the overall heat transfer if the walls have different temperatures. To model these effects, very detailed information about the heat exchanger is needed, and it is therefore preferred not to include this effect.

A comparison between model and measurement data is performed by Hammer et al. (2003). They show reasonable agreement between measurement data from a test facility and a model similar to the model used in these simulator. The main difference is the use of a correlation for the heat transfer coefficients. Their model is presented by Vist et al. (2003). A simplified flow description, using a Bernoulli relation, fits measured data for all but the low pressure shell streams. That is; despite all the simplifications, the model gives reasonable predictions.

5.2.4 Pressure - Flow - Linearization

The algebraic flow relations are differentiated in pressure to yield the necessary flow differentials.

$$\begin{aligned} \left(\frac{\partial W}{\partial P_{in}}\right)_{P_{out}} &= \frac{k}{2\sqrt{P_{in}-P_{out}}} \\ \left(\frac{\partial W}{\partial P_{out}}\right)_{P_{in}} &= \frac{-k}{2\sqrt{P_{in}-P_{out}}} \end{aligned} \tag{5.45}$$

5.2.5 Analytical derivatives

All alpha interpolation parameters are restricted to one, except α_{AT} that may vary freely.

When the property interpolation parameter, α_{AP1} , is restricted to one, the density of a CV is then described entirely by its inflowing density. The index of the density is therefore i-1 in Equation 5.34.

The linear interpolation of the stream temperature, is given by Equations 5.46 and 5.47, after using the general interpolation formula given in Equation 5.24

$$T_{I,s} = (1.0-\alpha_{AT})\frac{T_{i-1}+T_i}{2} + \alpha_{AT}T_{i-1} \tag{5.46}$$

$$\begin{aligned} \left(\frac{\partial T_{I,s}}{\partial T_i}\right) &= \frac{1.0-\alpha_{AT}}{2} \\ \left(\frac{\partial T_{I,s}}{\partial T_{i-1}}\right) &= \frac{1.0+\alpha_{AT}}{2} \end{aligned} \tag{5.47}$$

The energy conservation Equation 5.34 is restated in Equation 5.48, with the proper indexing.

$$\left(\frac{dh_i}{dt}\right) = \frac{1}{V\rho_{i-1}}\Big(W(h_{i-1}-h_i) + U_iA_i(Tw_i - T_{I,s})\Big) \tag{5.48}$$

Defining HR, to simplify the equation writing.

$$HR = W(h_{i-1}-h_i) + U_iA_i(Tw_i - T_{I,s}) \tag{5.49}$$

The heat transfer will be treated as in Equation 5.50.

$$U_i = U_i(W, T_i, P_i, N_{x,i,j}, N_{y,i,j}) \tag{5.50}$$

The dynamic description of the wall is restated, with the proper subscripts. Subscript s1 is used for the shell side, and s2 is used for the tube side.

$$\begin{aligned} \left(\tfrac{dhw_i}{dt}\right) &= -\frac{1}{V_W\rho_W}\Big[(U_iA_i)_1(T_{W,i}-T_{s1}) + (U_iA_i)_2(T_{W,i}-T_{s2})\Big] \\ &= \frac{1}{V_W\rho_W}\Big[(U_iA_i)_1\Big(\frac{hw_i}{C_{V,Wall}} - T_{s1}\Big) \\ &\quad +(U_iA_i)_2\Big(\frac{hw_i}{C_{V,Wall}} - T_{s2}\Big)\Big] \end{aligned} \tag{5.51}$$

The internal derivatives

First the energy conservation equations are differentiated in the internal heat exchanger states. The result is shown in Equations 5.52 through 5.57.

$$\begin{aligned}\left(\frac{\partial\left(\frac{dh_i}{dt}\right)}{\partial h_i}\right)_{W,P_i,N_{z,i,j}} &= -\frac{1}{V\rho_{i-1}}\left[W + U_iA_i\left(\frac{\partial T_{I,s}}{\partial h_i}\right)_{W,P_i,N_{z,i,j}}\right]\\ &= -\frac{1}{V\rho_{i-1}}\left[W + A_iU_i\frac{\left(\frac{\partial T_{I,s}}{\partial T_i}\right)}{\left(\frac{\partial h_i}{\partial T_i}\right)_{W,P_i,N_{z,i,j}}}\right]\\ &= -\frac{1}{V\rho_{i-1}}\left[W + \frac{A_iU_i(1.0-\alpha_{AT})}{2\left(\frac{\partial h_i}{\partial T_i}\right)_{W,P_i,N_{z,i,j}}}\right],\\ & i \in S_{N+1}\end{aligned} \tag{5.52}$$

$$\begin{aligned}\left(\frac{\partial\left(\frac{dh_i}{dt}\right)}{\partial h_{i-1}}\right)_{h_i,P_i,N_{z,i,j}} &= \frac{1}{V\rho_{i-1}}\Bigg[W - A_iU_i\left(\frac{\partial T_{I,s}}{\partial h_{i-1}}\right)\\ &+A_i(T_{W,i}-T_{I,s})\left(\frac{\partial U_i}{\partial h_{i-1}}\right)_{W,P_i,N_{z,i,j}}\\ &-\frac{HR}{\rho_{i-1}}\left(\frac{\partial \rho_{i-1}}{\partial h_{i-1}}\right)_{W,P_{i-1},N_{z,i-1,j}}\Bigg]\\ &= \frac{1}{V\rho_{i-1}}\Bigg\{W + \Bigg[-A_iU_i\frac{1.0+\alpha_{AT}}{2}\\ &+A_i(T_{W,i}-T_{I,s})\left(\frac{\partial U_i}{\partial T_{i-1}}\right)_{W,P_i,N_{z,i,j}}\\ &-\frac{HR}{\rho_{i-1}}\left(\frac{\partial \rho_{i-1}}{\partial T_{i-1}}\right)_{W,P_{i-1},N_{z,i-1,j}}\Bigg]\frac{1}{\left(\frac{\partial h_{i-1}}{\partial T_{i-1}}\right)_{W,P_{i-1},N_{z,i-1,j}}}\Bigg\},\\ & i \in S_{N+1}\end{aligned} \tag{5.53}$$

The water stream has constant properties, and the differentials become simpler. Equations 5.52 and 5.53 are restated, inserting the constant values, to give Equations 5.54 and 5.55.

$$\begin{aligned}\left(\frac{\partial\left(\frac{dh_i}{dt}\right)}{\partial h_i}\right)_{W,P_i,N_{z,i,j}} &= -\frac{1}{V\rho_{i-1}}\left[W + \frac{A_iU_i(1.0-\alpha_{AT})}{2C_{p,W}}\right],\\ & i \in S_{N+1}\end{aligned} \tag{5.54}$$

$$\begin{aligned}\left(\frac{\partial\left(\frac{dh_i}{dt}\right)}{\partial h_{i-1}}\right)_{h_i,P_i,N_{z,i,j}} &= \frac{1}{V\rho_{i-1}}\Bigg\{W + \Bigg[-A_iU_i\frac{1.0+\alpha_{AT}}{2}\\ &+A_i(T_{W,i}-T_{I,s})\left(\frac{\partial U_i}{\partial T_{i-1}}\right)_{W,P_i,N_{z,i,j}}\Bigg]\frac{1}{C_{p,W}}\Bigg\},\\ & i \in S_{N+1}\end{aligned} \tag{5.55}$$

The wall enthalpy energy equation differential is written out in Equation 5.56.

$$\left(\frac{\partial\left(\frac{dhw_i}{dt}\right)}{\partial h_{W,i}}\right) = -\frac{1}{V_W \rho_W C_{V,Wall}}[(U_iA_i)_1 + (U_iA_i)_2], \quad i \in S_{N+1} \tag{5.56}$$

The wall differential is only written out with respect to one of the streams. The result is given in Equation 5.57.

$$\begin{aligned}\left(\frac{\partial\left(\frac{dhw_i}{dt}\right)}{\partial h_{s1}}\right) &= -\frac{1}{V_W\rho_W}\Bigg[-(U_iA_i)_1\left(\frac{\partial T_{s1}}{\partial h_{s1}}\right)_{P_{s1},N_{z,s1,j}} \\ &\quad +\left(\frac{h_{W,i}}{C_{V,Wall}} - T_{s1}\right)\left(\frac{\partial (U_iA_i)_1}{\partial h_{s1}}\right)_{P_{s1},N_{z,s1,j}}\Bigg] \\ &= -\frac{1}{V_W\rho_W}\Bigg[-(U_iA_i)_1 \\ &\quad +\left(\frac{h_{W,i}}{C_{V,Wall}} - T_{s1}\right)\left(\frac{\partial (U_iA_i)_1}{\partial T_{s1}}\right)_{P_{s1},N_{z,s1,j}}\Bigg]\frac{1}{\left(\frac{\partial h_{s1}}{\partial T_{s1}}\right)_{P_{s1},N_{z,s1,j}}}, \\ &\quad i \in S_{N+1}\end{aligned} \tag{5.57}$$

The differential of $T_{I,s}$ must be written out using the interpolation relation described in equation 5.46. The differential will therefore give two terms per stream.

The derivatives with respect to the connecting node states

The composition through the coil is instantaneously updated, and all enthalpy equations depend on it. The first CV also depends on the inflowing enthalpy. The pressure is assumed linear through the coil, and all wall elements and CVs therefore depend on the pressure. The dependence is through the temperature function. A simple sketch in figure 5.7, should illustrate the various differentials.

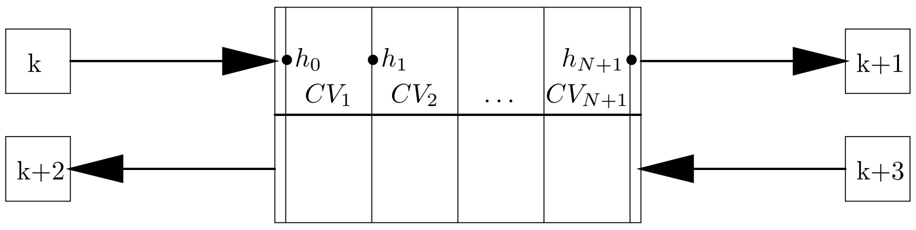


Figure 5.7: Reference system: Heat exchanger and four connecting nodes

Only the differentials for the stream between the k node and the k+1 node are written out. The second stream produces equal differentials, with different indexing. First the inlet CV, with the first dynamic state, h_1, is considered. The energy conservation equation is restated in Equation 5.58 with the correct indexes. The heat transfer coefficient is set constant. The relation $h_0 = h_k$ is substituted into the equation.

$$\left(\frac{dh_1}{dt}\right) = \frac{1}{V\rho_k}\Big(W(h_k - h_1) + U_1A_1(Tw_1 - T_{s,1})\Big) \tag{5.58}$$

The k node:

$$\begin{aligned}\left(\frac{\partial\left(\frac{dh_1}{dt}\right)}{\partial u_k}\right)_{\mathbf{N_{z,k}}} =& \frac{1}{V\rho_k}\Bigg[-V\left(\frac{dh_1}{dt}\right)\left(\frac{\partial\rho_k}{\partial u_k}\right)_{\mathbf{N_{z,k}}} + W\left(\frac{\partial h_k}{\partial u_k}\right)_{\mathbf{N_{z,k}}} \\ &+ (h_k - h_1)\left(\frac{\partial W}{\partial u_k}\right)_{\mathbf{N_{z,k}}} \\ &- U_1A_1\left(\frac{\partial T_{s,1}}{\partial T_k}\right)\left(\frac{\partial T_k}{\partial u_k}\right)_{\mathbf{N_{z,k}}} \\ &- U_1A_1\left(\frac{\partial T_{s,1}}{\partial P_k}\right)\left(\frac{\partial P_k}{\partial u_k}\right)_{\mathbf{N_{z,k}}}\Bigg]\end{aligned} \tag{5.59}$$

$$\begin{aligned}\left(\frac{\partial\left(\frac{dh_1}{dt}\right)}{\partial N_{z_i,k}}\right)_{u_k,N_{z_j,k}} =& \frac{1}{V\rho_k}\Bigg[-V\left(\frac{dh_1}{dt}\right)\left(\frac{\partial\rho_k}{\partial N_{z_i,k}}\right)_{u_k,N_{z_j,k}} \\ &+ W\left(\frac{\partial h_k}{\partial N_{z_i,k}}\right)_{u_k,N_{z_j,k}} \\ &+ (h_k - h_0)\left(\frac{\partial W}{\partial N_{z_i,k}}\right)_{u_k,N_{z_j,k}} \\ &- U_1A_1\left(\frac{\partial T_{s,1}}{\partial T_k}\right)\left(\frac{\partial T_k}{\partial N_{z_i,k}}\right)_{u_k,N_{z_j,k}} \\ &- U_1A_1\left(\frac{\partial T_{s,1}}{\partial P_k}\right)\left(\frac{\partial P_k}{\partial N_{z_i,k}}\right)_{u_k,N_{z_j,k}}\Bigg] \\ & i \in S_{NC} \quad j \in S_{NC,i}\end{aligned} \tag{5.60}$$

The k+1 node depends on both the dynamic states in the k node and the last CV state of the stream in the heat exchanger. The dependence in the flow is identical to the general case (as for simple valve flow in Section 5.1), and are not written out. The differential with respect to the last CV enthalpy must be written out. The only relation we need is the following:

$$\left(\frac{\partial\left(\frac{du_{k+1}}{dt}\right)}{\partial h_{Out}}\right)_{\dots} = \frac{W}{N_{z,k+1}} \tag{5.61}$$

The differentials for internal states (j), that is; not the inlet enthalpy, is written out below. Differentiate with respect to the inlet node. The heat exchanging temperature is interpolated in some linear manner.

First the composition is held constant:

$$\begin{aligned}\left(\frac{\partial\left(\frac{dh_j}{dt}\right)}{\partial u_k}\right)_{\mathbf{N_{z,k}}} =& \frac{1}{V\rho_{j-1}}\left[-V\left(\frac{dh_j}{dt}\right)\left(\frac{\partial \rho_{j-1}}{\partial u_k}\right)_{\mathbf{N_{z,k}}}\right.\\ &+ (h_{j-1}-h_j)\left(\frac{\partial W}{\partial u_k}\right)_{\mathbf{N_{z,k}}}\\ &\left.- UA\left(\frac{\partial T_s}{\partial P}\right)\left(\frac{\partial P_j}{\partial u_k}\right)_{\mathbf{N_{z,k}}}\right]\\ &j \in S_{N+1,1}\end{aligned} \tag{5.62}$$

Here the density changes as a result of the change in pressure and temperature. The temperature change is also through the change in pressure. Written out:

$$\begin{aligned}\left(\frac{\partial \rho_{j-1}}{\partial u_k}\right)_{\mathbf{N_{z,k}}} =& \left[\left(\frac{\partial \rho_{j-1}}{\partial T}\right)\left(\frac{\partial T_{j-1}}{\partial P}\right)+\left(\frac{\partial \rho_{j-1}}{\partial P}\right)\right]\left(\frac{\partial P_{j-1}}{\partial u_k}\right)_{\mathbf{N_{z,k}}},\\ &j \in S_{N+1,1}\end{aligned} \tag{5.63}$$

The temperature differential:

$$\begin{aligned}\left(\frac{\partial T_s}{\partial P}\right) =& \left(\frac{1+\alpha_{AT}}{2}\right)\left(\frac{\partial T_{j-1}}{\partial P}\right)+\left(\frac{1-\alpha_{AT}}{2}\right)\left(\frac{\partial T_j}{\partial P}\right)\\ &j \in S_{N+1,1}\end{aligned} \tag{5.64}$$

The temperature change is described as follows:

$$\left(\frac{\partial T}{\partial P}\right)_{h,\mathbf{N_z}} = -\frac{\left(\frac{\partial h}{\partial P}\right)_{T,\mathbf{N_z}}}{\left(\frac{\partial h}{\partial T}\right)_{P,\mathbf{N_z}}} \tag{5.65}$$

That is; the enthalpy change is required to be zero.

The compositional differentials take the same form. But the partial differentials of the properties must be written out.

$$\begin{aligned}\left(\frac{\partial T}{\partial N_{z_i}}\right)_{h,N_{z_j}} =& -\frac{1}{\left(\frac{\partial h}{\partial T}\right)_{P,\mathbf{N_z}}}\left(\left(\frac{\partial h}{\partial P}\right)_{T,\mathbf{N_z}}\left(\frac{\partial P}{\partial N_{z_i,k}}\right)_{N_{z,j}}\right.\\ &\left.+\left(\frac{\partial h}{\partial N_{z_i}}\right)_{P,T,N_{z_j}}\right),\\ &i \in S_{NC}, \quad j \in S_{NC,j}\end{aligned} \tag{5.66}$$

The density function is then a linear combination of its partial differentials and the P,T and direct change in composition.

The wall CV differentials are all in the form given in Equation 5.67.

$$\left(\frac{\partial\left(\frac{dhw_k}{dt}\right)}{\partial\gamma}\right) = -\frac{UA_s}{V_{CV}\rho_W}\left(\frac{\partial T_s}{\partial\gamma}\right) \tag{5.67}$$

Here γ is in the variable set $\{u, \mathbf{N_z}^T\}$.

5.3 Compressor

The compressor equations are given by Geankoplis (1993).

5.3.1 Algebraic flow relations

This section will present the equations used to describe the compressor.

Figure 5.8 shows a sketch of the compressor and shows how the subscripting is used. The subscript "s" will be used for the suction side, and the subscript "d" will be used for the discharge side.

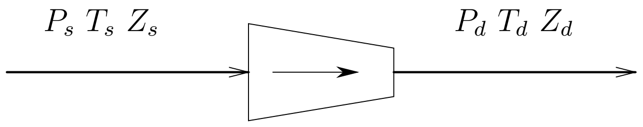


Figure 5.8: Compressor sketch

Some parameters used in the compressor model are defined in Equations 5.68 through 5.70.

$$\kappa = \frac{C_P}{C_V} \tag{5.68}$$

$$r = \frac{\kappa - 1}{\kappa\eta_P} \tag{5.69}$$

$$Z_{av} = \frac{Z_d + Z_s}{2} \tag{5.70}$$

The κ and r parameter are treated as constants, and are calculated only at the initial steady state. The η_P is the isentropic efficiency.

A compressor performance curve defined by the user, will be interpolated using cubic splines. The curve displays head, H [J/kg], as a function of flow, Q [m^3/h], for a given speed, N_{des} (design speed). An example of a compressor curve is given in Figure 5.9.

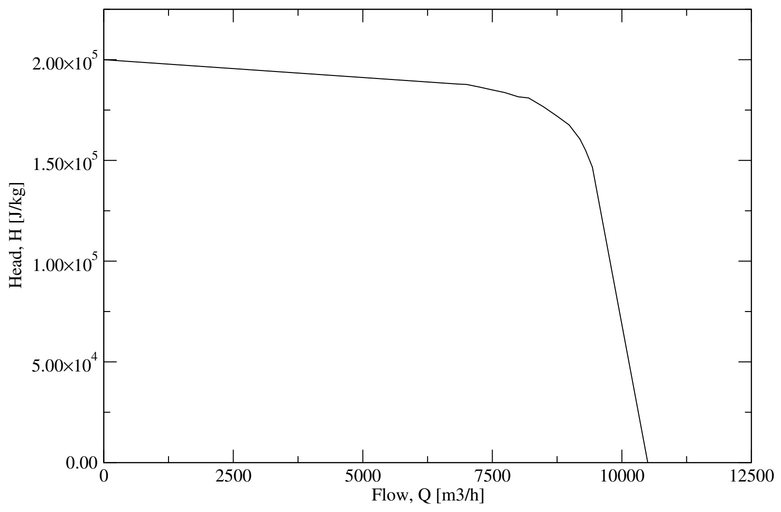


Figure 5.9: Compressor performance curve, C_{PC}

The head is given by the following equation:

$$H = \frac{RZ_{av}T_s}{Mwr}\left[\left(\frac{P_d}{P_s}\right)^r - 1\right] \tag{5.71}$$

The compressor performance curve, C_{PC}, is abstracted into equation 5.72.

$$\tilde{Q} = C_{PC}(\tilde{H}) \tag{5.72}$$

The flow, Q, is determined by interpolation of the performance curve. The fan laws, Equations 5.73 and 5.74, are used to extrapolate the curve to different compressor speeds.

$$\tilde{H} = H\frac{N_{des}^2}{N^2} \tag{5.73}$$

$$Q = \tilde{Q}\frac{N}{N_{des}} \tag{5.74}$$

The mass flow, W [kg/s], is given by equation 5.75.

$$W = \frac{Q\rho_s}{3600} \tag{5.75}$$

The power consumption, B [kW], is calculated as follows:

$$B = \frac{WH}{\eta_P 1000} \tag{5.76}$$

The discharge temperature, T_d [K], is given by equation 5.77.

$$T_d = T_s \left(\frac{P_d}{P_s} \right)^r \tag{5.77}$$

The compressor description assume adiabatic compression, and a constant polytopic efficiency is used. The flow is calculated from the curve data. The efficiency is typically described with curve data, and this should be allowed. Some of the compressor parameters are held constant. Both the average compression factor, Z, and the inlet heat capacity ratio, C_P/C_V, are held constant because it is difficult to describe their partial differentials. This was a necessity during the implementation and numerical testing of the compressor state variable differentials, to see that the analytical system Jacobian entries matched the numerically produced entries. It is believed that after finishing these tests, the parameters could be allowed to change, without affecting the simulation results.

5.3.2 Pressure - Flow - Linearization

Combining Equations 5.71, 5.72, 5.73, 5.74 and 5.75 the flow relation becomes:

$$W = \frac{\rho_s N}{3600 N_{des}} C_{PC} \left(H \frac{N_{des}^2}{N^2} \right) \tag{5.78}$$

The connecting nodes need mass flow differentials with respect to inlet and outlet pressure. Pressure dependence in Z_{av} are neglected.

$$\left(\frac{\partial W}{\partial P_d} \right)_{P_s,T_s} = \frac{\rho_s}{3600} \left(\frac{\partial C_{PC}(\tilde{H})}{\partial H} \right)_N \frac{N_{des}}{N} \left(\frac{\partial H}{\partial P_d} \right)_{P_s,T_s} \tag{5.79}$$

$$\left(\frac{\partial H}{\partial P_d} \right)_{P_s,T_s} = \frac{R Z_{av} T_s}{Mw P_d} \left(\frac{P_d}{P_s} \right)^r \tag{5.80}$$

$$\begin{aligned}\left(\frac{\partial W}{\partial P_s} \right)_{P_d,T_s} =& \frac{1}{3600} \left[\left(\frac{\partial C_{PC}(\tilde{H})}{\partial H} \right)_N \frac{N_{des}}{N} \left(\frac{\partial H}{\partial P_s} \right)_{P_d,T_s} \rho_s \right. \\ & \left. + Q \left(\frac{\partial \rho_s}{\partial P_s} \right)_{T_s} \right]\end{aligned} \tag{5.81}$$

$$\left(\frac{\partial H}{\partial P_s} \right)_{P_d,T_s} = - \frac{R Z_{av} T_s}{Mw P_s} \left(\frac{P_d}{P_s} \right)^r \tag{5.82}$$

The compressor performance curve differentials, C_{PC}, are to be given analytically from the spline parameters.

5.3.3 Analytical derivatives

Figure 5.10 gives a compressor connected to two nodes. This will be used for reference in the derivation of the compressor differentials.

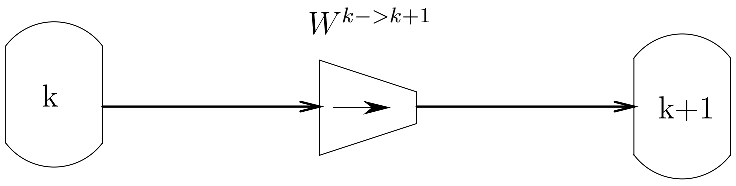


Figure 5.10: Compressor reference figure

Restate Equations 5.78, 5.71 and 5.77 with the new indexing. The superscript "$->k+1$" means sent to "k+1".

$$W^{k->k+1} = \frac{\rho_k N}{3600 N_{des}} C_{PC}\left(H \frac{N_{des}^2}{N^2}\right) \tag{5.83}$$

$$H = \frac{R Z_{av} T_k}{M w_k r}\left[\left(\frac{P_{k+1}}{P_k}\right)^r - 1\right] \tag{5.84}$$

$$T^{->k+1} = T_k \left(\frac{P_{k+1}}{P_k}\right)^r \tag{5.85}$$

The enthalpy leaving the compressor is given by the relation in Equation 5.86.

$$h^{->k+1} = h(T^{->k+1}, P_{k+1}, \mathbf{N_{z,k}}) \tag{5.86}$$

The parameters r, η and κ are still treated as constants. These parameters vary little and are difficult to differentiate without numerical perturbations.

Introduce an auxiliary partial differential:

$$\left(\frac{\partial C_{PC}}{\partial H}\right) = \left(\frac{\partial C_{PC}\left(H \frac{N_{des}^2}{N^2}\right)}{\partial H}\right)_N \tag{5.87}$$

The flow differentials:

Node k:

$$\begin{aligned}\left(\frac{\partial W^{k->k+1}}{\partial \gamma_k}\right)_{N,P_{k+1}} =& \frac{N}{3600 N_{des}} C_{PC}\left(\frac{\partial \rho_k}{\partial \gamma_k}\right) \\ &+ \frac{\rho_k N}{3600 N_{des}}\left(\frac{\partial C_{PC}}{\partial H}\right)\left(\frac{\partial H}{\partial \gamma_k}\right), \\ &\gamma \in \{T, P, \mathbf{N_z}^T\}\end{aligned} \tag{5.88}$$

$$\begin{aligned}\left(\frac{\partial H}{\partial T_k}\right)_{N,P_k,P_{k+1},\mathbf{N_{z,k}}} =& \frac{RZ_{av}}{Mw_k r}\left[\left(\frac{P_{k+1}}{P_k}\right)^r - 1\right] \\ &+ \frac{RT_k}{2Mw_k r}\left[\left(\frac{P_{k+1}}{P_k}\right)^r - 1\right]\left(\frac{\partial Z_k}{\partial T_k}\right)_{P_k,\mathbf{N_{z,k}}}\end{aligned} \tag{5.89}$$

$$\begin{aligned}\left(\frac{\partial H}{\partial P_k}\right)_{T_k,P_{k+1},\mathbf{N_{z,k}}} =& -\frac{RZ_{av}T_k}{Mw_k P_k}\left(\frac{P_{k+1}}{P_k}\right)^r \\ &+ \frac{RT_k}{2Mw_k r}\left[\left(\frac{P_{k+1}}{P_k}\right)^r - 1\right]\left(\frac{\partial Z_k}{\partial P_k}\right)_{T_k,\mathbf{N_{z,k}}}\end{aligned} \tag{5.90}$$

$$\begin{aligned}\left(\frac{\partial H}{\partial N_{z_i,k}}\right)_{N,T_k,P_k,P_{k+1},N_{z_j,k}} =& -\frac{RZ_{av}T_k}{(Mw_k)^2 r}\left[\left(\frac{P_{k+1}}{P_k}\right)^r - 1\right]\left(\frac{\partial Mw_k}{\partial N_{z_i,k}}\right)_{N_{z_j,k}} \\ &+ \frac{RT_k}{2Mw_k r}\left[\left(\frac{P_{k+1}}{P_k}\right)^r - 1\right]\left(\frac{\partial Z_k}{\partial N_{z_i,k}}\right)_{T_k,P_k,N_{z_j,k}}, \\ & i \in S_{NC},\ j \in S_{NC,i}\end{aligned} \tag{5.91}$$

We have:

$$Mw_k = \frac{\sum_{j=1}^{NC} N_{z_j,k} Mw_j}{N_{z,k}} \tag{5.92}$$

$$\begin{aligned}\left(\frac{\partial Mw_k}{\partial N_{z_i,k}}\right)_{N_{z_j,k}} =& \frac{Mw_i N_{z,k} - \sum_{l=1}^{NC} N_{z_l,k} Mw_l}{(N_{z,k})^2}, \\ & i \in S_{NC},\ j \in S_{NC,i}\end{aligned} \tag{5.93}$$

The k+1 node:

$$\left(\frac{\partial W^{k->k+1}}{\partial P_{k+1}}\right)_{N,T_k,P_k,\mathbf{N_{z,k}}} = \frac{\rho^k N}{3600 N_{des}}\left(\frac{\partial C_{PC}}{\partial H}\right)\left(\frac{\partial H}{\partial P_{k+1}}\right)_{N,T_k,P_k,\mathbf{N_{z,k}}} \tag{5.94}$$

$$\begin{aligned}\left(\frac{\partial H}{\partial P_{k+1}}\right)_{T_k,P_k,\mathbf{N_{z,k}}} =& \frac{RZ_{av}T_k}{Mw_k P_{k+1}}\left(\frac{P_{k+1}}{P_k}\right)^r \\ &+ \frac{RT_k}{2Mw_k r}\left[\left(\frac{P_{k+1}}{P_k}\right)^r - 1\right]\left(\frac{\partial Z_{k+1}}{\partial P_{k+1}}\right)_{T_{k+1},\mathbf{N_{z,k}}}\end{aligned} \tag{5.95}$$

The enthalpy differential:

The k node:

$$\left(\frac{\partial h^{->k+1}}{\partial u^k}\right) = \left(\frac{\partial h^{->k+1}}{\partial T^{->k+1}}\right)\left[\left(\frac{\partial T^{->k+1}}{\partial T^k}\right)\left(\frac{\partial T^k}{\partial u^k}\right) + \left(\frac{\partial T^{->k+1}}{\partial P^k}\right)\left(\frac{\partial P^k}{\partial u^k}\right)\right] \tag{5.96}$$

$$\left(\frac{\partial h^{->k+1}}{\partial N_{z,i}^k}\right) = \left(\frac{\partial h^{->k+1}}{\partial T^{->k+1}}\right)\left[\left(\frac{\partial T^{->k+1}}{\partial T^k}\right)\left(\frac{\partial T^k}{\partial N_{z,i}^k}\right) + \left(\frac{\partial T^{->k+1}}{\partial P^k}\right)\left(\frac{\partial P^k}{\partial N_{z,i}^k}\right)\right] + \left(\frac{\partial h^{->k+1}}{\partial N_{z,i}^k}\right) \tag{5.97}$$

The k+1 node:

$$\left(\frac{\partial h^{->k+1}}{\partial u^{k+1}}\right) = \left[\left(\frac{\partial h^{->k+1}}{\partial T^{->k+1}}\right)\left(\frac{\partial T^{->k+1}}{\partial P^{k+1}}\right) + \left(\frac{\partial h^{->k+1}}{\partial P^{k+1}}\right)\right]\left(\frac{\partial P^{k+1}}{\partial u^{k+1}}\right) \tag{5.98}$$

$$\left(\frac{\partial h^{->k+1}}{\partial N_{z,i}^{k+1}}\right) = \left[\left(\frac{\partial h^{->k+1}}{\partial T^{->k+1}}\right)\left(\frac{\partial T^{->k+1}}{\partial P^{k+1}}\right) + \left(\frac{\partial h^{->k+1}}{\partial P^{k+1}}\right)\right]\left(\frac{\partial P^{k+1}}{\partial N_{z,i}^{k+1}}\right) \tag{5.99}$$

The speed differential of the mass flow:

$$\left(\frac{\partial W^{k->k+1}}{\partial N}\right) = \frac{\rho^k}{3600}\left(\frac{C_{PC}}{N_{des}} - 2\left(\frac{\partial C_{PC}}{\partial H}\right)\frac{HN_{des}}{N^2}\right) \tag{5.100}$$

5.4 Pump

The pump model is not included in the LNG plant as it is described in the full LNG system. If/when the column models are included, the model will be needed. The model uses a fixed efficiency and a simple curve to predict the flow. The efficiency should have an option, allowing it to interpolate curve data.

5.4.1 Algebraic flow relations

A centrifugal description is used for the pump. Only simplified rules as described by Geankoplis (1993) is used.

The pump head, H, is defined in Equation 5.101.

$$H = \frac{\Delta P}{\rho g}[=]\ m \tag{5.101}$$

The fan laws is given in Equations 5.102 and 5.103.

$$\frac{q_1}{q_2} = \frac{N_1}{N_2} \tag{5.102}$$

$$\frac{H_1}{H_2} = \frac{q_1^2}{q_2^2} \tag{5.103}$$

The relation between mass flow and volume flow is given in Equation 5.104.

$$q = \frac{W}{\rho} \tag{5.104}$$

Here W is the mass flow.

The simple volume flow function used in the model is given in Equation 5.105.

$$q = q_{Max,Ref}\sqrt{1 - \frac{H}{H_{Max,Ref}}} \tag{5.105}$$

Restating Equations 5.102 and 5.103 in Equation 5.106 with new subscripts.

$$q_{Max} = q_{Max,Ref}\frac{N}{N_{Ref}}, \quad H_{Max} = H_{Max,Ref}\frac{N^2}{N_{Ref}^2} \tag{5.106}$$

Equation 5.101, 5.104, 5.105 and 5.106 are combined to Equation 5.107.

$$W = \frac{q_{Max,Ref}N\rho}{N_{Ref}}\sqrt{1 - \frac{\Delta P N_{Ref}^2}{\rho g H_{Max,Ref}N^2}} \tag{5.107}$$

Assuming the density does not vary too much we can transform the equation to Equation 5.108.

$$W_{Simp} = \frac{W_{Max,Ref}N}{N_{Ref}}\sqrt{1 - \frac{\Delta P N_{Ref}^2}{\Delta P_{Max,Ref}N^2}} \tag{5.108}$$

The mass flow is now only a function in pressure difference and rpm. Using the simplified description is a configuration choice. The default is using the equation without the simplification.

When no performance data is present, the parameters of Equation 5.107 can be estimated from the initial flow and difference pressure, making an assumption on the maximum delta pressure. If the rpm scaling is to make sense, the reference rpm must be given. The maximum delta P is guessed to 3x the initial delta P. The pump head-flow relation is illustrated in Figure 5.11.

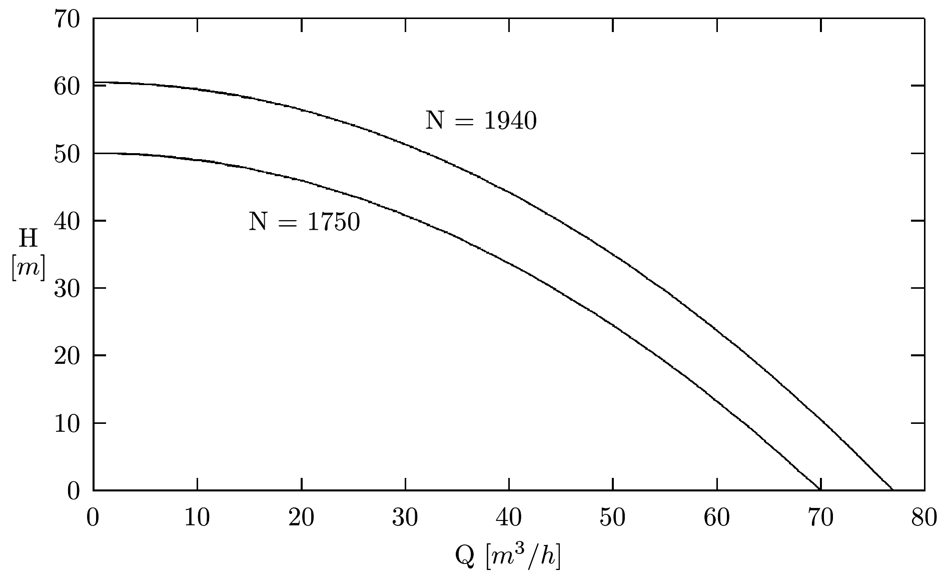


Figure 5.11: Pump characteristics, showing two curves

The energy consumed by the pump is added to the flowing fluid. The pump operates at a constant efficiency, e.

$$\Delta h = \frac{\Delta P}{\rho e}, \quad 0 < e \leq 1 \tag{5.109}$$

The Δh given by Equation 5.109 is added to the inlet enthalpy to give the outlet enthalpy from the pump.

The energy consumption by the pump is then given by Equation 5.110.

$$E = W \Delta h \ [J/s] \tag{5.110}$$

5.4.2 Pressure - Flow - Linearization

Equation 5.108 is linearized in pressure. The result is given in Equation 5.111.

$$\left(\frac{\partial W_{Simp}}{\partial \Delta P}\right) = -\frac{W_{Max,Ref}}{2\sqrt{1 - \frac{\Delta P N_{Ref}^2}{\Delta P_{Max,Ref} N^2}}} \frac{N_{Ref}}{\Delta P_{Max,Ref} N} \tag{5.111}$$

Linearizing the original flow Equation 5.107, assuming incompressible flow, gives Equation 5.112.

$$\left(\frac{\partial W}{\partial \Delta P}\right) = -\frac{q_{Max,Ref}}{2\sqrt{1 - \frac{H N_{Ref}^2}{H_{Max,Ref} N^2}}} \frac{N_{Ref}}{g H_{Max,Ref} N} \tag{5.112}$$

5.4.3 Analytical derivatives

The differentials for the pump will be written out using the reference system given in Figure 5.12.

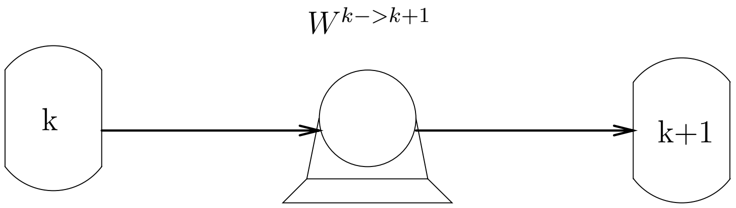


Figure 5.12: Pump reference system for the derivatives

Restating Equations 5.107 and 5.109 using indexes reflects the reference system in Figure 5.12. The result is shown in Equations 5.113 and 5.114.

$$W^{k->k+1} = \frac{q_{Max,Ref} N \rho^k}{N_{Ref}} \sqrt{1 - \frac{(P^{k+1} - P^k) N_{Ref}^2}{\rho^k g H_{Max,Ref} N^2}} \tag{5.113}$$

$$h^{->k+1} = h^k + \frac{P^{k+1} - P^k}{\rho^k e} \tag{5.114}$$

The flow equation

Start by defining three auxiliary partial differential.

$$\begin{aligned} \left(\frac{\partial W^{k->k+1}}{\partial \rho^k}\right)_{[a1]} &= \frac{q_{Max,Ref} N}{N_{Ref}} \sqrt{1 - \frac{(P^{k+1} - P^k) N_{Ref}^2}{\rho^k g H_{Max,Ref} N^2}} \\ &+ \frac{q_{Max,Ref}}{2\sqrt{1 - \frac{(P^{k+1} - P^k) N_{Ref}^2}{\rho^k g H_{Max,Ref} N}}} \frac{(P^{k+1} - P^k) N_{Ref}}{\rho^k g H_{Max,Ref} N^2} \end{aligned} \tag{5.115}$$

$$\left(\frac{\partial W^{k->k+1}}{\partial P^k}\right)_{[a2]} = \frac{q_{Max,Ref}}{2\sqrt{1 - \frac{(P^{k+1} - P^k) N_{Ref}^2}{\rho^k g H_{Max,Ref} N^2}}} \frac{N_{Ref}}{g H_{Max,Ref} N} \tag{5.116}$$

$$\left(\frac{\partial W^{k->k+1}}{\partial P^{k+1}}\right)_{[a3]} = -\frac{q_{Max,Ref}}{2\sqrt{1 - \frac{(P^{k+1} - P^k) N_{Ref}^2}{\rho^k g H_{Max,Ref} N^2}}} \frac{N_{Ref}}{g H_{Max,Ref} N} \tag{5.117}$$

Using these auxiliary differentials, the flow equation differentials are written out. First the differentials with respect to the state variables in node k are written out producing Equations 5.118 and 5.119.

$$\begin{aligned} \left(\frac{\partial W^{k->k+1}}{\partial u^k}\right)_{P^{k+1}, \mathbf{N_z^k}} &= \left(\frac{\partial W^{k->k+1}}{\partial \rho^k}\right)_{[a1]} \left(\frac{\partial \rho^k}{\partial u^k}\right)_{\mathbf{N_z^k}} \\ &+ \left(\frac{\partial W^{k->k+1}}{\partial P^k}\right)_{[a2]} \left(\frac{\partial P^k}{\partial u^k}\right)_{\mathbf{N_z^k}} \end{aligned} \tag{5.118}$$

$$\left(\frac{\partial W^{k->k+1}}{\partial N_{z,i}^{k}}\right)_{T^k,P^k,P^{k+1},N_{z,i}^{k}} = \left(\frac{\partial W^{k->k+1}}{\partial \rho^k}\right)\left(\frac{\partial \rho^k}{\partial N_{z,i}}\right)_{T,P,N_{z,i}^{k}} \tag{5.119}$$

Differentials in node k+1 state variables are written out in Equations 5.120 and 5.121.

$$\left(\frac{\partial W^{k->k+1}}{\partial u^{k+1}}\right)_{\mathbf{N_z^k}} = \left(\frac{\partial W^{k->k+1}}{\partial P^{k+1}}\right)_{[a3]}\left(\frac{\partial P^{k+1}}{\partial u^{k+1}}\right)_{\mathbf{N_z^k}} \tag{5.120}$$

$$\left(\frac{\partial W^{k->k+1}}{\partial N_{z,i}^{k+1}}\right)_{u^{k+1},N_{z,j}^{k}} = \left(\frac{\partial W^{k->k+1}}{\partial P^{k+1}}\right)_{[a3]}\left(\frac{\partial P^{k+1}}{\partial N_{z,i}^{k+1}}\right)_{u^{k+1},N_{z,j}^{k}} \tag{5.121}$$

The energy equation

The partial differentials of the energy equations are written out in Equation 5.122 and 5.123.

Node k:

$$\left(\frac{\partial h^{->k+1}}{\partial u^k}\right)_{N_{z,i}^k} = \left(\frac{\partial h^k}{\partial u^k}\right)_{N_{z,i}^k} - \frac{1}{\rho^k e}\left(\frac{\partial P^k}{\partial u^k}\right)_{N_{z,i}^k} - \frac{P^{k+1}-P^k}{(\rho^k)^2 e}\left(\frac{\partial \rho^k}{\partial u^k}\right)_{N_{z,i}^k} \tag{5.122}$$

$$\begin{aligned}\left(\frac{\partial h^{->k+1}}{\partial N_{z,i}^k}\right)_{u^k,N_{z,j}^k} &= \left(\frac{\partial h^k}{\partial N_{z,i}^k}\right)_{u^k,N_{z,j}^k} - \frac{1}{\rho^k e}\left(\frac{\partial P^k}{\partial N_{z,i}^k}\right)_{u^k,N_{z,j}^k} \\ &- \frac{P^{k+1}-P^k}{(\rho^k)^2 e}\left(\frac{\partial \rho^k}{\partial N_{z,i}^k}\right)_{u^k,N_{z,j}^k}\end{aligned} \tag{5.123}$$

Node k+1:

$$\left(\frac{\partial h^{->k+1}}{\partial u^{k+1}}\right)_{N_{z,i}^{k+1}} = \frac{1}{\rho^k e}\left(\frac{\partial P^{k+1}}{\partial u^{k+1}}\right)_{N_{z,i}^{k+1}} \tag{5.124}$$

$$\left(\frac{\partial h^{->k+1}}{\partial N_{z,i}^{k+1}}\right)_{u^{k+1},N_{z,j}^{k+1}} = \frac{1}{\rho^k e}\left(\frac{\partial P^{k+1}}{\partial N_{z,i}^{k+1}}\right)_{u^{k+1},N_{z,j}^{k+1}} \tag{5.125}$$

5.5 Liquid expander

The hydraulic turbine is treated in a simple manner. The flow dynamics are the same as for a valve. The expander is given an constant isentropic efficiency, γ. To improve the description of the hydraulic turbine, the possibility to describe both the efficiency and the flow using curve data, should be supported.

$$h_{Out} = \gamma h_{Out}|_{S_{In},P_{Out}} + (1-\gamma)h_{In} \tag{5.126}$$

$$\Delta h = h_{In} - h_{Out} \tag{5.127}$$

The energy produced are given by the Equation 5.128.

$$E = \Delta h W \quad [J/s] \tag{5.128}$$

This means that the unit model needs to evaluate the entropy at the inlet and do an SP-Flash, using the outlet pressure. The outlet enthalpy is thereafter evaluated.

5.5.1 Pressure - Flow - Linearization

The flow dynamics are identical to the valve flow dynamics. See Section 5.1.2.

5.5.2 Analytical derivatives

The derivatives are related to the reference Figure 5.13.

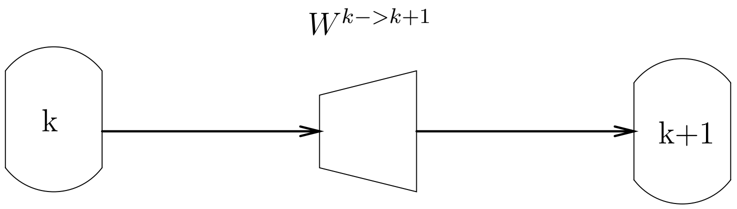


Figure 5.13: Index reference for the derivatives of the turbine

The flow equation is identical to the valve equation, and is not written out (see Section 5.1.3). The algebraic enthalpy relation is written out in Equation 5.129 using indexing as in figure 5.13.

$$h^{->k+1} = \gamma h|_{S^k,P^{k+1}} + (1-\gamma)h^k \tag{5.129}$$

Equation 5.129 is first differentiated with respect to the internal energy of node k. This gives Equation 5.130.

$$\left(\frac{\partial h^{->k+1}}{\partial u^k}\right) = \gamma\left(\frac{\partial h|_{S^k,P^{k+1}}}{\partial u^k}\right) + (1-\gamma)\left(\frac{\partial h^k}{\partial u^k}\right) \tag{5.130}$$

The first partial differential on the Right Hand Side, RHS, of Equation 5.130 needs some consideration. It is written out in Equation 5.131.

$$\left(\frac{\partial h|_{S^k,P^{k+1}}}{\partial u^k}\right) = \left(\frac{\partial h|_{S^k,P^{k+1}}}{\partial T^k}\right)\left(\frac{\partial T^k}{\partial S^k}\right)_{|\ Out}\left(\frac{\partial S^k}{\partial u^k}\right)_{|\ In} \tag{5.131}$$

The partial of s with respect to u is given in Equation 5.132.

$$\left(\frac{\partial S}{\partial u}\right)_{|\ In} = \left(\frac{\partial S}{\partial T}\right)\left(\frac{\partial T}{\partial u}\right) + \left(\frac{\partial S}{\partial P}\right)\left(\frac{\partial P}{\partial u}\right) \tag{5.132}$$

Differentiate Equation 5.130 with respect to node k composition in Equation 5.133.

$$\left(\frac{\partial h^{->k+1}}{\partial N^k_{z,i}}\right)_{u^k,N^k_{z,j}} = \gamma\left(\frac{\partial h|_{S^k,P^{k+1}}}{\partial N^k_{z,i}}\right)_{u^k,N^k_{z,j}} + (1-\gamma)\left(\frac{\partial h^k}{\partial N^k_{z,i}}\right)_{u^k,N^k_{z,j}} \tag{5.133}$$

The first RHS partial differential of Equation 5.133 is written out in Equation 5.134.

$$\left(\frac{\partial h|_{S^k,P^{k+1}}}{\partial N^k_{z,j}}\right) = \left(\frac{\partial h|_{S^k,P^{k+1}}}{\partial T_{Out}}\right)\left(\frac{\partial T_{Out}}{\partial N^k_{z,j}}\right) + \left(\frac{\partial h|_{S^k,P^{k+1}}}{\partial N^k_{z,j}}\right) \tag{5.134}$$

The partial of s with respect to $N_{z,j}$, at the inlet is given in Equation 5.135.

$$\left(\frac{\partial S}{\partial N_{z,j}}\right) = \left(\frac{\partial S}{\partial T}\right)\left(\frac{\partial T}{\partial N_{z,j}}\right) + \left(\frac{\partial S}{\partial P}\right)\left(\frac{\partial P}{\partial N_{z,j}}\right) + \left(\frac{\partial S}{\partial N_{z,j}}\right) \tag{5.135}$$

The partial of s with respect to $N_{z,j}$, at the outlet is given by Equation 5.136.

$$\left(\frac{\partial S_{Out}}{\partial N_{z,j}}\right) = \left(\frac{\partial S_{Out}}{\partial T_{Out}}\right)\left(\frac{\partial T_{Out}}{\partial N_{z,j}}\right) + \left(\frac{\partial S_{Out}}{\partial N_{z,j}}\right) \tag{5.136}$$

Equating Equations 5.135 and 5.136, the temperature differential at the outlet is determined. The differential in equation 5.134 can then be calculated.

Differentiate Equation 5.130 with respect to node k+1 state variables in Equation 5.137.

$$\left(\frac{\partial h^{->k+1}}{\partial u^{k+1}}\right) = \gamma\left[\left(\frac{\partial h_{Out}}{\partial P_{Out}}\right) - \left(\frac{\partial h_{Out}}{\partial T_{Out}}\right)\frac{\left(\frac{\partial S_{Out}}{\partial P_{Out}}\right)}{\left(\frac{\partial S_{Out}}{\partial T_{Out}}\right)}\right]\left(\frac{\partial P^{k+1}}{\partial u^{k+1}}\right) \tag{5.137}$$

$$\left(\frac{\partial h^{->k+1}}{\partial N^k_{z,i}}\right) = \gamma\left[\left(\frac{\partial h_{Out}}{\partial P_{Out}}\right) - \left(\frac{\partial h_{Out}}{\partial T_{Out}}\right)\frac{\left(\frac{\partial S_{Out}}{\partial P_{Out}}\right)}{\left(\frac{\partial S_{Out}}{\partial T_{Out}}\right)}\right]\left(\frac{\partial P^{k+1}}{\partial N^k_{z,i}}\right) \tag{5.138}$$

Chapter 6

Modelling of Controller Units

The low level controllers are PI controllers, and the input/output of the general controller is given in Figure 6.1. The controller equations are given in Section 6.1.

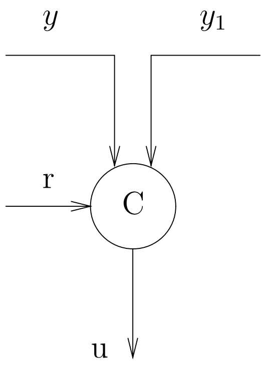


Figure 6.1: The general controller

The controller can take two measurement signals as input. In the case when two signals are defined, a simple algebraic relation between the two signals must be defined. The usual case, is a difference between the two signals.

The reference signal is given explicitly, or it can come from another controller. That is; controllers in cascade are supported.

6.1 Basic controller equations

The traditional PID controller has the following form, in the time domain.

$$u(t) = K_P \left\{ e_P(t) + \frac{1}{T_I} e_I^0(t) + T_D e_D(t) \right\} \tag{6.1}$$

Were:

$$e_P(t) = r - y \tag{6.2}$$

$$e_I^0(t) = \int e_P(t) dt \tag{6.3}$$

$$e_D(t) = -\left(\frac{dy}{dt}\right) \tag{6.4}$$

Here the reference value is denoted r, and the measured value is denoted y.

The controller used in this application is restricted to $T_D = 0$.

After redefining the integral term, the control law is described by Equation 6.5.

$$u(t) = K_P e_P(t) + e_I(t), \quad e_I(t) = \frac{K_P}{T_I} e_I^0(t) \tag{6.5}$$

The dynamic variable, e_I, with the differential given by Equation 6.6, is introduced.

$$\left(\frac{de_I}{dt}\right) = \frac{K_P}{T_I}(r - y) \tag{6.6}$$

6.1.1 Analytical derivatives

The derivatives for the integral part have the form of Equation 6.7, for all dynamic variables, x.

$$\left(\frac{\partial \left(\frac{de_I}{dt}\right)}{\partial x}\right) = -\frac{K_P}{T_I}\left(\frac{\partial y_i}{\partial x}\right) \tag{6.7}$$

Differentials for the controller output are also needed, since that affect some state variables.

$$\left(\frac{\partial u}{\partial x}\right) = K_P \left(\frac{\partial e_P}{\partial x}\right) + \left(\frac{\partial e_I}{\partial x}\right) \tag{6.8}$$

Here again x denotes arbitrary dynamic variables. The differentials of the proportional part is the same as for Equation 6.6, only scaled by a constant. The differential of the integral part is given in Equation 6.9.

$$\left(\frac{\partial e_I}{\partial x}\right) = \begin{cases} 0, & x \neq e_I \\ 1, & x = e_I \end{cases} \tag{6.9}$$

The derivative part of the controller, is given even though it is not used.

$$\left(\frac{\partial e_D}{\partial x}\right) = -\left(\frac{\partial\left(\frac{dy}{dt}\right)}{\partial x}\right) \tag{6.10}$$

These terms are generally not available, and must be determined numerically if they are to be used. This is one reason for neglecting them in this work.

6.2 Control of the LNG plant

The main objective of the LNG plant is to produce LNG at a minimal cost, and at the same time maintain stable operation. This should also be the objective for the control system. To achieve this an optimization layer must be added to the control system. This is not the focus of this thesis, and only low level PI controllers are available for the control layer. The objective for the control layer must therefore be to maintain stable operation, and to produce LNG.

During the production of LNG, the inflow of the NG must balance the MCR loops, in order for the MCR loops to remove the necessary energy. The energy removal from the NG stream must also be balanced on all three MCR loops. The following controllers are therefore added:

- Level controllers

  Level control of separation tanks in the system must be added, preventing single phase and an invalid simulation state.

- Pressure control

  The compressor speed is manipulated to control the compressor suction pressure. Control of other pressures (valve manipulation) is added to avoid drifting pressure. This control is not vital and is therefore not shown in the overview of the control structure in Figure 6.2

- Flow control to maintain the flows in all loops, and simplify the temperature controllers

  Flow controllers for all streams must be added, otherwise the flows will start to drift, when the node pressures change. The controllers are shown in Figure 6.2. All the temperature controllers send their output to a flow controller. These flow controllers are not drawn in order to avoid an overcrowded figure.

  Adding flow controllers, as slaves of the temperature controllers, also ease the tuning of temperature controllers.

- Temperature control to maintain the outlet temperature of the LNG, and to distribute the energy removal onto the three MCR streams

> *The close relationship between the flow of the cooling refrigerants and the outlet temperatures of the tube streams is used to control the NG and MCR temperatures. If the NG gets too warm, the shell stream is increased and vice versa.*
>
> *The outlet NG temperature of the* ***25-HX-102*** *heat exchanger is controlled by manipulating the flow of the NG. The boundary conditions for the NG stream therefore has to be pressure. This temperature is held constant by manipulating a valve after* ***25-HX-102, V25-155****. This dynamic are quite slow, and exhibit approximately first order dynamics with a time constant of ten to eleven minutes.*
>
> *The flow in the MCR2 loop is used to control the inlet temperature to* ***25-HX-102*** *of MCR3.*
>
> *The pre-cooling consists of four different heat-exchangers. Two are used to cool the NG, and two are used to cool MCR2 and MCR3 before they enter the spiral wound heat exchangers,* ***25-HX-101*** *and* ***25-HX-102****. The shell flow of* ***25-HG-101-I*** *and* ***25-HG-101-II*** *is used to control the intermediate temperatures of the NG in* ***N25-139*** *and* ***25-VD-107****. The shell flow of* ***25-HG-102-I*** *is held constant while the shell flow of* ***25-HG-102-II*** *is used to control the temperature of MCR3 in* ***N25-134****.*

The procedure applied for finding usable controller parameters is as follows.

- The system is configured and simulated to a steady state using only level control on the separator tanks.
- Step in compressor speed is used to get settings for the suction pressure controllers.
- Step in flow is used to get gain and integral time for the flow controllers.
- Step in flow produce controller parameters for the pressure controllers.
- Finally the temperature controller parameters are determined based on the step response in the manipulated flows. This process is iterative, since the system changes when a controller is introduced. Some parameters must therefore be retuned.

The initial steady state is used as the set point for the controllers.

It must be emphasized that the set of controller parameters, used in the MFCP simulation, is not optimal, but rather conservative parameters stabilizing the simulations. To produce the controller gains and integral times, simple shortcut rules are used.

Figure 6.2 shows the main control loops implemented on the system. It must be emphasized that no consideration with respect to the best control structure is performed. The selected control structure is applied to keep the simulator running. For further details see Appendix C.

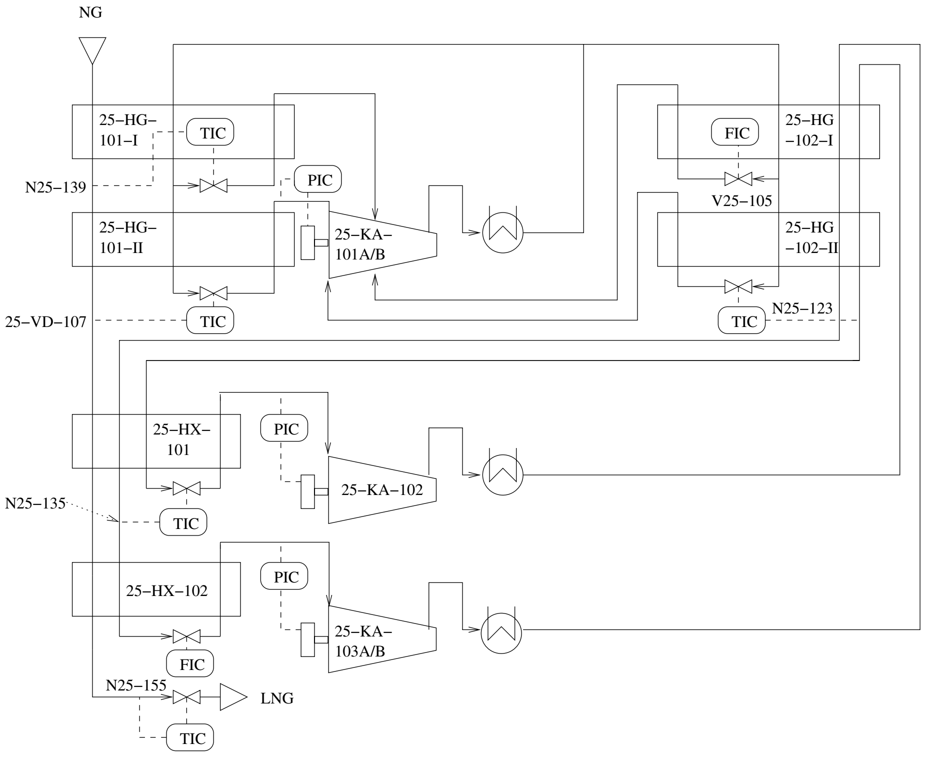


Figure 6.2: The Mixed Fluid Cascade Process (MFCP)

Chapter 7

Simulation Results

For simplicity, the system simulated with conservation of internal energy (U) and mass (in fixed volume (V)), will be referred as *the UV system* or *the UV formulation*. *UV integration* then implies Rosenbrock integration of the UV system. The other system, defined by conservation of enthalpy (H) and mass, requiring a HP flash is referred to as *the HP system* or *the HP formulation*. *HP integration* therefore implies split integration, where the pressure is integrated fully implicitly with an Euler scheme, and the composition and enthalpy states are integrated with fixed pressures and flows.

This chapter will present three simulation cases, where the first is a sub-system of the larger second system. The second system is an entire liquefaction process for natural gas. The entire system is not integrated with an acceptable real-time ratio, using the freeware integration codes, DVODPK, DASPK and LSODES. It is therefore of interest to see them perform in a simpler case. The smaller case also makes the comparison simpler.

The third case is a closer look at how the UV integration performs when there is change in phase from pure liquid to two-phase within a node.

A code profiling is executed to see which part of the code is the most demanding on the CPU. The result of the profiling indicates where the code is spending time, and therefore where it is possible to improve the overall performance. The result will therefore help in the further development of the simulator.

The last section of this chapter discusses possibilities for further developing the simulator.

All simulations are executed on the same hardware architecture. The PC used is a Zepto Z-note 4100 with a 1.7Ghz Intel Pentium M (Centrino) processor and 1GB of DDR ram. The Microsoft Windows XP platform is used.

The detailed process design for the LNG plant is given by the Norwegian oil company, Statoil. The general plant concept is shown in Foerg et al. (1998). The initial steady state is based on data provided by Statoil, and the simulated values are therefore to some extent normalized to hide the actual values.

All models are first built as a set of sub-models that are simulated to a steady state before aggregated into the entire flowsheet models. The flowsheet model is simulated to a steady state, and then saved. The simulations presented here are initialised from these steady state models. The steady state values for the dynamic state variables are used to produce a consistent set of initial values.

The HP steady state was used to be able to simulate from the same steady state, for both the UV and HP models. The UV models where then initialised from enthalpy, pressure and composition.

7.1 Test case - Sub LNG Plant

A simple case will be used to test the two solution strategies, before applying them on the entire LNG plant. A subsection of the LNG plant is chosen for convenience. The naming of the process units will therefore coincide with the naming given in Appendix C. The flow sheet of the test case is given in Figure 7.1.

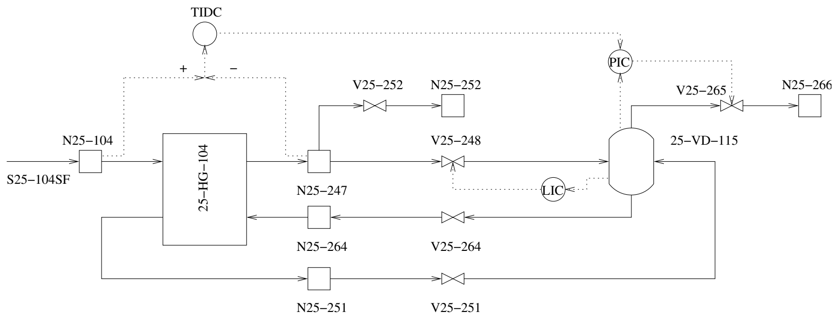


Figure 7.1: Test Case

Figure 7.1 shows a two stream heat exchanger, **25-HG-104**, a flash tank, **25-VD-115**, four dynamic nodes, **N25-104**, **N25-247** ,**N25-251**, **N25-264**, two static pressure nodes, **N25-252**, **N25-266**, and a static flow source, **S25-104SF**. It also shows five valves, and three PI controllers. The level controller, **LC**, of **25-VD-115** acts on the inlet valve **V25-248**. The pressure controller, **PIC**, controls the pressure of **25-VD-115**, manipulating **V25-265**. The reference signal is supplied by the temperature difference controller, **TIDC**. The **TIDC** will maintain a constant temperature difference over **25-HG-104**, for the stream between **N25-104** and **N25-247**.

It can be seen that the liquid flow from **25-VD-115** is driven by the static pressure of the liquid column in and under the flash tank.

To test the models, a temperature step change of 0.25 ^{o}C will be applied to the inlet stream, **S25-104SF**.

7.1.1 Simulation results - Sub LNG system

A sub-system of the LNG plant is chosen and simulated with a step on the inlet temperature. The flowsheet of this sub-system is given in Figure 7.1. This system is partly chosen because phase transition in/discontinuities in any of the state or state differentials not will cause integration difficulties. The simulation will only have time defined discontinuities.

The system is simulated using the HP formulation, and using the UV formulation. The UV formulation is simulated using four different integration routines/methods. The DVODPK, DASPK and LSODES codes, referenced in Section 3.4, are used. The codes produce the same results, given the same tolerance criteria, and therefore only the result of the DVODPK is presented. In addition to these three codes, the self implemented 1-stage Rosenbrock method is used to simulate the UV system.

The HP and UV Rosenbrock simulation is done using a fixed time step of 1 second. All simulations are sampled every second, and the equation set is allowed to change after being sampled. The integration must therefore be restarted every second.

The error tolerance for DVODPK, DASPK and LSODES are set loose, to 10^{-5} (absolute) and 10^{-4} (relative). The split HP integration and the Rosenbrock integration have no error control.

The system is simulated using SRK thermodynamics, and the initial steady state is defined by Tables 7.1, 7.2, 7.3 and 7.4.

Table 7.1: Initial node/flash proprieties

Name	Volume	Pressure	Temperature	Mode
	$[m^3]$	$[Bar]$	$[°C]$	
N25-104	4.00	20.34	10.00	DYNAMIC
N25-247	2.00	19.84	4.20	DYNAMIC
N25-251	2.00	8.47	7.08	DYNAMIC
N25-264	2.00	8.72	-6.79	DYNAMIC
25-VD-115	4.00	8.22	-6.79	DYNAMIC
N25-252	x	19.60	x	STATIC
N25-266	x	7.00	x	STATIC
S25-104SF	x	20.34	10.00	STATIC

Tables 7.1 and 7.2 respectively show the initial state of the pressure models and the flow models in this test case. A column is added to both tables to indicate if the mode of the unit models is STATIC or DYNAMIC. Static nodes are only a pressure boundary condition if, as in this case, flow is only entering. The **S25-104SF** is a source flow, defined by composition, pressure, temperature and flow.

Table 7.2: Initial flow properties

Name	Flow	Mode
	$[kg/s]$	
S25-104SF	141.82	STATIC
V25-252	133.30	DYNAMIC
V25-248	8.52	DYNAMIC
V25-264	10.46	DYNAMIC
V25-265	8.52	DYNAMIC

Table 7.3 gives the composition of the source flow, **S25-104SF**.

Table 7.3: Composition for **S25-104SF**

Name	Mass fraction $[kg/kg]$
Methane	3.88e-4
Ethane	5.46e-1
Propane	4.50e-1
i-Butane	2.52e-3
n-Butane	1.09e-3

A sub-optimal, but stable, set of parameters is supplied to the controllers. The parameters are given in Table 7.4.

Table 7.4: Controller parameters

Name	Proportional action	Integral time [sec]	Set point
TIDC	-1.00	100.00	5.8
PIC	-15.00	2.00	–
LIC	250.00	10.00	0.4

The HP system will have 46 ODE states and 5 pressure states. The heat exchanger, **25-HG-104**, will have 33 ODE states, 11 for each stream, and 11 for the wall. The four nodes will each have an enthalpy state and a pressure state. The flash tank, **25-VD-115**, will have an enthalpy, a pressure and five holdup states. The controllers have one ODE state for the integral time each. For each function call, 28 HP-Flash calls are executed.

The UV formulation will have 66 ODE states. 33 states are in the heat exchanger, and 3 states in the controller. Every node and flash tank has 6 ODE states. For each function call, 5 UV- and 23 HP-flash calls are executed.

Figure 7.2 shows the initial state of the heat exchanger.

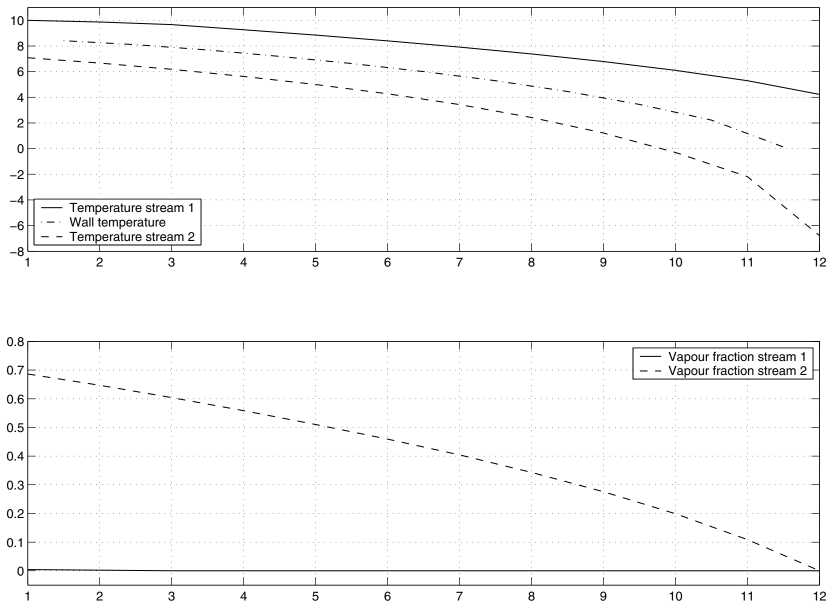


Figure 7.2: Initial heat exchanger, **25-HG-104**, profiles

The simulation is over 1000 seconds. All simulations start at the same steady state. After 100 seconds the inlet temperature is stepped from 10 $^\circ C$ to 10.25 $^\circ C$.

A performance summary for all the integration routines, is given in Table 7.5.

Table 7.5: Integration routine summary

Method	Simulation time [sec]	Function calls	RTSTR [sec/sec]
HP	4.818	1001	207.8
Rosenbrock	12.87	1001	77.70
DASPK	22.69	2449	44.13
DVODPK	44.6	5602	22.46
LSODES	41.18	3574	24.31

Results from the UV simulations

The level dynamics of the separation tank, **25-VD-115**, are plotted in Figure 7.3. All the flow dynamics are shown in Figure 7.4 and 7.5. The pressure dynamics of the nodes/flash are given in Figure 7.6 and 7.7. The temperature dynamics of the nodes/flash are given in Figure 7.8 and 7.9.

The dynamics of the internal states of the heat exchanger are not plotted.

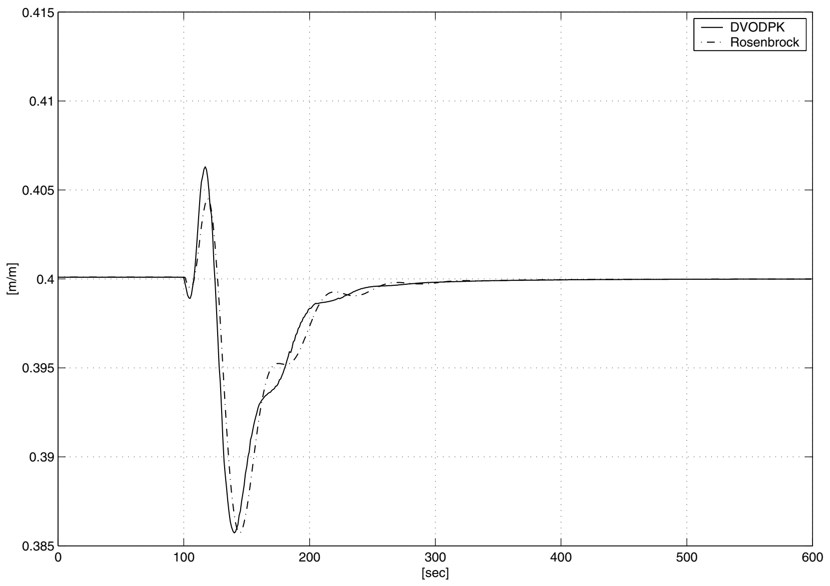


Figure 7.3: Level in **25-VD-115**

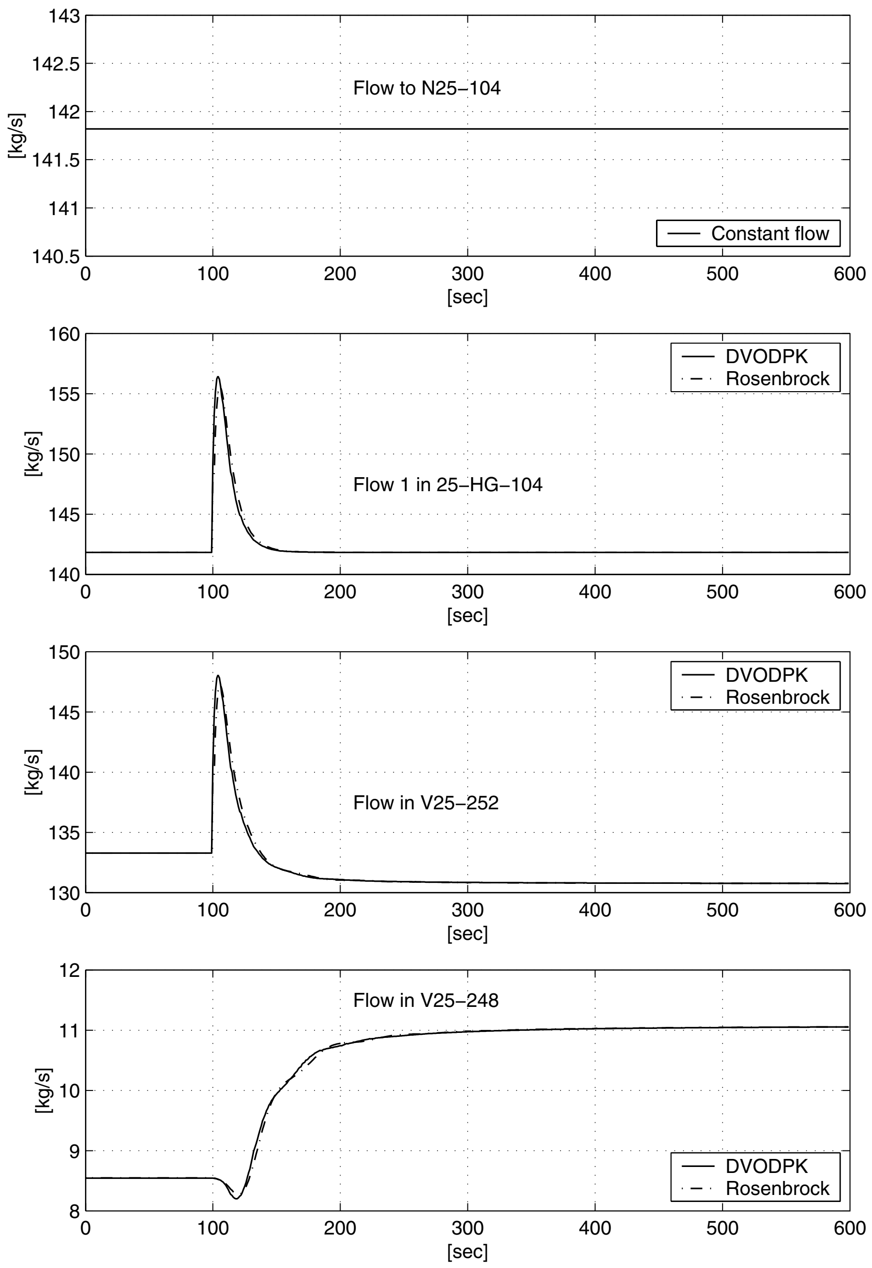


Figure 7.4: Flows in sub LNG plant (1/2)

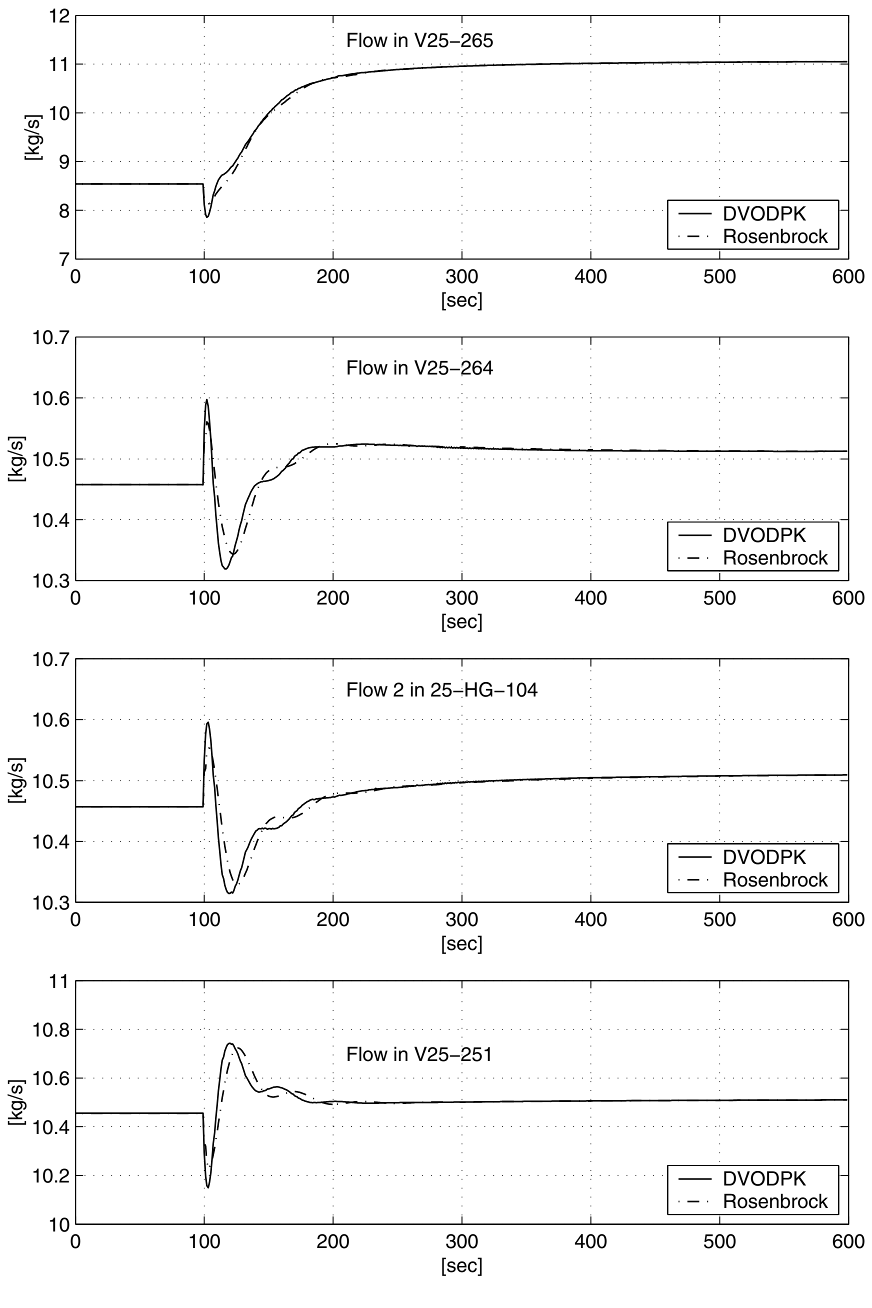
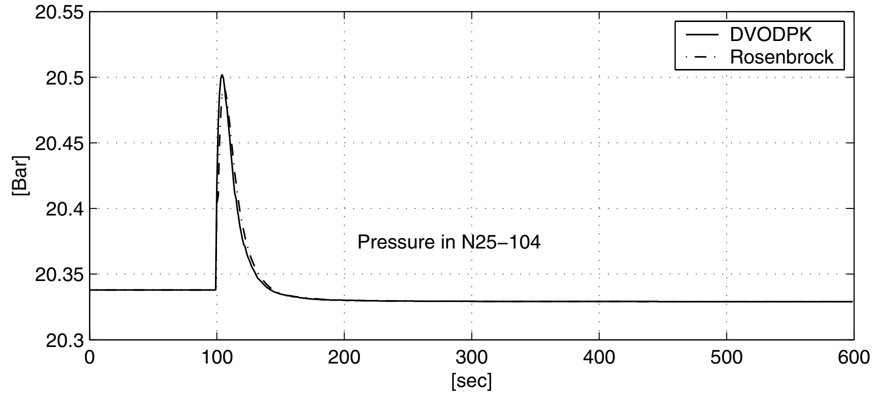
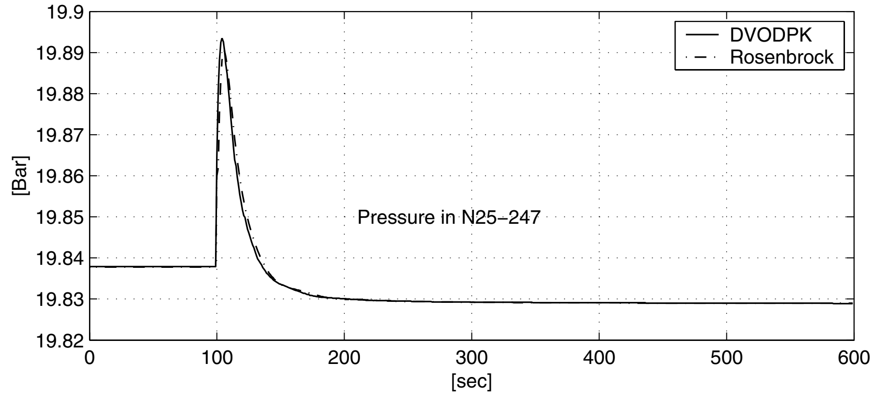


Figure 7.5: Flows in sub LNG plant (2/2)





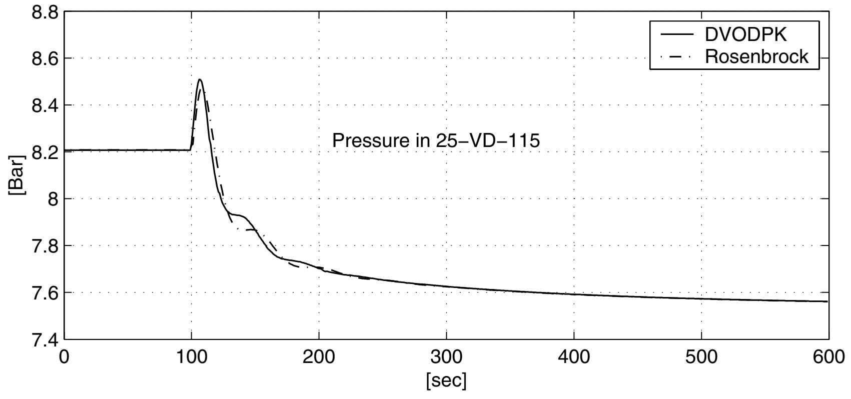


Figure 7.6: Pressures in sub LNG plant (1/2)

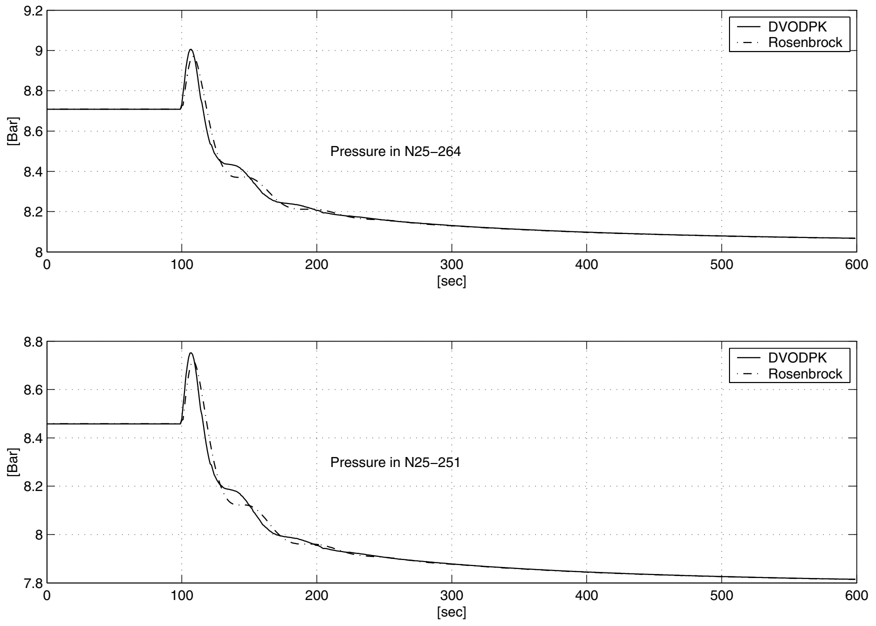


Figure 7.7: Pressures in sub LNG plant (2/2)

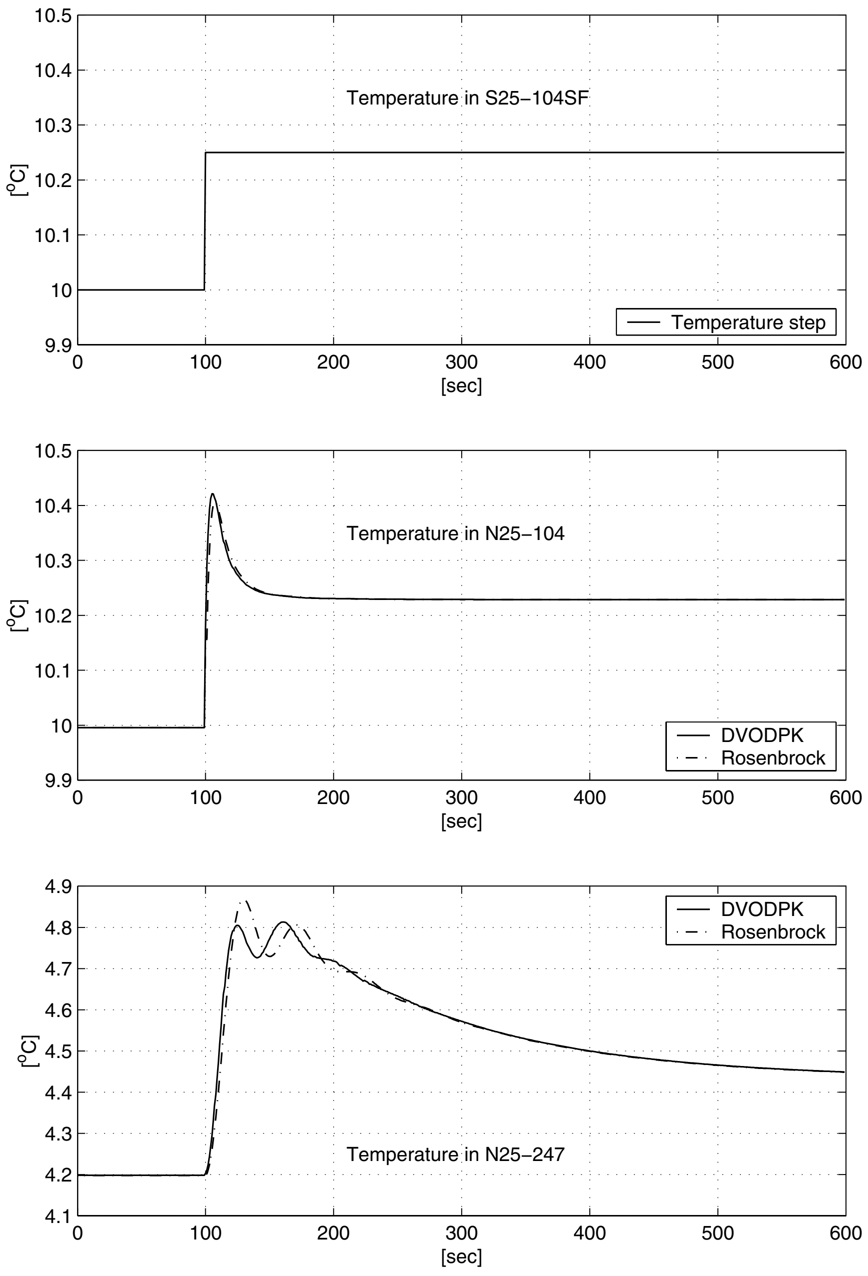


Figure 7.8: Temperatures in sub LNG plant (1/2)

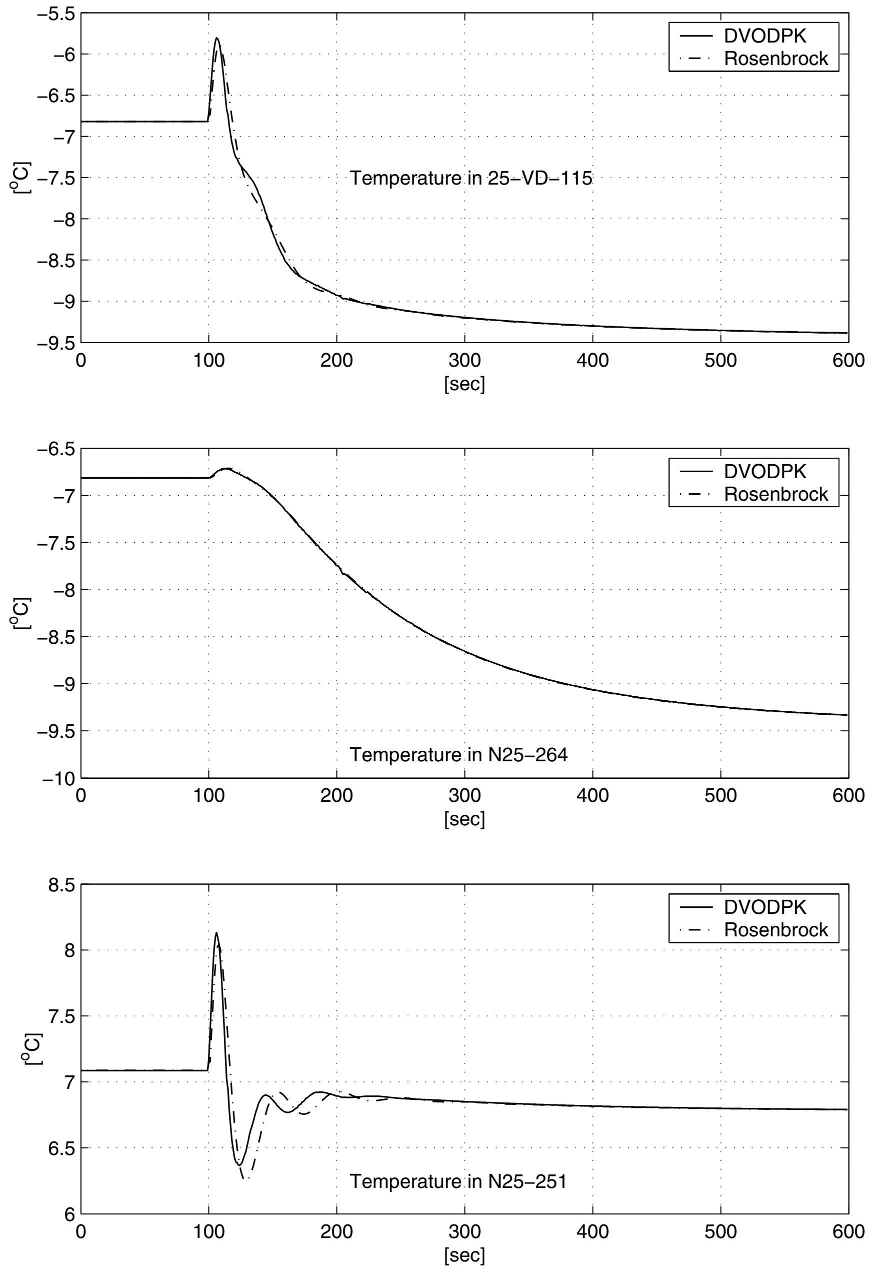


Figure 7.9: Temperatures in sub LNG plant (2/2)

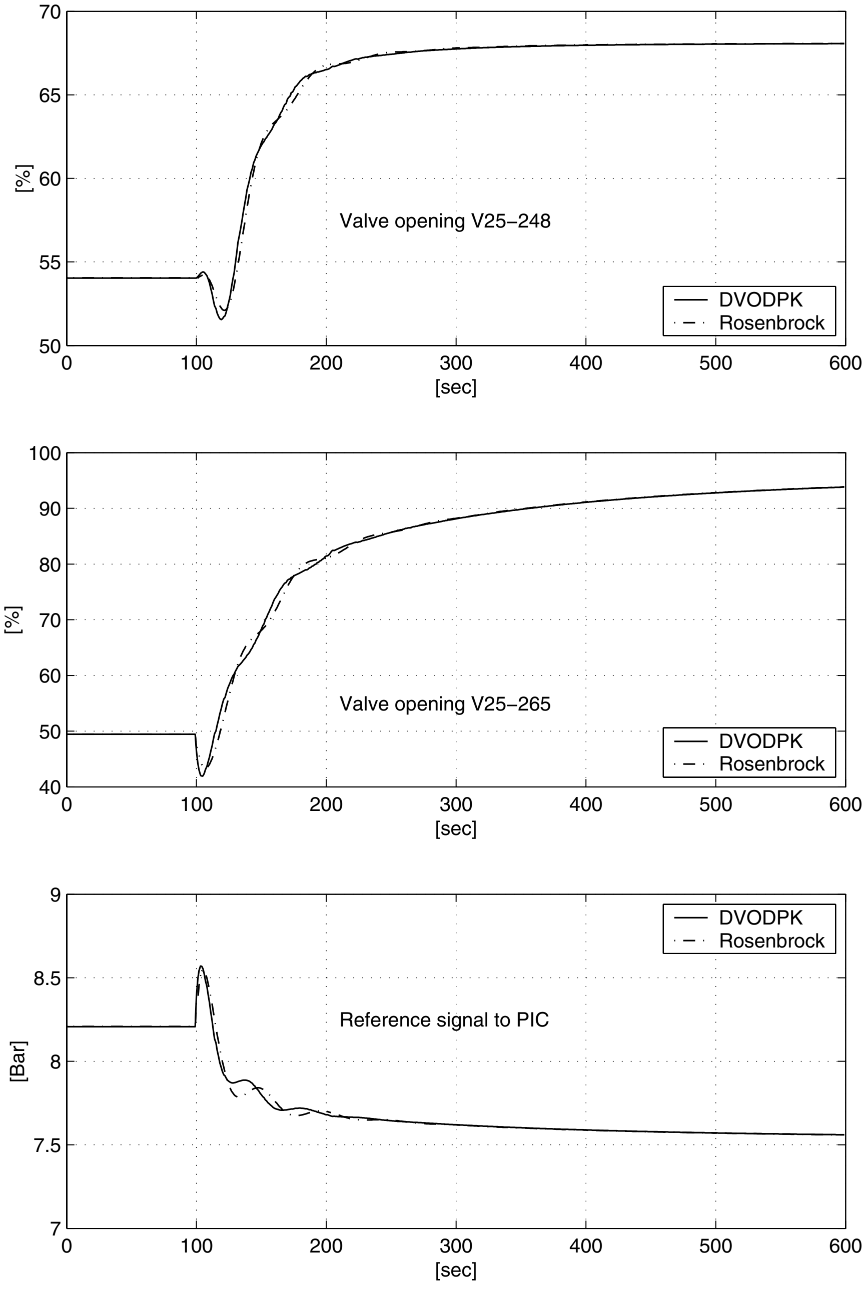


Figure 7.10: Manipulated variables

Results from the HP simulations

The level dynamics of the flash, **25-VD-115**, are plotted in Figure 7.11. All the flow dynamics are shown in Figure 7.12 and 7.13. The pressure of the nodes/flash are given in Figure 7.14 and 7.15. Temperature dynamics of the nodes/flash are given in Figure 7.16 and Figure 7.17.

The dynamics of the internal states of the heat exchanger are not plotted.

The UV simulation results, using DVODPK, are plotted in the same figures as the HP simulation results.

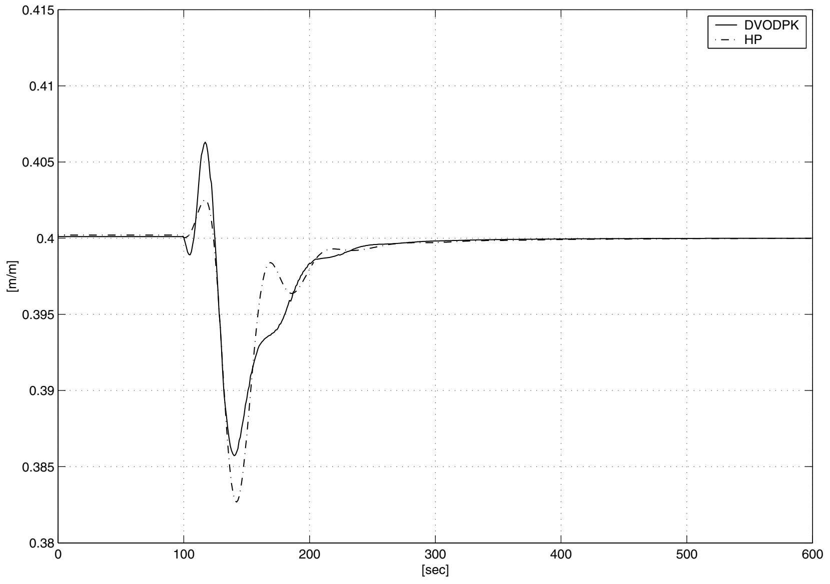


Figure 7.11: Level in **25-VD-115**

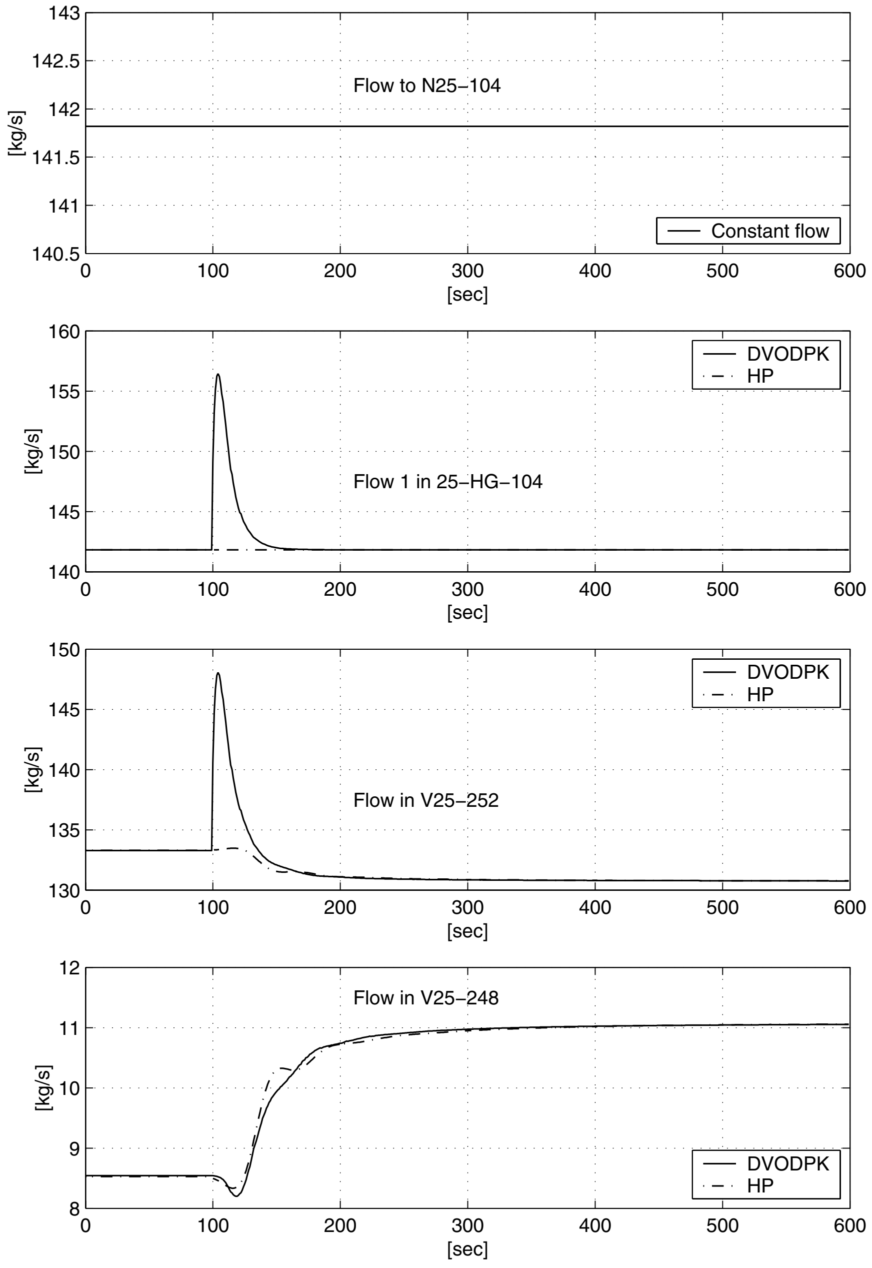


Figure 7.12: Flows in sub LNG plant (1/2)

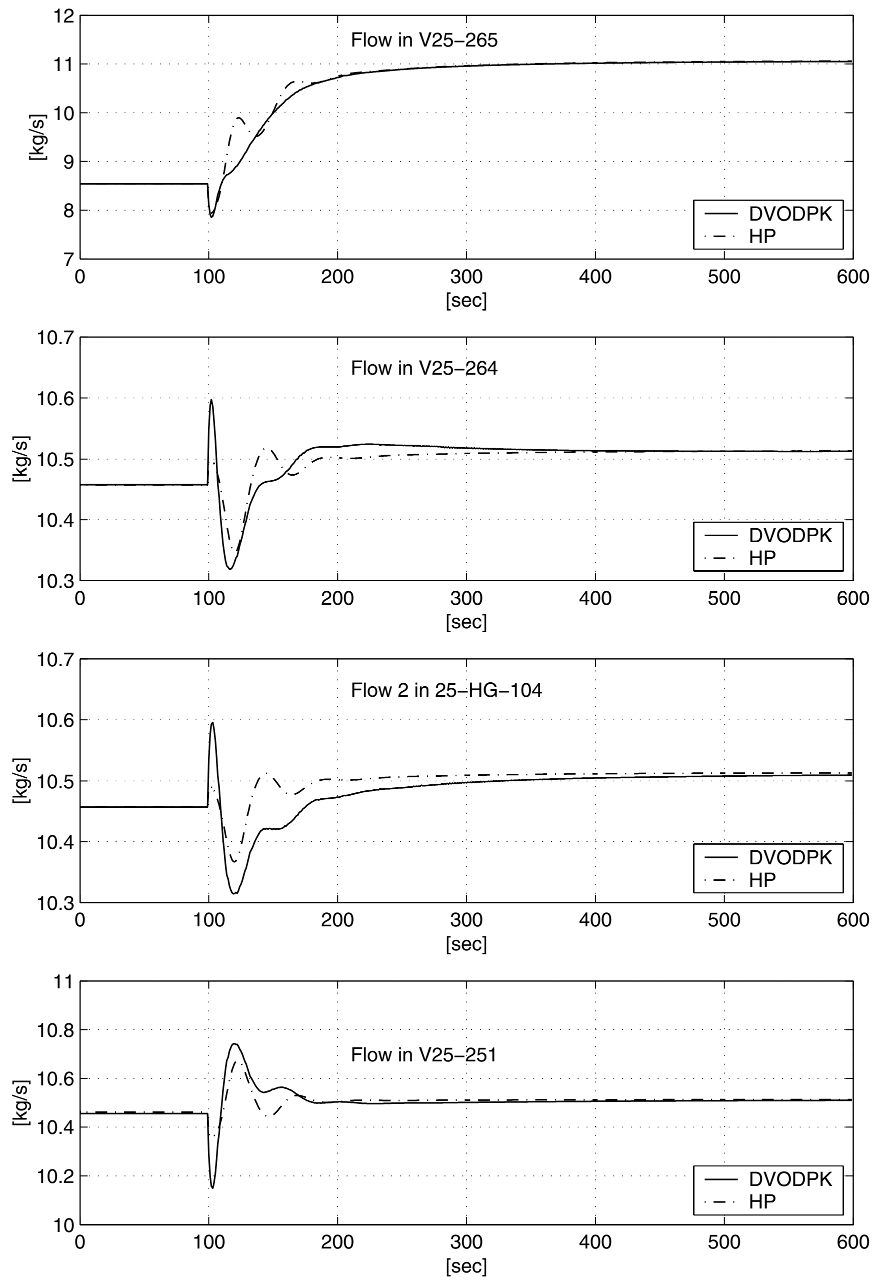


Figure 7.13: Flows in sub LNG plant (2/2)

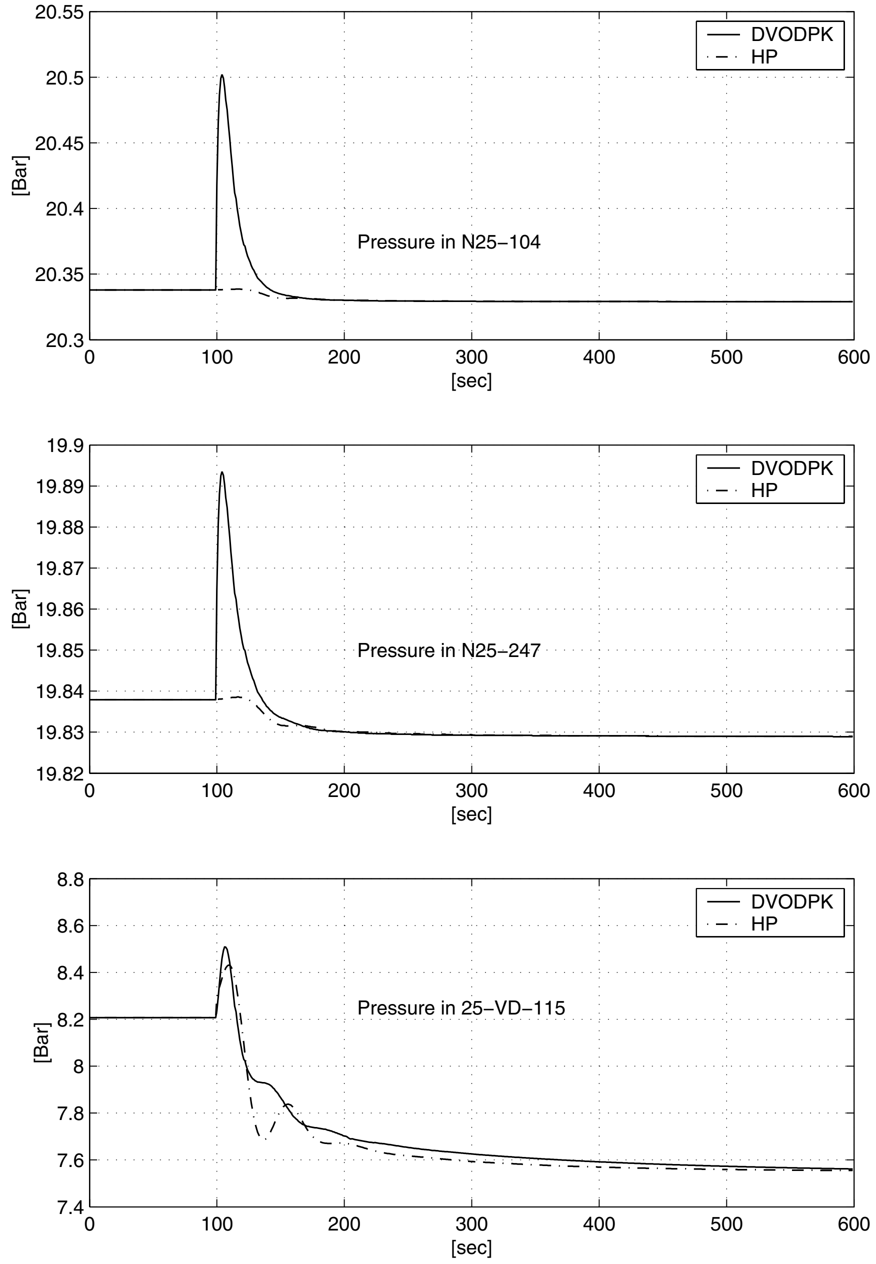


Figure 7.14: Pressures in sub LNG plant (1/2)

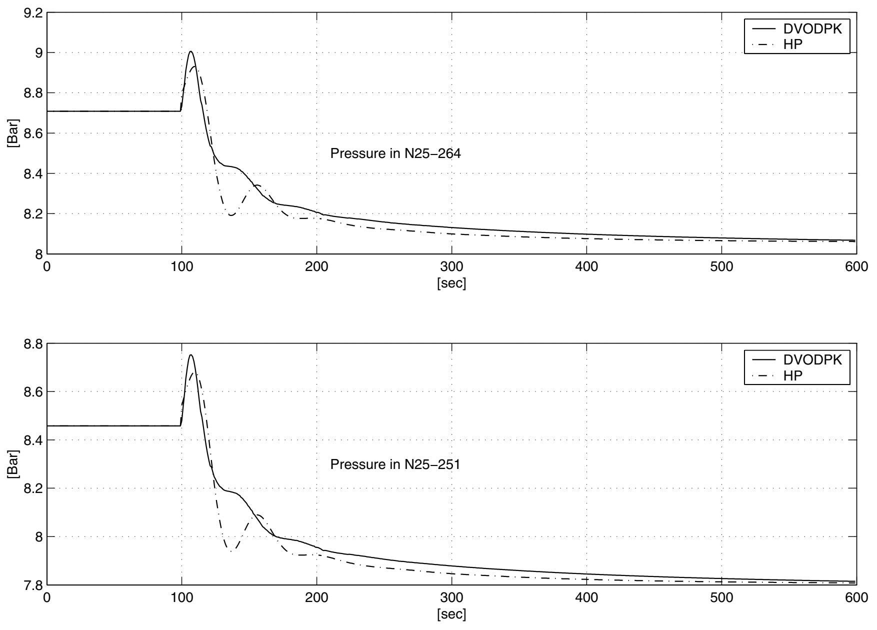


Figure 7.15: Pressures in sub LNG plant (2/2)

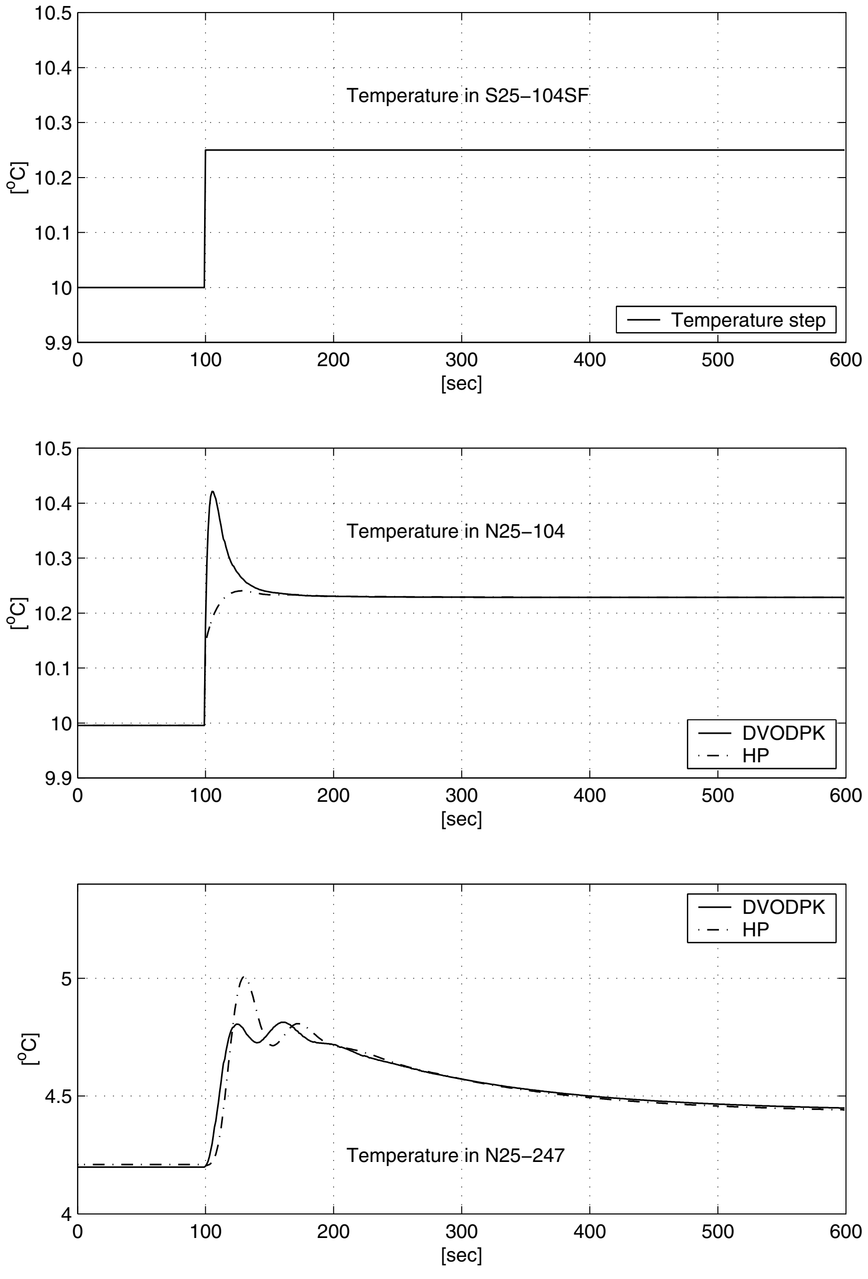


Figure 7.16: Temperatures in sub LNG plant (1/2)

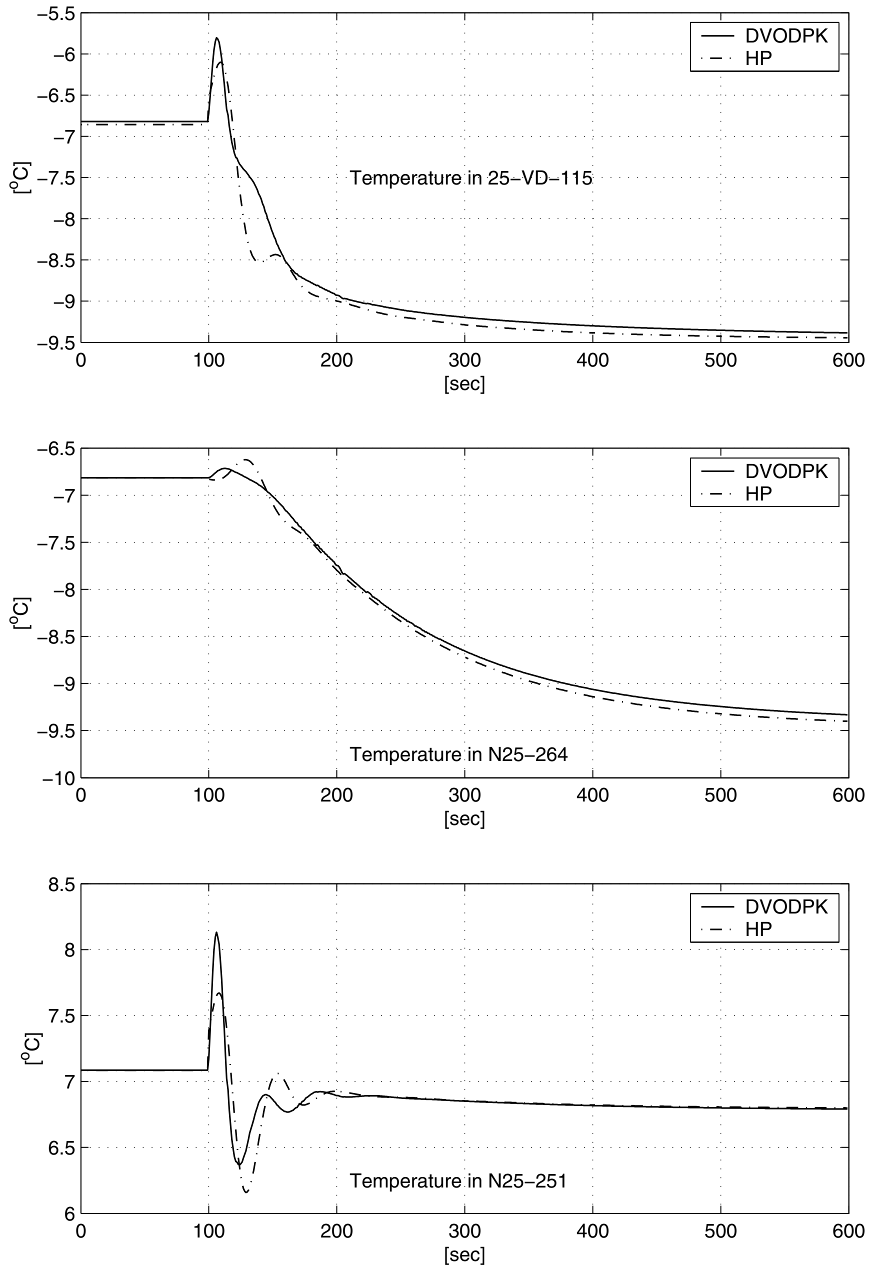


Figure 7.17: Temperatures in sub LNG plant (2/2)

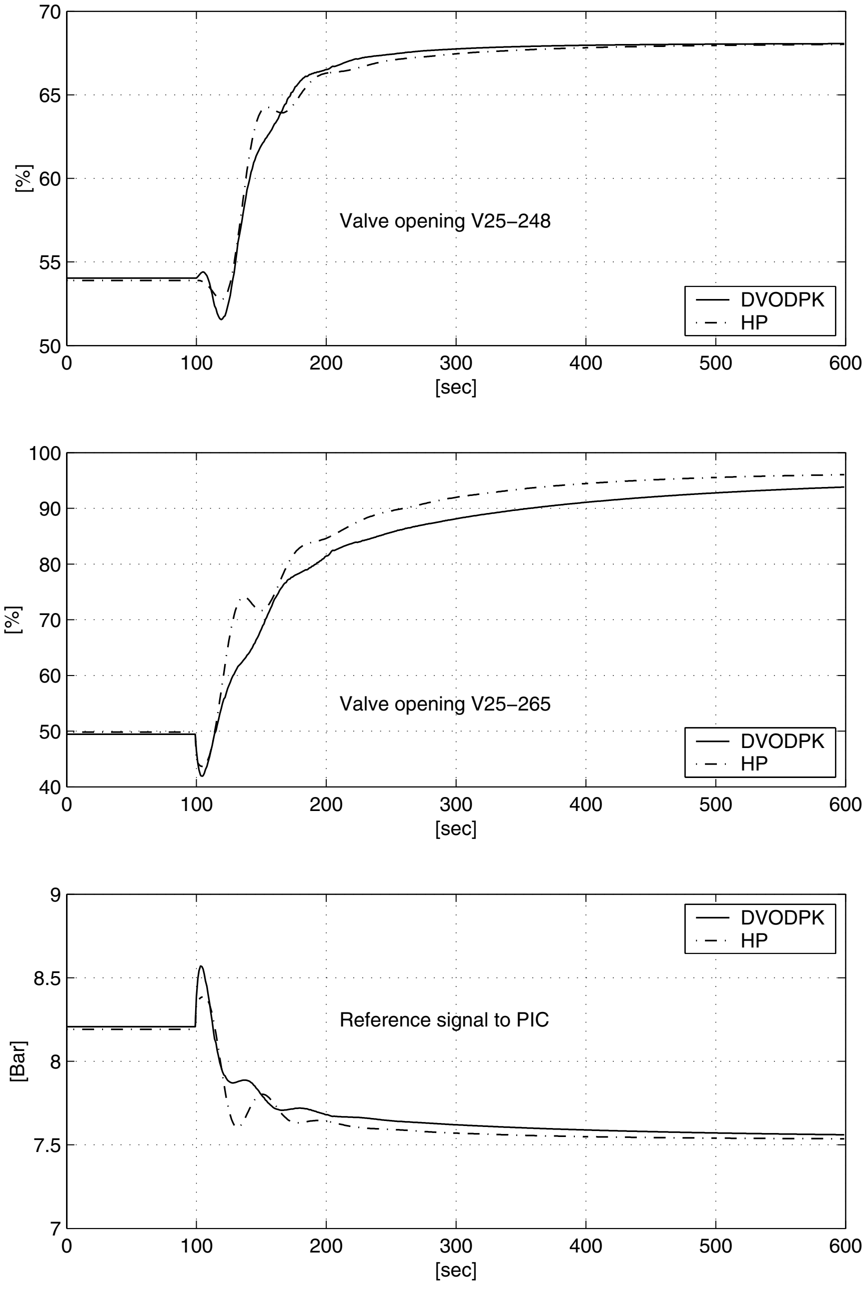


Figure 7.18: Manipulated variables

7.1.2 Comparison of the simulation results - Sub LNG Plant

This sub system is not particularly difficult to integrate, but it contains a two stream heat exchanger, a flash unit, four dynamic nodes, two static nodes, five valves, a constant flow source and three PI controllers. That is; it contains the most used unit models. The heat exchanger has slow dynamics, and is therefore integrable at a reasonable speed for the defined step case.

A step in temperature is chosen because it illustrates an effect of the pressure decoupling when using the enthalpy as a conserved property, instead of the internal energy. To show the difference between the error controlled integration, fixed step UV and HP integration, a selection of variables is plotted in Section 7.1.1. The error controlled integration, the plot of the DVODPK integration, is assumed to display the correct dynamic behavior of the system. Both the 1-stage Rosenbrock integration of the UV formulation and the split HP integration is therefore compared with the error controlled integration.

The error tolerance in the DVODPK/DASPK/LSODES integration is set loosely to 10^{-5} absolute and 10^{-4} relative. Most available dynamic simulators using split integration, are only concerned with stability and simulation time, and not integration error. A loose error tolerance will therefore be sufficient, and show the wanted difference in the various integration schemes.

The integrators are restarted at every sampling time. If the simulators would be able to integrate beyond the sampling time and interpolate to the output time, the simulation time would decrease. For the current case with only one time event and no state events the integrators could be restarted only once. The restart time is predefined by the time of the step change. This is not the general case for a dynamic simulator. Typically the model equation would change at every sample time, because of new set point to the stabilizing controllers.

The fixed step of 1.0 seconds, is chosen because it lies well within the stability limiting time step for the HP integration. The HP integration of the full LNG plant was stable for time steps up to about 1.5 seconds. The UV integration was stable beyond this. A longer time step would be possible, setting larger integral times, T_i, for the fast integrating flow controllers. The present configuration uses $T_i \approx 2$ seconds. Since one second is a reasonable sampling time, the integral times for the flow controllers is kept, and the system integrated with a fixed time step of 1.0 second.

The plots of the DVODPK and UV integration in Section 7.1.1, show a reasonably good fit between the two different integration schemes. Even the fast flow dynamics is well predicted with the simple 1-stage Rosenbrock scheme. Some dynamics are over/under predicted, and shifted slightly in time. This is due to a too large time step.

The plots for the DVODPK and HP integration in Section 7.1.1, show large transient differences. The final steady state is the same for both schemes/formulations. The difference is due to the decoupling in the pressure-flow dynamics and the other slower dynamics. This is particularly illustrated by flow 1 through **25-HG-104**, (from **N25-104** to **N25-247**). The change in inlet temperature to **N25-104** does not affect the pressure in **N25-104** in the same way as with the UV scheme. Therefore the flow through the heat exchanger remains constant. The other differences are mainly a consequence of this discrepancy.

7.2 The LNG plant

Several assumption and simplifications had to be made, before being able to simulate the LNG plant with the flowsheet given in Appendix C. These choices are summarized below.

Some of the material will define the basis for the simulation. This section will therefore be a combined description and discussion of the LNG plant.

7.2.1 Assumptions

The volume of the process equipment is of vital importance, when describing the dynamic behavior of a process plant. The volume sizes determine the time constants of the dynamics. Volumes of nodes and volumes and the base area of flash tanks are guessed.

In order to describe the heat exchangers in the process, heat transfer coefficients, heat transferring areas, wall mass, wall properties, flow cross sections and height of the heat exchanger are selected to fit a predefined steady state at the boundarys. Setting the flow rate, the composition of the flows, the inlet pressure and temperatures, a set of heat exchanger parameters could be determined. The choice of these parameters determines the process dynamics.

The dynamics of the water coolers, **25-HA-104**, **25-HA-111**, **25-HA-112**, **25-HA-113** and **25-HA-114**, is assumed to be fast. The outlet temperature of the hydrocarbon streams are therefore fixed. This implies no limits on the seawater supply.

The equations for the compressors have some parameters. The head/flow curve is defined through 12 points, and a design value for the compressor speed. The polytropic efficiency is set to a default values, that is; 100% efficiency.

The polytropic efficiency of the liquid expander is set to a default value, 75%.

The efficiencies are set to close the equation set.

To describe the flows, a composition vector with 9 components was defined. Table 7.6 shows which components were used in the simulation of the LNG plant.

Table 7.6: Component vector used in LNG plant simulation

Component no.	Component name
1	Nitrogen
2	Methane
3	Ethane
4	Propane
5	i-Butane
6	n-Butane
7	i-Pentane
8	n-Pentane
9	n-Hexane

7.2.2 Simplification

The nodes of the HP system do not include state variables for the component holdup. The nodes copy the composition from the inlet streams, and the outlet streams copy the composition vector from the source node/flash tank. This neglected dynamics will not be important in the simulation of the full LNG plant, since the compositions remain constant for the MCR2 and MCR3 loop. The composition of the NG stream and the MCR1 stream change slightly.

The MFCP process removes heavy hydrocarbons, HHC, after the first stage pre-cooling heat exchanger, **25-HG-101-I**. Figure 7.19 shows the location of the HHC removal column. A small nitrogen removal column for the LNG is also required. Since no column model is available, some of the effect of a column is introduced, using the static source streams **S25-142SF** (positive) and **S25-148SF** (negative). The stream of **S25-142SF** is contains light components, while **S25-148SF** has the composition of the node **25-VD-107**. The temperature from the column top stage is assumed constant, and the heat exchanger **25-HA-X** is added to ensure a fixed temperature.

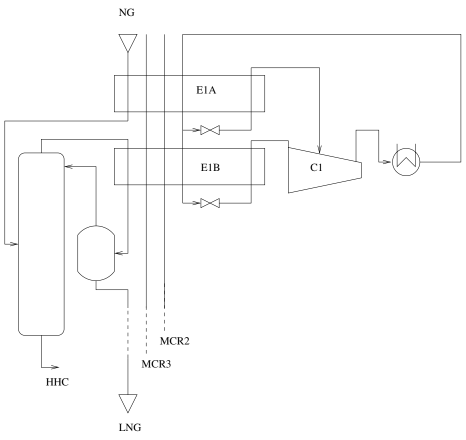


Figure 7.19: MFCP with the HHC column

The effect of the integration with other processes is introduced. Process 26 and 27 are dummy processes, and only defined through the fixed temperature heat exchangers **26-HA-001**, **26-HA-002** and **27-HA-001**.

The valve flow is defined without the density, ρ, dependency. This simplification can be undone re-specifying a flag for the flow units. This will require some retuning of the controllers, and a lot of extra simulation time. This will definately change the physical description, but is not implemented, because the simulator still can be demonstrated.

7.2.3 Process control

The control loops in the LNG plant are given in Chapter 6.

To produce the control parameters, open loop step responses were generated. The response exhibited dynamics that are similar to first or second order dynamics with, time delay, overshoot, and inverse response. Using least square fitting to parametric equations describing these dynamics, easily analyzable dynamic model approximations were generated. Using the fitted equations and parameters, control parameters was guessed/calculated. The control parameters were then roughly tuned until sufficiently good behavior was observed on the closed loop system.

A common inverse response was observed when perturbing the flows used in the temperature controllers. Increasing the cooling agent will lower the outlet tube temperatures. But the initial response is determined by the pressure dynamics at the shell inlet. This dynamics will dampen fast, and the major time constants of the response will depend strongly on the choice of heat transfer coefficients within the heat exchangers.

The HP formulation and the UV formulation use the same set of control parameters. The parameters were generated using the UV-model. This will affect the simulation results. The plots of the UV integration will be smoother.

7.2.4 Solver

The need for a fast simulator means fast integration. For fast integration the evaluation time of every function call, or the number of function calls must be at a minimum. After selecting to use the SRK EOS to describe equilibrium and physical properties, it is given that the function calls will be time consuming. This implies an integrator able to take large time steps. In order for the integration routine to take long steps, several approaches can be taken. Two different methods were selected and implemented. One of the goals of this work was to generate a simulator integrating faster than real-time, locally at all times.

First the splitting of the slow and the fast dynamics, solving the pressure flow system separately from all other dynamics. Initially a 1-stage Rosenbrock integrator was used to solve the pressure flow system, the other dynamics where integrated using an explicit Euler. This implementation managed a fixed time step of approximately 0.1 seconds. Increasing the time step beyond this, the integration became unstable, starting with wrong predictions of the compressor flows. A time step of 0.1 seconds did not enable the simulator to run faster than real-time, and some action had to be taken, further stabilizing the integration. A fully implicit integration of the pressure flow dynamics was implemented. The equations were slightly reformulated and a line search algorithm was added. This increased the simulation time per step slightly, but it had a dramatic effect on the stability. The integrator became stable for time steps more than ten times larger than with only a 1-stage Rosenbrock integrator.

In the present version of the simulator the pressure flow network Jacobian is only partly analytical if flow controllers are added to the flowsheet. The simulation time of the HP formulation would improve if this Jacobian is made fully analytical. The improvement is difficult to quantify, but the required work to implement these Jacobian elements analytical are modest, and should therefore be made analytical. Adding actuator dynamics would probably remove the need to include the controller information in the Jacobian.

The second approach involved the conservation of internal energy and mass in constant volumes. A splitting of the fast and the slow dynamics is not as straight-forward as in the HP case. Even when neglecting the compositional dynamics, the pressure will be a function of internal energy, and holdup. To solve these dynamics implicitly would therefore require several flowsheet function evaluations, and the whole system of dynamic states had to be considered. In order to make the simulator fast when integrating stiff systems, a cheap Jacobian had to be produced. An analytical approach was chosen.

A 1-stage Rosenbrock was found to be the least time consuming integration, using Jacobian information. This method was therefore implemented without any error/step control. The method is A-stable, but can generate a step into a nonfeasible region (also for Jacobian with negative eigenvalues), where the thermodynamic or unit model equations/functions do not produce results, and the simulator can therefore crash. The method was found to be robust. The step size was restricted by the low integral times in the flow controllers, and the same time step as for the HP approach was chosen for the simulations. This produced smooth data for the sub LNG plant simulation and the LNG plant simulation.

The interest in 1-stage Rosenbrock integrator for the UV setting and the split integration for the PH setting is simple. In many applications the dynamic simulator must guarantee a local simulation time faster than real-time. In these cases the integration error is less important. Dealing with implicit nonlinear equations, thermodynamics, it is impossible to guarantee real-time simulation. The lowest RTSTR was 3.69 and 6.25, for the HP formulation and the UV formulation, respectively, simulating the full LNG plant.

The three different freeware integrators, DVODPK/DASPK/LSODES, were tested on the LNG sub-plant, Section 7.1.1, and on the full LNG plant, Section 7.2.5. The integrators worked well on the small LNG sub-plant example, but integrated too slowly, to be useful, on the full LNG plant. The inhibiting part of the flowsheet was some of the heat exchangers. Two possible reasons that can cause this slow simulation were established. One was the discontinuity associated with phase transitions, and the accuracy of the Jacobian information. The problem is not resolved for the current version of the simulator.

Used in control or as a training simulator, the flowsheet function must be evaluated once more every time step. The simulation time therefore increases, relative to the data given in Section 7.2.5. In the present version of the simulator the flow sheet function is evaluated only at the start of a integration step, generating new ODE differentials and Jacobian entries for the integrator. The need for updated temperatures and property values at the sampling point, will require the simulator to evaluate the flow sheet function also before returning from a integration step. Because the starting values for the thermodynamic functions at the entry to a integration step, generally will be very good (no change), the real-time ratios will be reduced approximately in the following way.

Combining the information given in Tables 7.10 and 7.11, an approximate lower bound on RTSTR is calculated for an integrator doing two function evaluations per time step.

Table 7.7: Reduction in real-time simulation time ratio, (RTSTR)

Method	Rel. time in func. eval. [%]	RTSTR scaling [-]	Approx. new RTSTR [-]
HP	< 72.2	> 0.58 (1/1.772)	> 4.22
UV	63.4	> 0.61 (1/1.634)	> 4.73

Calculated, using the same procedure, the local RTSTR will have a lower bound of 2.08 and 3.82, for the HP formulation and UV formulation, respectively.

7.2.5 Simulation results - The LNG plant

The flowsheet for the entire plant is given in Appendix C. Some information about the simulation and the plant description is given in Section 7.2.

The case simulated is a set point change for the temperature controller manipulating **V25-155**, to maintain the set point for the node **N25-155**. The step is implemented after 550 seconds, and the total time span for the simulation is 9000 seconds, $T_{end} = 9000$. If the dynamics come to a steady state before T_{end}, the plotted time span is shortened.

Plots for the important and manipulated streams are shown. The dynamic response of the controlled temperatures are also shown.

The flashes needed to be solved in every function call are tabulated in Table 7.8. The entire LNG plant is described with ODE, network ODE and sub-model ODE

Table 7.8: Flashes per evaluation of the flowsheet function

Method	Flash specification				Overall
	HP	UV	TP	SP	
HP	330	0	10	2	342
UV	276	54	10	2	342

variables, see Table 7.9. The network ODE variables are the dynamic pressures of the nodes and separation tanks, and the integral part of the flow controllers. For the HP formulation, both ODE and sub-model ODE variables are really sub-model variables. The difference is that the sub-model ODEs are within the heat exchangers, and are integrated with a 1-stage Rosenbock algorithm, while the variables referred to as ODE variable are solved using an explicit Euler formulation. The network ODEs are solved using a fully implicit scheme. The

Table 7.9: Dynamic states

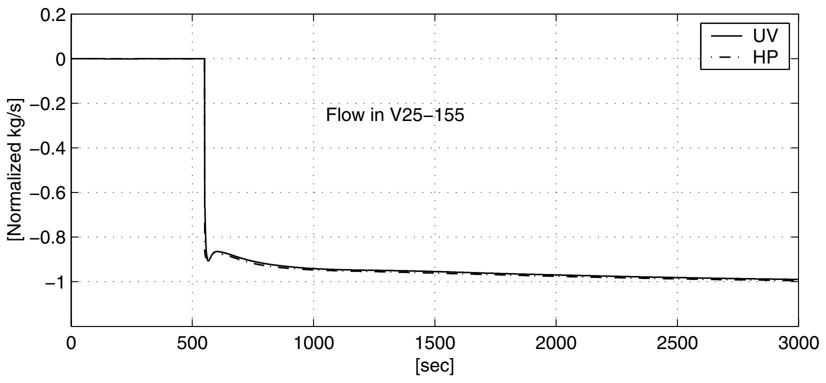
Method	Network	Sub-model ODEs solved using		Total
	ODEs	1-stage Rosenbrock	Explicit Euler	
HP	61	462	88	611
UV	0	1025	0	1025

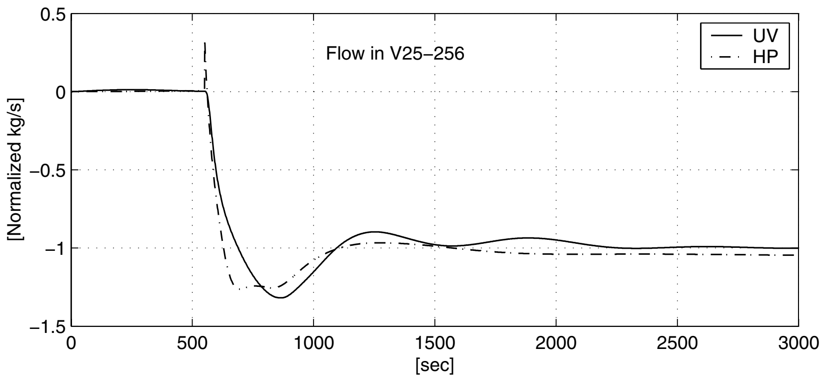
overall PC simulation time for the 9000 second long simulation is given in Table 7.10. The real-time simulation time ratio, RTSTR, is the simulated time span divided by the PC simulation time. If RTSTR ¿ 1, the simulation is faster than real-time. Figures 7.20 and 7.21 shows the flow of the NG, MCR2 and the manipulated streams of MCR1. The fixed MCR1 streams, and the MCR3 stream are not shown.

Table 7.10: Simulation time, simulating 9000 seconds

Method	PC simulation time	RTSTR	Min local RTSTR
	[sec]	[sec/sec]	[sec/sec]
HP	1237.6	7.27	3.69
UV	1164.1	7.73	6.25

Each plot contains a describing text, defining the plotted variable.





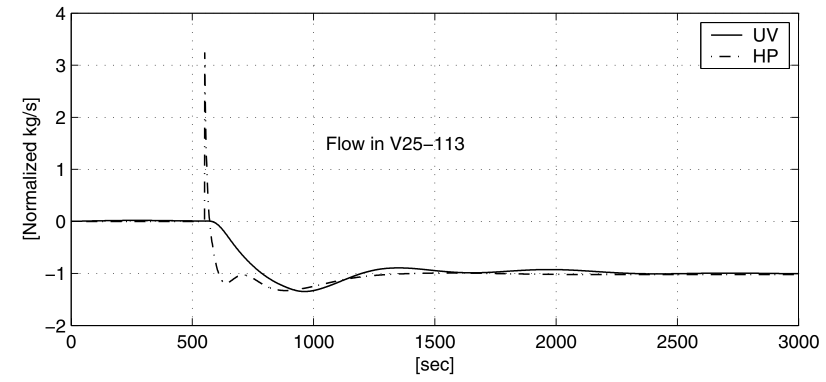


Figure 7.20: HP and UV flows (1/2)

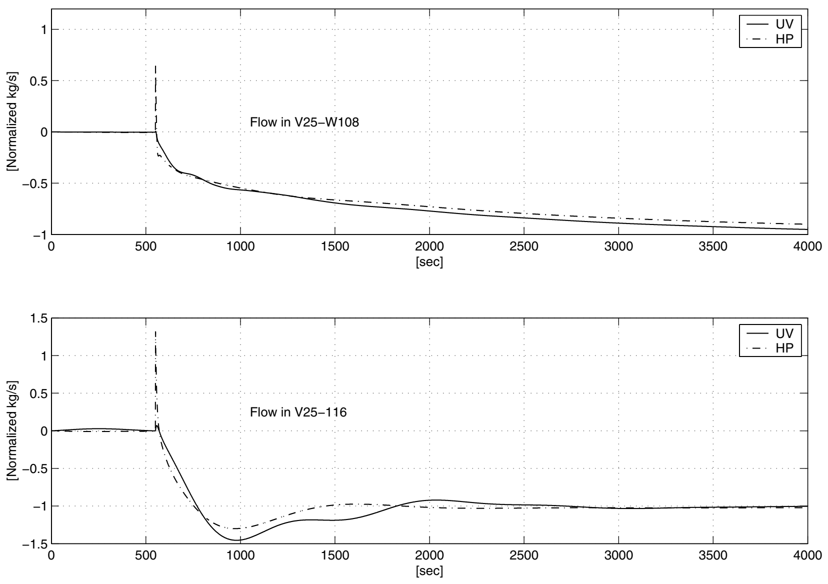


Figure 7.21: HP and UV flows (2/2)

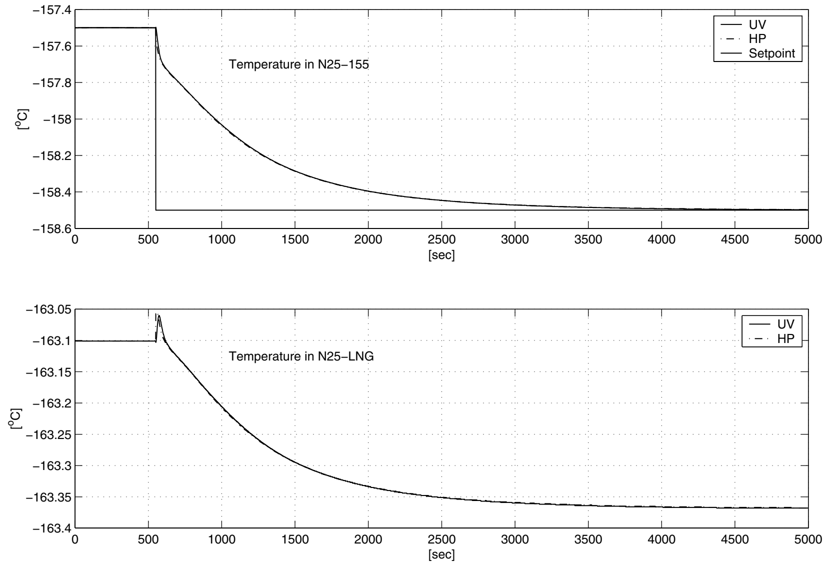


Figure 7.22: HP and UV temperature plots (1/3)

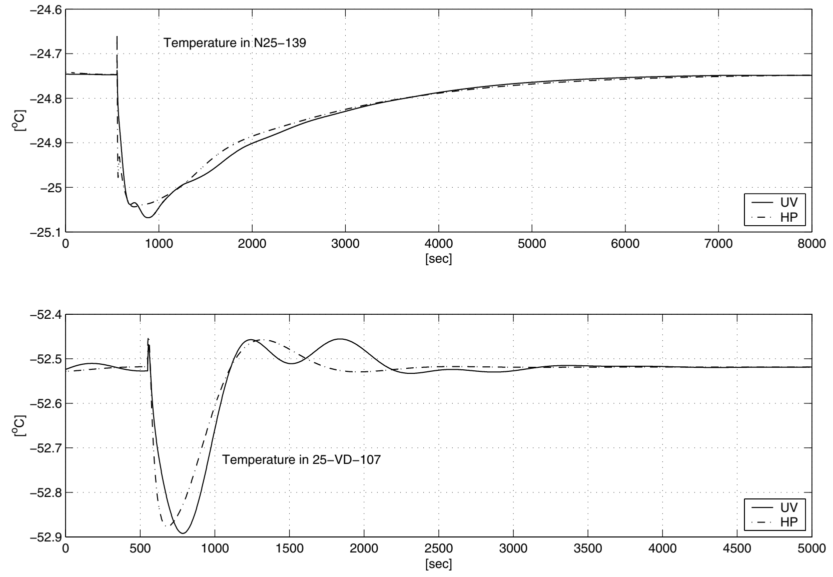


Figure 7.23: HP and UV temperature plots (2/3)

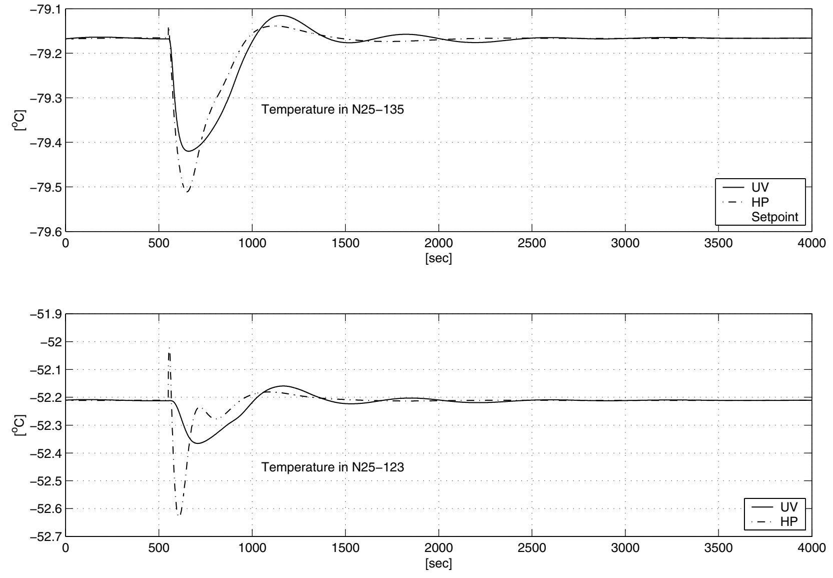


Figure 7.24: HP and UV temperature plots (3/3)

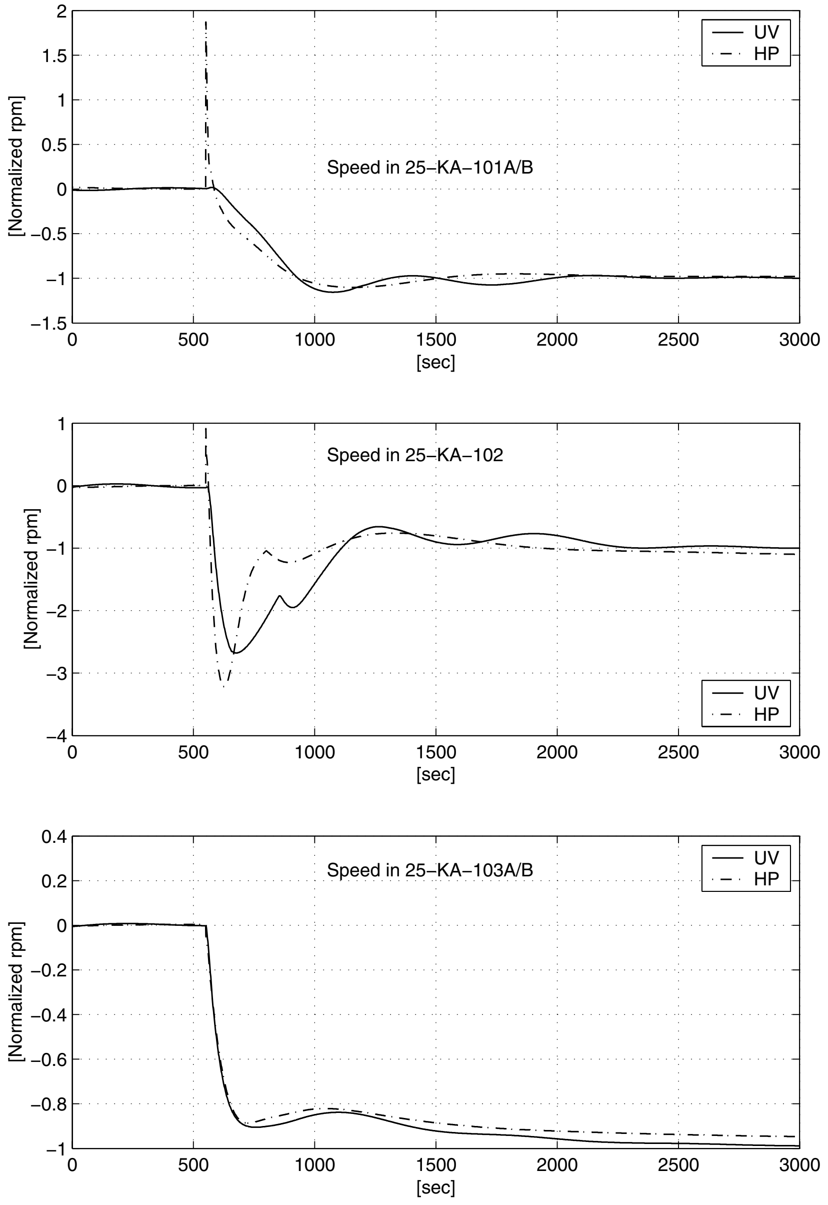


Figure 7.25: The manipulated and normalized compressor speed for all compressors

7.2.6 Comparison of the simulation results - The LNG plant

The simulation results from the full LNG plant are given in the previous section, 7.2.5. The results will be discussed here.

The impact of not including the two separation columns in the LNG plant is qualitatively huge. The large volume of columns will affect the pressure/flow dynamics. The slow compositional dynamics will also influence the heat transfer in the heat exchangers. The changing composition will influence the temperature profile through the heat exchanger.

The influence of not adding the two columns is so big it could be argued that simpler models, and/or simpler thermodynamics should have been used. But this work had the ambition to include a column model and the modelling should be viewed in light of that.

It is seen from Table 7.9 that the UV simulation is executed with 414 dynamic states more than the HP simulation. This is due to the neglected composition dynamics in the HP nodes. The states of all nine components are neglected in 52 nodes. Since the HP formulation has a dynamic state for the pressure in the 54 nodes and flash tanks, the difference in ODE states is 52x9-54=414.

Table 7.10 shows that the UV formulation for the overall simulation is 6.3% faster than the HP formulation. But the minimum local simulation speed of the UV formulation is 1.7 times faster than the HP formulation. This minimum RTSTR values for the HP formulation occurs under some fast dynamics in the seconds following the setpoint change for the outlet temperature. The cause of this low value is the increased time spent solving the implicit flow-pressure network equations. The HP formulation is expected to simulate faster if the Jacobian is made fully analytical.

The fact that the minimum RTSTR is larger than one for both formulations makes them usable in real-time systems.

Looking at the plots in Figure 7.20 through 7.25, the following general trends are seen. The HP simulated variables show an inverse peak initially after the set point change. The dynamics described with the PH and UV formulation show noticeable differences, before approaching approximately the same steady state.

The initial peak in the PH formulation can be caused by several factors. The first is the fixed time step. Reducing the time step by a factor of 10 does not alter this initial peak, and it is therefore not believed to be an integration error. Second, it can be related to an error in the implementation or the configuration of the controllers. This is thoroughly checked, and incorrect implementation is not causing this effect. Last, it can be associated with the controller tunings. The controllers were tuned with the UV model, and some of the dynamics have different behavior on the HP and the UV formulations.

The differences in the simulated dynamics are caused by the decoupling of the pressure from the energy conserved property. When integrating the pressure with a network solver the mass conserved in the node will not always fit within the given volume. That is; mass will not equal volume*density at all times. This will give incorrect dynamics. The effect of changing the time step is minimal on the initial inverse top of the HP integration, and the integration error is therefore not believed to be the causing effect.

7.3 Discontinuous differentials for phase change in a node

When increasing the MCR2 flow from its initial steady state flow, the outlet temperature MCR2 in **25-HG-102-I** will increase, because the shell flow of **25-HG-102-I** is constant. The node **N25-122B**, initially a pure liquid state, will enter the two-phase region. The pressure in this node is 17.7-20.2 bar. The compressibility for the liquid phase, close to the bubble point, is 0.018 (kmol/m3/bar) and the compressibility of the two-phase, at low vapor fractions ($< 10^{-3}$) , is $\approx$ 7.81 (kmol/m3/bar). That is; a ratio of $\approx$ 438.

When this change occurs, the node will allow for more flow to enter, resulting in increased pressure. These dynamics are fast, and must be described using a small time step. The flow in and out of **N25-122B**, tube streams in **25-HG-102-I** and **25-HG-102-II**, is plotted vs. time in Figure 7.26. The temperature and pressure of the node are plotted in Figure 7.27.

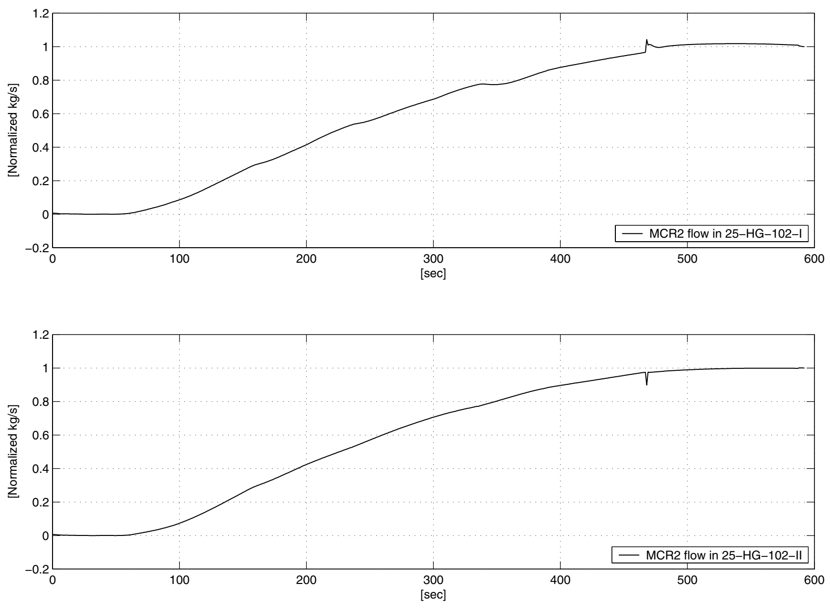


Figure 7.26: MCR2 flow in **25-HG-102-I** and **25-HG-102-II**

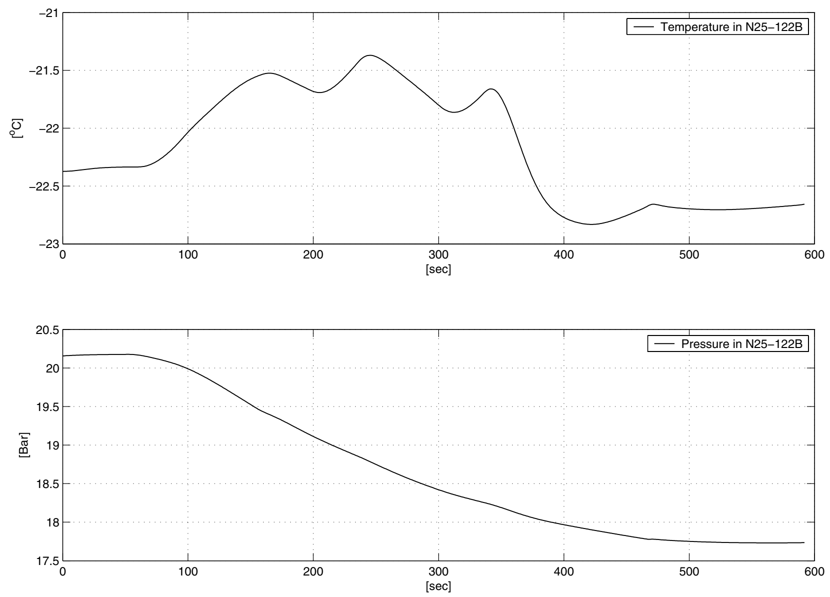


Figure 7.27: Temperature and pressure in node between **25-HG-102-I** and **25-HG-102-II**, **N25-122B**

Figures 7.28 and 7.29, are a “magnification” of a shorter time span, where the state event occurs. To simulate the phase transition with different time steps, the process has to be saved and restarted and simulated with every time step. The only save possibility, is a save to an ASCII file with moderate precision, and the plots below might therefore be slightly time shifted, relative Figures 7.26 and 7.27.

The new “magnified” flow plots in Figure 7.28 are renormalized to zero at t=452, and one at t=512. All the predicted flows are scaled using the final value, t=512, of the simulation using the $\Delta t = 1.0$.

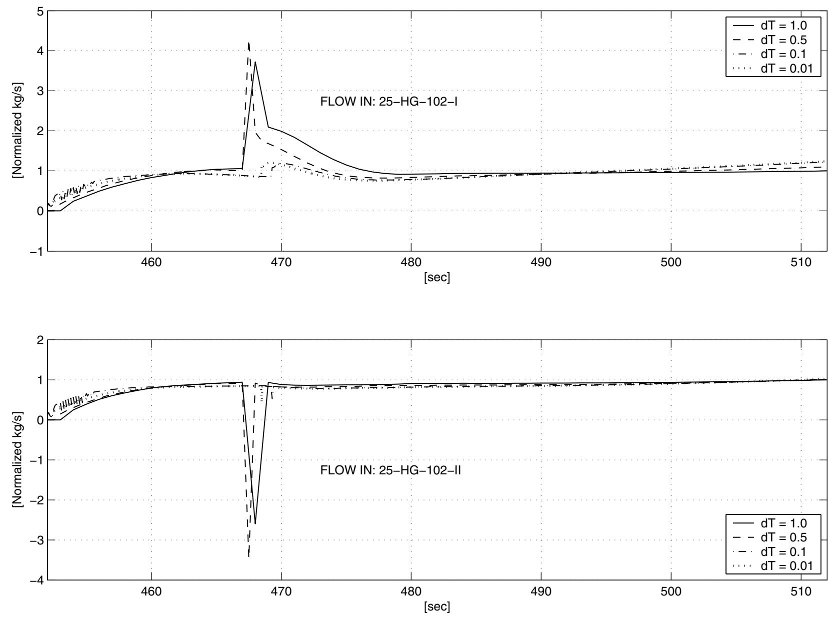


Figure 7.28: MCR2 flow in **25-HG-102-I** and **25-HG-102-II**

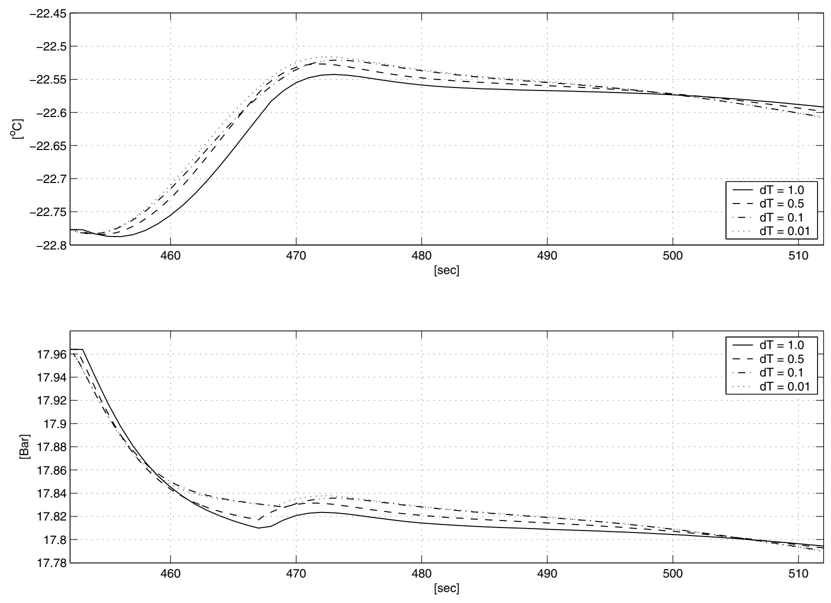


Figure 7.29: Temperature and pressure in node between **25-HG-102-I** and **25-HG-102-II**, **N25-122B**

7.3.1 Comparison of the simulation results - Discontinuities

One example of a state discontinuity is shown in Section 7.3. The effect is seen using the UV formulation.

The effect of the discontinuity and the change in properties is best seen in Figure 7.28. The plot shows how the flow of the two heat exchangers, connecting to the node changing phase, behaves at the phase transition. The simulation with the 0.01 time step gives the most accurate prediction of the flow, and is therefore used as reference for the other simulations. One unit on the y-axis is approximately 1.0 kg/s.

The phase change at different times depending on the fixed time step time. This variance is due to the integration error prior to the change.

The long time steps, 0.5 and 1.0, overpredict the flow dynamics. The size of the overprediction depends on both the step time size and the node state when first entering two phase. The volume of the node become more compressible when the phase change from liquid to two phase. As a result of this, the inflow and the pressure increase.

The discontinuity is best seen on the pressure plot, Figure 7.29. The time pressure differential is clearly discontinuous.

This effect was not observed when simulating the full LNG plant, Section 7.2.5, but will also occur for the LNG plant under the right circumstances. The integrator should therefore detect phase changes, and adjust the time step if an accurate prediction is required.

7.4 Code profiling

The profiling is conducted using Microsoft PROFILE v 6.00. This is an instrumenting profiler. The profiler generates and maps the average time spent in every function call.

A profiling run, simulating 1000 seconds with the LNG plant configuration, with the same temperature set point step as implemented for the simulation results in Section 7.2.5 is done using both the UV and HP integration. The set point is changed after 200 seconds.

The result of a profiling should give clear indications about where the model is spending most time. These insights should indicate the overall improvement if part of the code reduces its simulation time. It can also give an idea about the potential of an optimal code. Table 7.11 shows the result of the profiling run.

Table 7.11: Simulation time

Method	TD	Plotting	ODE solver	Unit models
	[%]	[%]	[%]	[%]
HP	69.7	18.0	9.8	2.5
UV	60.9	26.7	9.9	2.5

The results are not accurate, but can be used in a qualitative discussion.

The time spent in four different parts of the code is considered.

- TD - Thermodynamics - mainly flash calls, but also property calls.
- Plotting - Plotting to screen, writing to file and general administration in the main program. (Mainly plotting to screen.)
- ODE solver - Time spent in the ODE solver, but not in “TD”, “Plotting” or “Unit models”
  - HP - time spent in the Gauss LU solver and implicit integration.

  - UV - The ILU reconditioner and the GMRES solver. (Equal percentage spent on both functions)

- Unit models - Time spent in the unit modules, excluding the time spent in TD and plotting.

The profiling shows that the thermodynamics dominate over the other two categories. That is; speeding up the thermodynamic calculations have the greatest potential to reduce the overall calculation time. Several actions could improve the time spent calculating equilibrium and properties. A first action would be to remove unnecessary calls and multiple calls to property functions. Reimplementing all the flash calls using a Newton-Raphson solver instead of the current nested loop approach would also have improved the performance. The UV-formulation could be improved if all the implicit equations where solved simultaneous together with the ODEs.

7.5 Further work - simulator development

This section will highlight some of the simulator deficiencies, and propose development actions to overcome them. Some subjects have been discussed earlier, but are restated here, to collect all comments for further work in this chapter.

The simulator only has a simple column model for the HP formulation. The LNG plant simulated in Section 7.2.5, MFCP, has two columns. These two columns should therefore be added to the model. A column model for both the HP and UV formulations, with rigorous thermodynamics should therefore be implemented.

One measure to improve the heat transfer descriptions in the heat exchanger. This is believed to be the one of the weakest simplifications made in all the unit models. A more realistic heat transfer coefficient is needed to give better predictions. It is also difficult to set reasonable fixed heat transfer coefficients in the current model. The reason for this is that the set of heat transfer coefficients yielding the correct initial conditions is not unique. The initial conditions for a heat exchanger will typically be; fixed flow, fixed composition, fixed inlet and outlet temperatures and pressures. This means that the heat transfer coefficients, one for each stream, are the free variables. The outlet energy states, one for each stream, are the desired steady state/specifications. The energy balance must also be maintained, reducing the number of specifications by one. That is; one heat transfer coefficient must be set and the remaining must be tuned.

Using correlations for the heat transfer still will require some tuning/scaling to produce the initial steady state. But more realistic heat transfer coefficients, valid over a wider range of working conditions, will be easier available.

One of the most important type of applications for a dynamic model is for control purposes. To make simple studies using the simulator, the state data produced must be easily accessible. This requires an interface from an analyzing tool. The control community at Norwegian University of Science and Technology has a strong history using Simulink and Matlab from “The MathWorks, Inc.” To make the model more useful such an interface should be made for the unit models as well as the entire plant configured in SEPTIC. Testing more advanced controllers (not PI) should also be supported from the configuration file. These controllers would not be implemented as continuous controllers as is the case for the PI controllers already included. The output of these controllers would therefore remain constant between the predefined sample times, and therefore not be included in the Jacobian matrix. The inclusion of an arbitrary controller into the Jacobian matrix is possible but not a priority task.

The PI controllers implemented in the current version of the simulator have restrictions on the number of inputs/outputs, and the number of cascade levels. Two inputs and one output is supported. The sign of the two input signals can be manipulated. Only one cascade level is supported. This is restrictive and troublesome when configuring a process, and the simulator should be extended to handle multiple inlets and outlets. The need to include support for additional cascade levels is questionable, since the outer loops of cascade need to be slower than the inner loops. The outer loops therefore could be added without interfering with the Jacobian entries.

Today some fixed model parameters are recalculated at startup. The restarting point of a simulation, ended at steady state, will drift slightly away from the steady state before returning. This is time consuming and unnecessary, and should be improved. An example of a recalculated property is the compression factor for the compressor equations.

For the UV formulation a large integration error was observed when the phase changed from liquid to two-phase in a node. Similar effects might appear otherwise at phase change state events. To prevent a large error in the integration, an integrator able to track the phases, and reduce the time step when such phase changes should be implemented.

The HP formulation only utilize a partly analytical pressure-flow Jacobian in the present version . The Jacobian elements in the pressure flow network depending on the integral parts in the flow controllers are generated through perturbation. At the same time the pressure controllers, manipulating flow, are neglected from the implicit pressure integration, and integrated with a forward Euler scheme. To speed up the HP simulations, controllers with measured flow or pressure, manipulating flow should be included in the Jacobian for the pressure flow system, and generate analytical entries.

The thermodynamic implementation is far from optimal. The major opportunity for improvement is to remove unnecessary property/function calls. In the present version of the code, the Jacobian for the flash system is not reused when calculating the linearization along equilibrium. The result is excess calls to the fugacity and property functions (enthalpy and density). A conversion of the Fortran code thermodynamics library, TPLIB, to C/C++ would simplify the further optimization of the code, easing the memory handling.

The implementation of the simulator permits parallelization of the code. The unit operations can be calculated independently of each other, and the GMRES algorithm also can be solved in parallel, see Sosonkina, Allison, and Watson (1998), for example. Therefore, if desired, the simulator could utilize multiple CPU's.

Chapter 8

Conclusions

A general purpose dynamic process simulator is implemented and tested. The principles of mass and energy conservation, are used in combination with a simplified quasi-steady state momentum equation to govern the process ODE equations. The simulator is configured and tested on the MFCP NG liquefaction plant.

Two model approaches is used. One approach conserves energy in an enthalpy state, and the other conserves internal energy. The two methods are referred to as the *HP formulation* and *UV formulation*. The HP formulation defines a dynamic state for the pressure, and splitting the integration of the fast and slow dynamics. The pressure state and the algebraic flow relations are solved with a fully implicit Euler scheme, while the internal unit model equation is solved with tailormade integration routines. The UV formulation, utilizing an analytical Jacobian, is integrated using both a 1-stage Rosenbrock and freeware BDF codes.

Unit models for tanks/pipes, separation tanks, valves, liquid expanders, pumps, compressors, heat exchangers, and PI controllers are described using rigorous thermodynamics. Equilibrium is assumed, and equilibrium and physical properties are predicted with the SRK or the PR EOS. In order for the UV formulation to generate an analytical Jacobian the equilibrium is linearized in dynamic state variables, to produce differentials in the internal flash equations.

The simulator is tested on a portable Zepto Z-Note 4100 with a 1.7Ghz Intel Pentium M (Centrino) processor and 1GB of DDR ram. The integration speed is defined through the Real Time Simulation Time Ratio, RTSTR.

The freeware BDF integrators DVODPK, DASPK and LSODES are tested on the entire MFCP plant, but do not integrate the plant with a usable RTSTR. A modified sub-plant of the MFCP is therefore used to test the BDF integrators.

The 1-stage Rosenbrock UV formulation and the HP formulation are compared with the results from the BDF test. All three BDF integrators generate the same output, and the 1-stage Rosenbrock show similar dynamics. The HP formulation produces different dynamics. The dynamic results of the UV formulation are believed to be most correct, since the formulation is physically more correct than the HP formulation.

The full MFCP LNG plant is simulated with a fixed time step of 1.0 seconds, for both the HP and UV formulations. The UV formulation use the 1-stage Rosenbrock method. The case is a set point step for the LNG temperature controller. The plant is simulated from t=0 to t=9000 seconds. The major dynamics for the process are sampled every second, and plotted. The average performance, RTSTR, of the HP and UV formulation is 7.27 and 7.73 respectively. The local performance of the UV formulation proved better than the HP formulation. The minimum local RTSTR of the HP formulation was 3.69 while the UV formulation integrated at 6.25. The HP and UV formulations gave significantly different dynamic predictions.

The major deficiencies in the current version of the simulator, is the lack of the column model. In order to describe general process plants, the simulator needs a unit model describing column dynamics.

Profiling of the code shows an approximately percentage time spent in the thermodynamics of 70 and 61 for the HP and UV simulation.

The dynamic simulators in the industry today are typically using a HP formulation, as described here, with precalculated thermodynamic data stored in look-up tables. The simulations of the full LNG plant show that simulators, utilizing EOS equilibrium descriptions, SRK and PR, soon will be able to compare with modern industrial simulators. A simulator conserving internal energy instead of enthalpy and calculating thermodynamic properties during integration will give a better dynamic process description than HP network simulators.

Bibliography

Abbott, K.A. 1996. “Very Large Scale Modeling.” Ph.D. diss., Department of Chemical Engineering, Carnegie-Mellon University, Pittsburgh.

Anderko, A., J.E. Coon, and S.M. Goldfarb. 1994. “Local thermodynamic models for dynamic process simulation.” *IFAC Symposium: ADCHEM'94.*

Anderson, E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbau, S. Hammarling, A. McKenny, O. Susan, and D. Sorensen. 2000. LAPACK v.3.0. http://www.netlib.org/lapack/. National Science Foundation and Department of Energy, USA.

Atkins, P.W. 1994. *Physical Chemistry.* 5th ed. Oxford University Press.

Aunan, B. 2000. “Shell-side heat transfer and pressure drop in coil-wound LNG heat exchangers, Laboratory measurements and modelling.” Ph.D. diss., Norwegian University of Science and Technology.

Bachmann, R., L. Brüll, TH. Mrziglod, and U. Pallaske. 1990. “On Methods for Reducing the Index of Differential Algebraic Equations.” *Computers Chem. Engng.* 12 (11): 1271–1273.

Barton, P.I. 1992. “The Modelling and Simulation of Combined Discrete/Continuous Processes.” Ph.D. diss., Department of Chemical Engineering, Imperial College of Science, London.

———. 2000. The Equation Oriented Strategy for Process Flowsheeting. Available at: http://yoric.mit.edu/abacuss2/Lecture1.pdf. (Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139).

Bezzo, F., S. Macchietto, and C.C. Pantelides. 2003. “An Object-Oriented Approach to Hybrid CFD/Multizonal Modelling.” *The sixth Italian Conference on Chemical and Process Engineering, Pisa, Italy.*

Bischof, C. H., A. Carle, P. D. Hovland, P. Khademi, , and A. Mauer. 1998. “ADIFOR 2.0 users guide (Revision D).” Technical Report MCS: 192 and CRPC-95516-S, Mathematics and Computer Science Division (MCS), Argonne National Laboratory and Center for Research on Parallel Computation (CRPC), Rice University.

Bischof, C.H., L. Rho, and A. Mauer. 1997. “ADIC - An extensible automatic differentiation tool for ANSI-C.” *Software—Practice & Experience* 27 (12): 1427–1456.

Brosilow, C.B., Y. Liu, J. Cook, and J. Klatt. 1985. “Modular Integration Methods for Simulation of Large Scale Dynamic Systems.” *Modelling, identification and control* 6 (3): 153–179.

Brown, P.N., A.C. Hindmarsh, and L.R. Petzold. 1994. “Using Krylow Methods in the Solution of Large-Scale Differential-Algebraic Systems.” *SIAM J. Sci. Comput.* 15 (6): 1467–1488.

———. 1998. “Consistent initial condition calculation for Differential-Algebraic Systems.” *SIAM J. Sci. Comput.* 19 (5): 1495–1512.

Campbell, S.L., and C.W. Gear. 1995. “The index of general nonlinear DAEs.” *Numer. Math., Springer-Verlag* 72:173–196.

Caracotsios, M., and W. Stewart. 1985. “Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations.” *Computers in Chemicsl Engineering* 9:359–365.

Carver, M.B. 1978. “Efficient integration over discontinuities in ordinary differential equation simulation.” *Mathematics and Computers in Simulation* 20:190–196.

Cheney, W., and D. Kincaid. 1999. *Numerical Mathematics and Computing.* 4th ed. Brooks/Cole Publishing Company.

Chimowitz, E.H., and C.S. Lee. 1985. “Local Thermodynamic Models for High Pressure Process Calculations.” *Computers chem. Engen* Vol. 9, No. 2:195–200.

Choe, Y-S., and W.L. Luyben. 1987. “Rigorous Dynamic Models of Distillation Columns.” *Ind. Eng. Chem. Res.* 26:2158–2161.

Cormen, T.H., C.E. Leiserson, R.L. Rivest, and C. Stein. 2001, September. *Introduction to Algorithms.* 2nd ed. The MIT Press.

Dennis Jr., J.E., and R.B. Schnabel. 1996. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* SIAM Society for Industrial & Applied Mathematics.

Dongarra, J., A. Lumsdaine, and R. Pozo. 1996. SparseLib++ v.1.5d. http://math.nist.gov/sparselib++/. National Institute of Standards and Technology’s, USA.

Dongarra, J., A. Lumsdaine, R. Pozo, and K.A. Remington. 1995. IML++ v.1.2a. http://math.nist.gov/iml++/. Oak Ridge National Laboratory, University of Tennessee, Unieversity of Notre Dame.

Duff, I.S., and J.K. Reid. 1978a. “Algorithm 529: Permutations To Block Triangular Form [F1].” *ACM Transactions on Mathematical Software (TOMS)* 4 (2): 189–192.

———. 1978b. "An implementation of Tarjans algorithm for the block triangularization of a matrix." *ACM Trans. Math. Softw.* 4:137–147.

———. 1996. "The Design of MA48: A Code for the Direct Solution of Sparse Unsymmetric Linear Systems of Equations." *ACM Transactions on Mathematical Software (TOMS)* 22 (2): 187–226.

Eich-Soellner, E., P. Lory, P. Burr, and A. Kroner. 1997. "Stationary and dynamic flowsheeting in the chemical engineering industry." *Surveys on Mathematics for Industry* 7 (1): 1–28.

Endrestøl, G., T. Sira, M. Østenstad, T. Malik, M. Meeg, and J. Thrane. 1989. "Simultanious Computation Within a Sequential Process Simulation Tool." *Moledling, identification and Control* 10 (4): 203–211.

Feehery, W.F., and P.I. Barton. 1996a. "A differentiation-based approach to dynamic simulation and optimization with high-index differential-algebraic equations." In *Computational Differentiation. Techniques, Applications, and Tools*, edited by M. Berz, C. Bischof, G. Corliss, and A. Griewank, 239–253. Philadelphia, PA: SIAM.

———. 1996b. "Dynamic Simulation and Optimization with Inequality Path Constraints." *Computers Chem. Engng.* 20 (Suppl.): S707–S712.

Foerg, W., W. Bach, R.. Stockmann, R.S. Heiersted, P. Paurola, and A.O. Fredheim. 1998. "A New LNG Baseload Process and the Manufacturing of the Main Heat Exchangers." *Paper LNG-12*, p. 14.

Fogler, H.S. 1999. *Elements of Chemical Reaction Engineering.* 3rd ed. Prentice Hall.

Forth, S.A., M. Tadjouddine, J.D. Pryce, and J.K. Reid. 2004. "Jacobian Code Generated by Source Transformation and Vertex Elimination Can Be as Efficient as Hand-Coding." *ACM Transactions on Mathematical Software* 30 (3): 266–299.

Fredheim, A., O. Jørstad, G. Owren, S. Vist, and B. O. Neeraas. 2000. "Coil, a model for simulation of spiral wound LNG heat exchangers." *World Gas Conference, Nice.*

Fredheim, A.O. 1994. "Thermal design of coil-wound LNG heat exchangers, shell-side heat transfer and pressure drop." Ph.D. diss., Norwegian University of Science and Technology.

Geankoplis, C.J. 1993, April. *Transport processes and unit operations.* 3rd ed. Prentice Hall Inc.

Gear, C.W., and O. Østerby. 1984. "Solving Ordinary Differential Equations with Discontinuities." *ACM Transactions on Modeling and Computer Simulation* 10 (1): 23–44.

Giering, R., and T. Kaminski. 1998. "Recipies for adjoint code construction." *ACM Trans. Math. Softw.* 24 (4): 437–474.

Golub, G.H., and C.F. Van Loan. 1996. *Matrix Computation.* 3rd ed. The Johns Hopkins University Press, Baltimore.

Griewank, A. 2000, April. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* SIAM Society for Industrial & Applied Mathematics.

Griewank, A., D. Juedes, and J. Utke. 1996. “ADOL-C: A package for the automatic differentiation of algorithms written in C/C++.” *ACM TOMS* 22 (2): 131–167.

Grini, P.G. 1994. “Flow calorimetry and enthalpy increment measurements for natural gas.” Ph.D. diss., Norwegian University of Science and Technology.

Hairer, E., and G. Wanner. 1996. *Solving Ordinary Differential Equations II Stiff and Differential-Algebraic Problems.* 2nd rev. ed. Springer-Verlag, Berlin, Heidelberg.

Hammer, M., S. Vist, H. Nordhus, I.L. Sperle, G. Owren, and O. Jørstad. 2003. “Dynamic modelling of spiral wound LNG heat exchangers - comparison to experimental results.” *AIChE Spring Meeting, New Orleans.*

Hillestad, M., and T. Hertzberg. 1986. “Dynamic Simulation of Chemical Engineering Systems by the Sequential Modular Approach.” *Modeling, identification and control* 7 (3): 107–127.

Hillestad, M., C. Sørlie, T.F. Anderson, I. Olsen, and T. Hertzberg. 1989. “On Estemating the Error of Local Thermodynamic Models - A General Approach.” *Computers chem. Engen.* Vol. 13, Nr. 7:789–796.

Hindmarsh, A.C. 1980. “LSODE and LSODI, two new initial value ordinary differential equation solvers.” *ACM-Signum Newsletter* 15:10–11.

———. 1983. “ODEPACK, A Systematized Collection of ODE Solvers.” In *Scientific Computing*, edited by R. Stapleman, M. Carver, R. Peskin, W.F. Ames, and R. Vichnevetsky, 55–64. North-Holland publishing company.

Hindmarsh, A.C., P.N. Brown, and G.D. Byrne. 2002. DVODPK v. of 26 April 2002. http://www.netlib.org/ode/.

Hindmarsh, A.C., and A.H. Sherman. 1987. LSODES v.of March 30 1987. http://www.netlib.org/alliant/ode/prog/.

Hlavacek, V. 1977. “Analysis of a complex plant–steady state and transient behavior.” *Computers and Chemical Engineering* 1:75–100.

Holl, P., W. Marquardt, and E.D. Gilles. 1988. “DIVA - A Powerful Tool for Dynamic Process Simulation.” *Computers & Chemical Engineering* 12 (5): 421–426.

Jørstad, O. 1993. “Equation of state for hydrocarbon mixtures.” Ph.D. diss., Norwegian University of Science and Technology.

Köhler, R., A. Gerstlauer, and M. Zeitz. 2001. “Symbolic Preprocessing for Simulation of PDE Models of Chemical Processes.” *Journal of Mathematics and Computers in Simulation* 56 (2): 157–170.

Koup, T.G., E.H. Chimowitz, A. Blonz, and L.F. Stutzman. 1981. "An equation analyzer package for the manipulation of mathematical expressions–I." *Computers Chem. Engng.* 5:151–159.

Kroner, A., P. Holl, W. Marquardt, and E.D. Gilles. 1990. "DIVA - An Open Architecture for Dynamic Simulation." *Computers & Chemical Engineering* 14 (11): 1289–1295.

Laganier, F.S., J-M. Le Lann, X. Joulia, B. Koehret, and M. Morari. 1993. "Simultaneous Modular Dynamic Simulation : Application to Interconnected Distillation Columns." *Computers and Chemical Engineering* 17:287–97.

Locke, M.H., and A.W. Westerberg. 1983. "The ASCEND-II System–A Flowsheeting Application of a Successive Quadratic Programming Methodology." *Computers and Chemical Engineering* 7 (5): 615–630.

Mangold, M., A. Kienle, E.D. Gilles, and K.D. Mohl. 2000. "Nonlinear computation in DIVA – methods and applications." *Chemical Engineering Science* 55:441–454.

Mao, G., and L.R. Petzold. 2002. "Efficient integration over discontinuities for differential-algebraic systems." *Computers & Mathematics with Applications* 43:65–79.

Marquardt, W. 1995. "Numerical Methods for the Simulation of Differential-Algebraic Process Models." *NATO ASI series. Series E, Applied science* 293:41–79.

Martinson, W.S. 2000. "Index and Characteristic Analysis of Partial Differential Equations." Ph.D. diss., Department of Chemical Engineering, Massachusetts Institute of Technology.

Martinson, W.S., and P.I. Barton. 2000. "A Differential index for partial differential-algebraic equations." *SIAM J. Sci. Comput.* 21 (6): 2295–2315.

Mattsson, S.E., and G. Söderlind. 1993. "Index reduction in differential-algebraic equations using dummy derivatives." *SIAM Journal on Scientific Computing* 14 (3): 677–692.

Melaaen, E. 1994. "Dynamic Simulation of The Liquefaction Section in Baseload LNG Plants." Ph.D. diss., Department of Refrigeration Engineering, Norwegian University of Science and Technology.

Melaaen, I.S. 1993. "Probabilistic modelling on the influence of inaccuracies in the thermodynamic properties on process olant design." Ph.D. diss., Department of Refrigeration Engineering, Norwegian University of Science and Technology.

Michelsen, M.L., and J. Mollerup. 1998. *Thermodynamic Models Fundamental and Computional Aspects.* DTU, Technical University of Denmark, Lyngby, Department of Chemical Engineering.

Moe, H.I., H.C. Riksheim, and T. Hertzberg. 1996. "Stiffness Reduction of Gas System Models." *Computers Chem. Engng.* 20 (Suppl.): S901–906.

Neeraas, B.O. 1994. “Condensation of hydrocarbon mixtures in coil-wound LNG heat exchangers, shell-side heat transfer and pressure drop.” Ph.D. diss., Norwegian University of Science and Technology.

Nocedal, J., and S.J. Wright. 1999. *Numerical Optimization.* Springer-Verlag New-York.

Oh, M., and C.C. Pantelides. 1996. “A Modelling and Simulation Language for Combined Lumped and Distributed Parameter Systems.” *Compurers Chem. Engng.* 20 (6/7): 611–633.

Owren, G.A. 1988. “Condensation of multicomponent mixtures.” Ph.D. diss., Norwegian University of Science and Technology.

Pantelides, C.C. 1988. “SPEEDUP-Recent Advances in process simulation.” *Comput. chem. Engng.* 12 (7): 745–155.

Pantelides, C.C., and H.I. Britt. 1995. “Multipurpose Process Modelling Environments.” In *Proc. Conf. on Foundations of Computer Aided Process Design ’94*, edited by L.T. Biegler and M.F. Doherty, 128–141. CACHE Publications.

Pantelides, C.C., D. Gritsis, K.R. Morison, and R.W.H. Sargent. 1988. “The Mathematical Modelling of Transient Systems using Differential-Algebraic Equations.” *Comput. Chem. Engng.* 12 (5): 449–454.

Park, T., and P.I Barton. 1996. “State Event Location in Differential-Algebraic Models.” *ACM Transactions on Modeling and Computer Simulation* 6 (2): 137–165.

Patterson, G.K., and R.B. Tozsa. 1980. “DYNSYL: A general-purpose dynamic simulator for chemical processes.” *Computers and Chemical Engineering* 4:1–20.

Peng, D.Y., and D.B. Robinson. 1976. “A New Two-Constant Equation of State.” *Ind. Eng. Chem. Fund.*, vol. 15.

Perry, R.H., D.W. Green, and J.O. Maloney. 1999. *Perry’s Chemical Engineers’ Handbook.* 7th ed. McGraw-Hill.

Petzold, L.R. 1983. “A Description of DASSL: A Differential/Algebraic System Solver.” In *Scientific Computing*, edited by R. Stapleman, M. Carver, R. Peskin, W.F. Ames, and R. Vichnevetsky, 65–68. North-Holland publishing company.

Petzold, L.R., P.N. Brown, A.C. Hindmarsh, and S. Li. 2000. DASPK v. 3.0. http://www.engineering.ucsb.edu/∼cse/software.html.

Piela, P., R. McKelvey, and A. Westerberg. 1992-1993. “An introduction to the ASCEND modeling system: Its language and interactive environment.” *Journal of Management Information Systems* 9 (3): 91–121.

Piela, P.C., T.G. Epperly, K.M. Westerberg, and A.W. Westerberg. 1991. “ASCEND: An Object-Oriented Computer Environment for Modeling and

Analysis: The Modeling Language." *Computers Chem. Engng.* 15 (1): 53–72.

Ponton, J.W. 1982. "The numerical evaluation of analytical derivatives." *Computers Chem. Engng.* 6:331–333.

Ponton, J.W., and P.J. Gawthrop. 1991. "Systematic Construction of Dynamic Models for Phase Equilibrium Processes." *Computers Chem. Engng.* 15 (12): 803–808.

Saad, Yousef. 2003, April. *Iterative Methods for Sparse Linear Systems.* 2nd ed. Society for Industrial & Applied Mathematics.

Soave, G. 1972. "Equilibrium Constants from a Modified Redlich-Kwong Equation of State." *Chem. Eng. Sci.*, 27 (6): 1197–1203.

Sosonkina, M., D.C.S. Allison, and L.T. Watson. 1998. "Scalable parallel implementations of the GMRES algorithm via Householder reflections." Edited by T.H. Lai, *Proceedings 1998 - International Conference on Parallel Processing.* Los Alamitos, CA, USA: IEEE Comput. Soc, 396–404.

Stockmann, R., W. Förg, M. Bölt, M. Steinbauer, C. Pfeiffer, P. Pentti, A.O. Fredheim, and Ø Sørensen. 1998. Fremgangsmåte for å gjøre en karbonhydrogenrik strøm flytende. Norsk Patent No. 310124.

Tarjan, R.E. 1972. "Depth first search and linear graph algorithms." *SIAM J. Comptg.* 1:146–160.

Tolsma, J., and P.I. Barton. 1999. "Efficient Calculation of Sparse Jacobians." *SIAM Journal on Scientific Computing* 20 (6): 2282–2296.

———. 2000. "DAEPACK: An Open Modeling Environment for Legacy Models." *Ind. Eng. Chem. Res.* 39:1826–1839.

———. 2004. "DAEPACK: A Symbolic and Numeric Library for Open Modeling." *Available at: http://yoric.mit.edu/daepack/daepack.html.*

Tolsma, J., J.A. Clabaugh, and P.I. Barton. 2002. "Symbolic Incorporation of External Procedures into Process Modeling Environments." *Ind. Eng. Chem. Res.* 41:3867–3876.

———. 2004. "ABACUSS II: Advanced Modeling Environment and Embedded Process Simulator." *Available at: http://yoric.mit.edu/abacuss2/abacuss2.html.*

Tolsma, J.E., and P.I. Barton. 1998. "On computational differentiation." *Computers chem. Engng.* 22 (4/5): 475–490.

Trankle, F., A. Gerstlauer, M. Zeitz, and E.D. Gilles. 1997. "Application of the modeling and simulation environment PRoMoT/DIVA to the modeling of distillation processes." *Computers & Chemical Engineering* 21 (Suppl.): S841–S846.

Trankle, F., M. Zeitz, M. Ginkel, and E.D. Gilles. 2000. "PROMOT: a modeling tool for chemical processes." *Mathematical and Computer Modelling of Dynamical Systems* 6 (3): 283–307.

Verwer, J.G., J.G. Blom, and J.M. Sanz-Serna. 1989. “An Adaptive Moving Grid Method for One-Dimensional Systems of Partial Differential Equations.” *Journal of Computational Physics* 82:454–486.

Verwer, J.G., and R.A. Trompert. 1993. “Analysis of Local Uniform Grid Refinement.” *Applied Numerical Mathematics* 13:251–270.

Vist, S., M. Hammer, H. Nordhus, I.L. Sperle, G. Owren, and O. Jørstad. 2003. “Dynamic modelling of spiral wound LNG heat exchangers - model description.” *AIChE Spring Meeting, New Orleans.*

Wagner, Y. 1998. “A Moving Grid Algorithm for a Heat Exchanger with Phase Changes.” *Applied Numerical Mathematics* 28:477–491.

Werma, A. 2000. “An introduction to automatic differentiation.” *Current Science* 78 (7): 804–807.

Zaïm, A. 2002. “Dynamic optimization of an LNG plant, Case study: GL2Z LNG plant in Arzew, Algeria.” Ph.D. diss., RWTH Aachen University.

Appendix A

Thermodynamics

The set definitions in Section 4.1, will be used in this appendix.

A.1 The variables and equations describing the equilibrium

The variables:

$$
\begin{array}{llll}
N_{z_i} & i \in S_{NC} & (NC) & \text{(A.1)} \\
z_i & i \in S_{NC} & (NC) & \text{(A.2)} \\
N_{V,i} & i \in S_{NC} & (NC) & \text{(A.3)} \\
N_{L,i} & i \in S_{NC} & (NC) & \text{(A.4)} \\
T & & (1) & \text{(A.5)} \\
P & & (1) & \text{(A.6)} \\
s_1 & & (1) & \text{(A.7)} \\
s_2 & & (1) & \text{(A.8)} \\
 & & (\mathbf{4NC+4}) &
\end{array}
$$

The equations:

$$
\begin{array}{lll}
z_i = N_{V,i} + N_{L,i}, \quad i \in S_{NC} & (NC) & \text{(A.9)} \\
g_i = g_i(N_{V,j}, N_{L,j}, T, P), \quad i \text{ and } \quad j \in S_{NC} & (NC) & \text{(A.10)} \\
c_i = c_i(N_{V,j}, N_{L,j}, T, P), \quad i \in \{1, 2\} \quad j \in S_{NC} & (2) & \text{(A.11)} \\
z_i = \dfrac{N_{z,i}}{\sum\limits_{j=1}^{NC} N_{z,j}}, \quad i \in S_{NC} & (NC) & \text{(A.12)} \\
 & (\mathbf{3NC+2}) &
\end{array}
$$

The functions, c_i, are general flash specifications equations, and s_i are general specifications.

Specifying the overall composition, s_1 and s_2, gives an solvable set of equations. The general problem is formulated in the following way.

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} g_1 \\ \vdots \\ g_{NC} \\ c_1 \\ c_2 \end{bmatrix} = \mathbf{0}, \qquad \mathbf{X} = \begin{bmatrix} N_{V,1} \\ \vdots \\ N_{V,NC} \\ lnT \\ lnP \end{bmatrix} \tag{A.13}$$

The specifications are given by Michelsen and Mollerup (1998). The specifications are given in dimensionless form, and two sets are shown in Table A.1.

Table A.1: Flash specifications and state functions

Specifications	State functions	
	r_T	r_P
(U,V)	$\frac{1}{RT}(U^{spec} + PV^{spec} - H)$	$\frac{P}{RT}(V - V^{spec})$
(H,P)	$\frac{1}{RT}(H^{spec} - H)$	-

A.2 UV-flash

A.2.1 Newton - Raphson approach

The UV-Flash using a Newton based approach. The formulation is a first order linearization of Equation A.13.

$$\begin{pmatrix} \mathbf{M_{1,g}} & \mathbf{g_T} & \mathbf{g_P} \\ \mathbf{g_T^T} & E_{TT} & E_{TP} \\ \mathbf{g_P^T} & E_{TP} & E_{PP} \end{pmatrix} \begin{pmatrix} \mathbf{\Delta N_V} \\ \Delta lnT \\ \Delta lnP \end{pmatrix} + \begin{pmatrix} \mathbf{g} \\ c_1 = r_T \\ c_2 = r_P \end{pmatrix} = \mathbf{0} \tag{A.14}$$

The deviation vector is:

$$g_i = lny_i + lnf_{V,i}(\mathbf{N_V}, T, P) - lnx_i - lnf_{L,i}(\mathbf{N_L}, T, P) \tag{A.15}$$

For the UV-Flash we use the following functions:

$$r_T = \frac{1}{RT}\left(U^{spec} + PV^{spec} - H\right) \tag{A.16}$$

$$r_P = \frac{P}{RT}\Big(V - V^{spec}\Big) \tag{A.17}$$

The differentials are given as follows:

$$\begin{aligned}
M_{i,g,ij} &= \left(\frac{\partial g_i}{\partial N_{V,j}}\right)_{T,P,N_{V,k}}, \ i \in S_{NC}, \ j \in S_{NC}, \ k \in S_{NC,j} \\
g_{T,i} &= T\Bigg(\left(\frac{\partial lnf_{V,i}}{\partial T}\right)_{P,\mathbf{N_V}} - \left(\frac{\partial lnf_{L,i}}{\partial T}\right)_{P,\mathbf{N_L}}\Bigg), \ i \in S_{NC} \\
g_{P,i} &= P\Bigg(\left(\frac{\partial lnf_{V,i}}{\partial P}\right)_{T,\mathbf{N_V}} - \left(\frac{\partial lnf_{L,i}}{\partial P}\right)_{T,\mathbf{N_L}}\Bigg), \ i \in S_{NC} \\
E_{TT} &= -\frac{C_P}{R}, \ E_{TP} = \frac{P}{R}\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{N_V}}, \ E_{PP} = \frac{P^2}{RT}\left(\frac{\partial V}{\partial P}\right)_{T,\mathbf{N_V}}
\end{aligned} \tag{A.18}$$

Here:

$$C_P = \left(\frac{\partial H}{\partial T}\right)_{P,\mathbf{N_V}} \tag{A.19}$$

$$V = \frac{N_V}{\rho_V} + \frac{N_Z - N_V}{\rho_L} \tag{A.20}$$

$$\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{N_V}} = -\frac{N_V}{\rho_V^2}\left(\frac{\partial \rho_V}{\partial T}\right)_{P,\mathbf{N_V}} - \frac{N_Z - N_V}{\rho_L^2}\left(\frac{\partial \rho_L}{\partial T}\right)_{P,\mathbf{N_L}} \tag{A.21}$$

$$\left(\frac{\partial V}{\partial P}\right)_{T,\mathbf{N_V}} = -\frac{N_V}{\rho_V^2}\left(\frac{\partial \rho_V}{\partial P}\right)_{T,\mathbf{N_V}} - \frac{N_Z - N_V}{\rho_L^2}\left(\frac{\partial \rho_L}{\partial P}\right)_{T,\mathbf{N_L}} \tag{A.22}$$

Note that since the overall composition is specified, the following holds:

$$\mathbf{N_V} \text{ constant} \Leftrightarrow \mathbf{N_L} \text{ constant} \tag{A.23}$$

Writing out the terms:

$$dlnT = \tfrac{1}{T}dT \quad dlnP = \tfrac{1}{P}dP \tag{A.24}$$

$$\begin{aligned}
\left(\frac{\partial r_T}{\partial lnT}\right)_{P,\mathbf{N_V}} &= \frac{\partial}{\partial lnT}\Bigg(\frac{U^{spec} + PV^{spec} - H}{RT}\Bigg)_{P,\mathbf{N_V}} \\
&= \frac{T}{R}\frac{\partial}{\partial T}\Bigg(\frac{U^{spec} + PV^{spec} - H}{T}\Bigg)_{P,\mathbf{N_V}} \\
&= \frac{T}{R}\Bigg(-\frac{1}{T}\left(\frac{\partial H}{\partial T}\right)_{P,\mathbf{N_V}} + \frac{H - U^{spec} - PV^{spec}}{T^2}\Bigg) \\
&= \frac{1}{R}\Bigg(-\left(\frac{\partial H}{\partial T}\right)_{P,\mathbf{N_V}} + \frac{H - U^{spec} - PV^{spec}}{T}\Bigg) \\
&\approx -\frac{1}{R}\left(\frac{\partial H}{\partial T}\right)_{P,\mathbf{N_V}}
\end{aligned} \tag{A.25}$$

$$\begin{aligned}
\left(\frac{\partial r_T}{\partial lnP}\right)_{T,\mathbf{N_V}} &= \frac{\partial}{\partial lnP}\left(\frac{PV^{spec}}{RT} - \frac{H}{RT}\right)_{T,\mathbf{N_V}} \\
&= \frac{P}{TR}\frac{\partial}{\partial P}\left(PV^{spec} - H\right)_{T,\mathbf{N_V}} \\
&= \frac{P}{RT}\left(V^{spec} - \left(\frac{\partial H}{\partial P}\right)_{T,\mathbf{N_V}}\right) \\
&= \frac{P}{RT}\left(V^{spec} - V + T\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{N_V}}\right) \\
&\approx \frac{P}{R}\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{N_V}}
\end{aligned} \tag{A.26}$$

$$\begin{aligned}
\left(\frac{\partial r_T}{\partial N_{V,i}}\right)_{P,T,N_{V,j}} &= \frac{\partial}{\partial N_{V,i}}\left(-\frac{H}{RT}\right)_{P,T,N_{V,j}} \\
&= -\frac{1}{RT}\left(\frac{\partial H}{\partial N_{V,i}}\right)_{P,T,N_{V,j}}, \\
& i \in S_{NC}, \quad j \in S_{NC,i}
\end{aligned} \tag{A.27}$$

$$\begin{aligned}
\left(\frac{\partial r_P}{\partial lnT}\right)_{P,\mathbf{N_V}} &= \frac{\partial}{\partial lnT}\left(\frac{P(V - V^{spec})}{RT}\right)_{P,\mathbf{N_V}} \\
&= \frac{PT}{R}\frac{\partial}{\partial T}\left(\frac{V - V^{spec}}{T}\right)_{P,\mathbf{N_V}} \\
&= \frac{PT}{R}\left(\frac{1}{T}\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{N_V}} - \frac{V - V^{spec}}{T^2}\right) \\
&= \frac{P}{R}\left(\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{N_V}} - \frac{V - V^{spec}}{T}\right) \\
&\approx \frac{P}{R}\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{N_V}}
\end{aligned} \tag{A.28}$$

$$\begin{aligned}
\left(\frac{\partial r_P}{\partial lnP}\right)_{T,\mathbf{N_V}} &= \frac{\partial}{\partial lnP}\left(\frac{P(V - V^{spec})}{RT}\right)_{T,\mathbf{N_V}} \\
&= \frac{P}{TR}\frac{\partial}{\partial P}\left(P(V - V^{spec})\right)_{T,\mathbf{N_V}} \\
&= \frac{P}{RT}\left(V - V^{spec} + P\left(\frac{\partial V}{\partial P}\right)_{T,\mathbf{N_V}}\right) \\
&\approx \frac{P^2}{RT}\left(\frac{\partial V}{\partial P}\right)_{T,\mathbf{N_V}}
\end{aligned} \tag{A.29}$$

$$\begin{aligned}\left(\frac{\partial r_P}{\partial N_{V,i}}\right)_{P,T,N_{V,j}} &= \frac{\partial}{\partial N_{V,i}}\left(\frac{PV}{RT}\right)_{P,T,N_{V,j}} \\ &= \frac{P}{RT}\left(\frac{\partial V}{\partial N_{V,i}}\right)_{P,T,N_{V,j}}, \\ & i \in S_{NC}, \quad j \in S_{NC,i}\end{aligned} \tag{A.30}$$

$$\begin{aligned}\left(\frac{\partial g_i}{\partial lnT}\right)_{P,\mathbf{N_V}} &= \frac{\partial}{\partial lnT}\Big(lnf_{V,i} - lnf_{L,i}\Big)_{P,\mathbf{N_V}} \\ &= T\Bigg(\left(\frac{\partial lnf_{V,i}}{\partial T}\right)_{P,\mathbf{N_V}} - \left(\frac{\partial lnf_{L,i}}{\partial T}\right)_{P,\mathbf{N_L}}\Bigg)\end{aligned} \tag{A.31}$$

$$\begin{aligned}\left(\frac{\partial g_i}{\partial lnP}\right)_{T,\mathbf{N_V}} &= \frac{\partial}{\partial lnP}\Big(lnf_{V,i} - lnf_{L,i}\Big)_{T,\mathbf{N_V}} \\ &= P\Bigg(\left(\frac{\partial lnf_{V,i}}{\partial P}\right)_{T,\mathbf{N_V}} - \left(\frac{\partial lnf_{L,i}}{\partial P}\right)_{T,\mathbf{N_V}}\Bigg)\end{aligned} \tag{A.32}$$

$$\begin{aligned}\left(\frac{\partial g_i}{\partial N_{V,j}}\right)_{P,T,N_{V,k}} =& \frac{\partial}{\partial N_{V,j}}\Big(lny_i + lnf_{V,i} - lnx_i - lnf_{L,i}\Big)_{P,T,N_{V,k}} \\ =& \frac{1}{y_i}\left(\frac{\partial y_i}{\partial N_{V,j}}\right)_{P,T,N_{V,k}} + \left(\frac{\partial lnf_{V,i}}{\partial N_{V,j}}\right)_{P,T,N_{V,k}} \\ &+ \frac{1}{x_i}\left(\frac{\partial x_i}{\partial N_{L,j}}\right)_{P,T,N_{V,k}} + \left(\frac{\partial lnf_{L,i}}{\partial N_{L,j}}\right)_{P,T,N_{V,k}}, \\ & i \in S_{NC}, \quad j \in S_{NC}, \quad k \in S_{NC,j}\end{aligned} \tag{A.33}$$

It is shown by Michelsen and Mollerup (1998) that the following two relations hold, and the matrix generation is simplified.

$$\left(\frac{\partial r_T}{\partial N_{V,i}}\right)_{P,T,N_{V,j}} = \left(\frac{\partial g_i}{\partial lnT}\right)_{P,\mathbf{N_V}}, \quad i \in S_{NC}, \quad j \in S_{NC,i} \tag{A.34}$$

$$\left(\frac{\partial r_P}{\partial N_{V,i}}\right)_{P,T,N_{V,j}} = \left(\frac{\partial g_i}{\partial lnP}\right)_{T,\mathbf{N_V}}, \quad i \in S_{NC}, \quad j \in S_{NC,i} \tag{A.35}$$

A.2.2 The nested loop approach.

To guarantee convergence of the UV-Flash, it must be combined with a nested loop iteration. An objective function is formulated:

$$Q = \frac{1}{T}\Big(G - U^{spec} - PV^{spec}\Big) \tag{A.36}$$

The outer loop variables are given in equation A.37.

$$x = \begin{bmatrix} T \\ P \end{bmatrix} \tag{A.37}$$

$$\begin{aligned}\left(\frac{\partial Q}{\partial T}\right)_{P,\mathbf{z}} &= -\tfrac{1}{T}S - \tfrac{1}{T^2}\Big(G - U^{spec} - PV^{spec}\Big)\\ &= -\tfrac{1}{T^2}TS - \tfrac{1}{T^2}\Big(H - TS - U^{spec} - PV^{spec}\Big)\\ &= -\tfrac{1}{T^2}\Big(H - U^{spec} - PV^{spec}\Big)\end{aligned} \tag{A.38}$$

$$\begin{aligned}\left(\frac{\partial Q}{\partial P}\right)_{T,\mathbf{z}} &= \tfrac{1}{T}\Big(\left(\tfrac{\partial G}{\partial P}\right)_{T,\mathbf{z}} - V^{spec}\Big)\\ &= \tfrac{1}{T}\Big(V - V^{spec}\Big)\end{aligned} \tag{A.39}$$

$$\left(\frac{\partial^2 Q}{\partial T^2}\right)_{P,\mathbf{z}} = \tfrac{2}{T^3}\Big(H - U^{spec} - PV^{spec}\Big) - \tfrac{1}{T^2}\left(\tfrac{\partial H}{\partial T}\right)_{P,\mathbf{z}} \tag{A.40}$$

$$\left(\frac{\partial^2 Q}{\partial P^2}\right)_{T,\mathbf{z}} = \tfrac{1}{T}\left(\tfrac{\partial V}{\partial P}\right)_{T,\mathbf{z}} \tag{A.41}$$

$$\left(\frac{\partial^2 Q}{\partial P \partial T}\right) = -\tfrac{1}{T^2}\Big(V - V^{spec}\Big) + \tfrac{1}{T}\left(\tfrac{\partial V}{\partial T}\right)_{P,\mathbf{z}} \tag{A.42}$$

We need to evaluate the following differentials along the equilibrium.

$$\begin{aligned}\left(\tfrac{\partial H}{\partial T}\right)_{P,\mathbf{z}} &= \left(\frac{\partial h^V}{\partial T}\right)_{P,\mathbf{y}} + \left(\frac{\partial h^L}{\partial T}\right)_{P,\mathbf{x}}\\ &+\sum_j\left(\left(\frac{\partial h^V}{\partial N_{V,j}}\right)_{T,P,\mathbf{y}} - \left(\frac{\partial h^L}{\partial N_{L,j}}\right)_{T,P,\mathbf{x}}\right)\left(\frac{\partial N_{V,j}}{\partial T}\right)_{P,\mathbf{z}}\end{aligned} \tag{A.43}$$

$$\begin{aligned}\left(\tfrac{\partial V}{\partial T}\right)_{P,\mathbf{z}} &= \left(\frac{\partial V^V}{\partial T}\right)_{P,\mathbf{y}} + \left(\frac{\partial V^L}{\partial T}\right)_{P,\mathbf{x}}\\ &+\sum_j\left(\left(\frac{\partial V^V}{\partial N_{V,j}}\right)_{T,P,\mathbf{y}} - \left(\frac{\partial V^L}{\partial N_{L,j}}\right)_{T,P,\mathbf{x}}\right)\left(\frac{\partial N_{V,j}}{\partial T}\right)_{P,\mathbf{z}}\end{aligned} \tag{A.44}$$

$$\begin{aligned}\left(\tfrac{\partial V}{\partial P}\right)_{T,\mathbf{z}} &= \left(\frac{\partial V^V}{\partial P}\right)_{T,\mathbf{y}} + \left(\frac{\partial V^L}{\partial P}\right)_{T,\mathbf{x}}\\ &+\sum_j\left(\left(\frac{\partial V^V}{\partial N_{V,j}}\right)_{T,P,\mathbf{y}} - \left(\frac{\partial V^L}{\partial N_{L,j}}\right)_{T,P,\mathbf{x}}\right)\left(\frac{\partial N_{V,j}}{\partial P}\right)_{T,\mathbf{z}}\end{aligned} \tag{A.45}$$

The gradient:

$$\begin{bmatrix}\left(\frac{\partial Q}{\partial T}\right)\\ \left(\frac{\partial Q}{\partial P}\right)\end{bmatrix} = \frac{1}{T}\begin{bmatrix}-\frac{1}{T}\Big(H - U^{spec} - PV^{spec}\Big)\\ V - V^{spec}\end{bmatrix} \tag{A.46}$$

The Hessian:

$$\begin{bmatrix}\left(\frac{\partial^2 Q}{\partial T^2}\right) & \left(\frac{\partial^2 Q}{\partial T \partial P}\right)\\ \left(\frac{\partial^2 Q}{\partial P \partial T}\right) & \left(\frac{\partial^2 Q}{\partial P^2}\right)\end{bmatrix} =$$

$$\frac{1}{T^2}\begin{bmatrix}\frac{2}{T}\Big(H - U^{spec} - PV^{spec}\Big) - \left(\frac{\partial H}{\partial T}\right)_{P,\mathbf{z}} & V^{spec} - V + T\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{z}}\\ V^{spec} - V + T\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{z}} & T\left(\frac{\partial V}{\partial P}\right)_{T,\mathbf{z}}\end{bmatrix} \tag{A.47}$$

Michelsen and Mollerup (1998) states that the overall problem is indefinite. Splitting the problem into two subproblems as described above, will give an iteration in two convex problems, that always will converge.

That means that the eigenvalues of the Hessian are both negative. To have two negative eigenvalues, the Hessian elements must satisfy the following relations.

$$\begin{aligned} &\left(\frac{\partial^2 Q}{\partial T^2}\right) + \left(\frac{\partial^2 Q}{\partial P^2}\right) < 0 \\ &\left(\frac{\partial^2 Q}{\partial T^2}\right)\left(\frac{\partial^2 Q}{\partial P^2}\right) - \left(\frac{\partial^2 Q}{\partial T \partial P}\right)^2 > 0 \end{aligned} \tag{A.48}$$

Neglecting the deviation terms in the Hessian we get the following:

$$\begin{bmatrix} \left(\frac{\partial^2 Q}{\partial T^2}\right) & \left(\frac{\partial^2 Q}{\partial T \partial P}\right) \\ \left(\frac{\partial^2 Q}{\partial P \partial T}\right) & \left(\frac{\partial^2 Q}{\partial P^2}\right) \end{bmatrix} = \begin{bmatrix} -\frac{1}{T^2}\left(\frac{\partial H}{\partial T}\right)_{P,\mathbf{z}} & \frac{1}{T}\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{z}} \\ \frac{1}{T}\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{z}} & \frac{1}{T}\left(\frac{\partial V}{\partial P}\right)_{T,\mathbf{z}} \end{bmatrix} \tag{A.49}$$

For this Hessian the first condition is clearly satisfied:

$$\left(\frac{\partial^2 Q}{\partial T^2}\right) < 0, \quad \left(\frac{\partial^2 Q}{\partial P^2}\right) < 0 \quad \Rightarrow \quad \left(\frac{\partial^2 Q}{\partial T^2}\right) + \left(\frac{\partial^2 Q}{\partial P^2}\right) < 0 \tag{A.50}$$

The second condition would hold if the following two conditions hold:

$$\begin{aligned} \mathbf{abs}\left[\left(\frac{\partial^2 Q}{\partial T^2}\right)\right] &> \mathbf{abs}\left[\left(\frac{\partial^2 Q}{\partial T \partial P}\right)\right] \\ \mathbf{abs}\left[\left(\frac{\partial^2 Q}{\partial P^2}\right)\right] &> \mathbf{abs}\left[\left(\frac{\partial^2 Q}{\partial T \partial P}\right)\right] \end{aligned} \tag{A.51}$$

This cannot be shown generally, and therefore the eigenvalues of the Hessian must be checked and corrected, to provide a descent Newton search direction.

First we solve the outer loop problem. The problem can be solved by repeatedly solving the convex problem defined above. That is; an unconstrained (*) convex optimization of a nonlinear system in two variables. A SQP approach with line-search is chosen. The inner loop is a PT-Flash. The formal problem description is:

$$\begin{aligned} \underset{T,P}{Min} \quad & Q^* = -Q(T,P) \\ St. \quad & T > 0 \\ & P > 0 \end{aligned} \tag{A.52}$$

The equation set can be treated as non-constrained, since the solution in practice never will lie on the boundary of T and P. Formulate the Newton search direction.

$$Q^*(x_k + p) \approx Q_k^* + p^T \nabla Q_k^* + \tfrac{1}{2} p^T \nabla^2 Q_k^* p = m_k(p) \tag{A.53}$$

$$\begin{aligned} \underset{p}{Min} \quad m_k(p), \quad \nabla m_k(p) = \nabla\left(Q_k^* + p^T \nabla Q_k^* + \tfrac{1}{2} p^T \nabla^2 Q_k^* p \right) &= 0 \\ \nabla Q_k^* + \tfrac{1}{2} p^T \nabla^2 Q_k^* + \tfrac{1}{2} \nabla^2 Q_k^* p &= 0 \end{aligned} \tag{A.54}$$

Using the symmetry:

$$\left(p^T \nabla^2 Q_k^*\right)^T = \nabla^2 Q_k^{*T} p \ = \nabla^2 Q_k^* p \tag{A.55}$$

$$p_k^N = -\nabla^2 Q_k^{*-1} \nabla Q_k^* = -\nabla^2 Q_k^{-1} \nabla Q_k \tag{A.56}$$

Writing out p_k:

$$\begin{aligned} p_k^N &= -\frac{1}{\left(\frac{\partial^2 Q}{\partial T^2}\right)\left(\frac{\partial^2 Q}{\partial P^2}\right) - \left(\frac{\partial^2 Q}{\partial T \partial P}\right)^2} \begin{bmatrix} \left(\frac{\partial^2 Q}{\partial P^2}\right) & -\left(\frac{\partial^2 Q}{\partial P \partial T}\right) \\ -\left(\frac{\partial^2 Q}{\partial T \partial P}\right) & \left(\frac{\partial^2 Q}{\partial T^2}\right) \end{bmatrix} \begin{bmatrix} \left(\frac{\partial Q}{\partial T}\right) \\ \left(\frac{\partial Q}{\partial P}\right) \end{bmatrix} \\ &= -\frac{1}{\left(\frac{\partial^2 Q}{\partial T^2}\right)\left(\frac{\partial^2 Q}{\partial P^2}\right) - \left(\frac{\partial^2 Q}{\partial T \partial P}\right)^2} \begin{bmatrix} \left(\frac{\partial^2 Q}{\partial P^2}\right)\left(\frac{\partial Q}{\partial T}\right) - \left(\frac{\partial^2 Q}{\partial P \partial T}\right)\left(\frac{\partial Q}{\partial P}\right) \\ -\left(\frac{\partial^2 Q}{\partial T \partial P}\right)\left(\frac{\partial Q}{\partial T}\right) + \left(\frac{\partial^2 Q}{\partial T^2}\right)\left(\frac{\partial Q}{\partial P}\right) \end{bmatrix} \end{aligned} \tag{A.57}$$

We demand that the step satisfies the Wolfe condition, see Nocedal and Wright (1999), that is;

$$\begin{array}{rll} Q^*(x_k + \alpha_k p_k) & \leq Q^*(x_k) + c_1 \alpha_k \nabla Q_k^{*T} p_k & (a) \\ \nabla Q^*(x_k + \alpha_k p_k)^T p_k & \geq c_2 \nabla Q_k^{*T} p_k & (b) \end{array} \tag{A.58}$$

Due to the evaluation cost relation between the function evaluation, and the differential evaluation, only a simple line search is used. A backtracking algorithm that must satisfy only the first Wolfe criterion is used. The parameters chosen are $\alpha_0 = 1$, $c_1 = 0.001$ the backtracking parameter is set to 0.75.

A.3 Linearizations

The equations describing the equilibrium are given in Section A.1. The intention here is to use these equations to get the implicitly defined partial differentials of the internal flash variables with respect to specification variables. The equations used are restated for readability.

The internal flash variables are defined in Equation A.59.

$$\mathbf{X} = \begin{bmatrix} N_{V,1} \\ \vdots \\ N_{V,NC} \\ lnT \\ lnP \end{bmatrix} \tag{A.59}$$

The specification variables are given in Equation A.60.

$$\mathbf{S} = \begin{bmatrix} N_{z_1}^* \\ \vdots \\ N_{z_{NC}}^* \\ s_1 \\ s_2 \end{bmatrix} \tag{A.60}$$

Some auxiliary vectors are defined in Equation A.61.

$$\hat{\mathbf{S}} = \begin{bmatrix} z_1 \\ \vdots \\ z_{NC} \\ s_1 \\ s_2 \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_{NC} \end{bmatrix}, \quad \mathbf{N}^*_{\mathbf{z}} = \begin{bmatrix} N^*_{z_1} \\ \vdots \\ N^*_{z_{NC}} \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \tag{A.61}$$

The asterisk notation means normalized mole numbers. The definition is given in Equation A.62.

$$\sum_{j=1}^{NC} N^*_{z_j} = 1, \text{ that is; } N^*_{z_j} = \frac{N_{z_j}}{\sum\limits_{j=1}^{NC} N_{z_j}} \tag{A.62}$$

The asterisk superscript indicates that the molar content is a scaled value, and the vector elements sum to one.

The function vector describing the flash was defined in Equation A.13, given a set of specifications. The function is restated in Equation A.63.

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} g_1 \\ \vdots \\ g_{NC} \\ c_1 \\ c_2 \end{bmatrix} \tag{A.63}$$

The internal flash variables depend on the flash specifications, and therefore $\mathbf{X} = \mathbf{X}(\mathbf{S})$. The relation between the phase normalized holdup and the composition is:

$$\mathbf{z} = \mathbf{N_V} + \mathbf{N_L} \tag{A.64}$$

Equation A.64 is used to substitute $\mathbf{N_L}$ out of the flash system, Equation A.63. The flash equations is therefore also a function of $\hat{\mathbf{S}} = \hat{\mathbf{S}}(\mathbf{S})$.

The equilibrium is then described in the following form:

$$\mathbf{F}(\mathbf{S}) = \mathbf{F}(\mathbf{X}(\mathbf{S}), \hat{\mathbf{S}}(\mathbf{S})) = \begin{bmatrix} g_1(\mathbf{X}(\mathbf{S}), \hat{\mathbf{S}}(\mathbf{S})) \\ \vdots \\ g_{NC}(\mathbf{X}(\mathbf{S}), \hat{\mathbf{S}}(\mathbf{S})) \\ c_1(\mathbf{X}(\mathbf{S}), \hat{\mathbf{S}}(\mathbf{S})) \\ c_2(\mathbf{X}(\mathbf{S}), \hat{\mathbf{S}}(\mathbf{S})) \end{bmatrix}, \tag{A.65}$$

The direct linearization of $\mathbf{F}$ around $\mathbf{X_0}, \mathbf{\hat{S}_0} = \mathbf{X}(\mathbf{S_0}), \mathbf{\hat{S}}(\mathbf{S_0})$ yield Equation A.66, using the chain rule of differentiation.

$$\begin{aligned}\mathbf{F}(\mathbf{S}) \approx& \mathbf{G}(\mathbf{S}) = \mathbf{F}(\mathbf{S_0}) + \nabla_{\mathbf{S}}\mathbf{F}^T_{\mathbf{S_0}}\Delta\mathbf{S} \\ \nabla_{\mathbf{S}}\mathbf{F}^T_{\mathbf{S_0}} =& \nabla_{\mathbf{X}}\mathbf{F}^T_{\mathbf{S_0}}\nabla_{\mathbf{S}}\mathbf{X}^T + \nabla_{\mathbf{\hat{S}}}\mathbf{F}^T_{\mathbf{S_0}}\nabla_{\mathbf{S}}\mathbf{\hat{S}}^T\end{aligned} \tag{A.66}$$

Where $\nabla_{\mathbf{X}}\mathbf{F}^T$ is recognized as the Jacobian matrix used in the internal solution of the flash problem. $\nabla_{\mathbf{S}}\mathbf{X}^T$ is the wanted differential solution matrix. The entries of both $\nabla_{\mathbf{X}}\mathbf{F}^T$ and $\nabla_{\mathbf{\hat{S}}}\mathbf{F}^T$ are shown below. The $\nabla_{\mathbf{S}}\mathbf{\hat{S}}^T$ is given below in Equation A.67.

$$\nabla_{\mathbf{S}}\mathbf{\hat{S}}^T = \begin{bmatrix} \nabla_{\mathbf{N_z^*}}\mathbf{z}^T & \mathbf{0} \\ \nabla_{\mathbf{N_z^*}}\mathbf{s}^T & \mathbf{I_2} \end{bmatrix} \tag{A.67}$$

It is seen from Equation A.66, that if the first order Taylor expansion is performed in an equilibrium point, that $\nabla_{\mathbf{S}}\mathbf{F}^T_{\mathbf{S_0}}$ should be a zero matrix. This is shown in Equation A.68 and Equation A.69.

$$\mathbf{G}(\mathbf{S}) = \mathbf{F}(\mathbf{S_0}) + \nabla_{\mathbf{S}}\mathbf{F}^T_{\mathbf{S_0}}\Delta\mathbf{S} = \nabla_{\mathbf{S}}\mathbf{F}^T_{\mathbf{S_0}}\Delta\mathbf{S} \tag{A.68}$$

Requiring the linear function to be zero, for all changes in $\mathbf{S}$.

$$\mathbf{G}(\mathbf{S}) = \nabla_{\mathbf{S}}\mathbf{F}^T_{\mathbf{S_0}}\Delta\mathbf{S} = \mathbf{0} \Rightarrow \nabla_{\mathbf{S}}\mathbf{F}^T_{\mathbf{S_0}} = \mathbf{0} \tag{A.69}$$

This gives the following relation for $\nabla_{\mathbf{S}}\mathbf{X}^T$:

$$\nabla_{\mathbf{S}}\mathbf{X}^T = -(\nabla_{\mathbf{X}}\mathbf{F}^T_{\mathbf{S_0}})^{-1}\nabla_{\mathbf{\hat{S}}}\mathbf{F}^T_{\mathbf{S_0}}\nabla_{\mathbf{S}}\mathbf{\hat{S}}^T \tag{A.70}$$

The linearization matrices are given new names in Equation A.71.

$$\begin{aligned}\mathbf{G}(\mathbf{S}) =& (\mathbf{M_1}\mathbf{N_1} + \mathbf{M_2}\mathbf{N_2})\Delta\mathbf{S} \\ \mathbf{M_1} =& \nabla_{\mathbf{X}}\mathbf{F}^T_{\mathbf{S_0}}, \qquad \mathbf{N_1} = \nabla_{\mathbf{S}}\mathbf{X}^T, \qquad \mathbf{M_2} = \nabla_{\mathbf{\hat{S}}}\mathbf{F}^T_{\mathbf{S_0}}, \qquad \mathbf{N_2} = \nabla_{\mathbf{S}}\mathbf{\hat{S}}^T\end{aligned} \tag{A.71}$$

The form of the four matrices in Equation A.71 matrices are stated below. All matrices are square with dimension NC+2. The derivation of the matrices entries will be shown afterwards.

$$\begin{aligned}&\mathbf{M_1} = \\ &\begin{bmatrix} \left(\frac{\partial g_1}{\partial N_{V,1}}\right)_{T,P,N_{V,j}} & \cdots & \left(\frac{\partial g_1}{\partial N_{V,NC}}\right)_{T,P,N_{V,j}} & \left(\frac{\partial g_1}{\partial lnT}\right)_{P,\mathbf{N_V}} & \left(\frac{\partial g_1}{\partial lnP}\right)_{T,\mathbf{N_V}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \left(\frac{\partial g_{NC}}{\partial N_{V,1}}\right)_{T,P,N_{V,j}} & \cdots & \left(\frac{\partial g_{NC}}{\partial N_{V,NC}}\right)_{T,P,N_{V,j}} & \left(\frac{\partial g_{NC}}{\partial lnT}\right)_{P,\mathbf{N_V}} & \left(\frac{\partial g_{NC}}{\partial lnP}\right)_{T,\mathbf{N_V}} \\ \left(\frac{\partial c_1}{\partial N_{V,1}}\right)_{T,P,N_{V,j}} & \cdots & \left(\frac{\partial c_1}{\partial N_{V,NC}}\right)_{T,P,N_{V,j}} & \left(\frac{\partial c_1}{\partial lnT}\right)_{P,\mathbf{N_V}} & \left(\frac{\partial c_1}{\partial lnP}\right)_{T,\mathbf{N_V}} \\ \left(\frac{\partial c_2}{\partial N_{V,1}}\right)_{T,P,N_{V,j}} & \cdots & \left(\frac{\partial c_2}{\partial N_{V,NC}}\right)_{T,P,N_{V,j}} & \left(\frac{\partial c_2}{\partial lnT}\right)_{P,\mathbf{N_V}} & \left(\frac{\partial c_2}{\partial lnP}\right)_{T,\mathbf{N_V}} \end{bmatrix}, \\ &j \in S_{NC,p}\end{aligned} \tag{A.72}$$

$$\mathbf{N_1} = \begin{bmatrix} \left(\frac{\partial N_{V,1}}{\partial N_{z_1}}\right)_{\mathbf{s},N_{z_j}} & \cdots & \left(\frac{\partial N_{V,1}}{\partial N_{z_{NC}}}\right)_{\mathbf{s},N_{z_j}} & \left(\frac{\partial N_{V,1}}{\partial s_1}\right)_{s_2,\mathbf{N_z}} & \left(\frac{\partial N_{V,1}}{\partial s_2}\right)_{s_1,\mathbf{N_z}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \left(\frac{\partial N_{V,NC}}{\partial N_{z_1}}\right)_{\mathbf{s},N_{z_j}} & \cdots & \left(\frac{\partial N_{V,NC}}{\partial N_{z_{NC}}}\right)_{\mathbf{s},N_{z_j}} & \left(\frac{\partial N_{V,NC}}{\partial s_1}\right)_{s_1,\mathbf{N_z}} & \left(\frac{\partial N_{V,NC}}{\partial s_2}\right)_{s_1,\mathbf{N_z}} \\ \left(\frac{\partial lnT}{\partial N_{z_1}}\right)_{\mathbf{s},N_{z_j}} & \cdots & \left(\frac{\partial lnT}{\partial N_{z_{NC}}}\right)_{\mathbf{s},N_{z_j}} & \left(\frac{\partial lnT}{\partial s_1}\right)_{s_2,\mathbf{N_z}} & \left(\frac{\partial lnT}{\partial s_2}\right)_{s_1,\mathbf{N_z}} \\ \left(\frac{\partial lnP}{\partial N_{z_1}}\right)_{\mathbf{s},N_{z_j}} & \cdots & \left(\frac{\partial lnP}{\partial N_{z_{NC}}}\right)_{\mathbf{s},N_{z_j}} & \left(\frac{\partial lnP}{\partial s_1}\right)_{s_2,\mathbf{N_z}} & \left(\frac{\partial lnP}{\partial s_2}\right)_{s_1,\mathbf{N_z}} \end{bmatrix}, \quad j \in S_{NC,p} \tag{A.73}$$

$$\mathbf{M_2} = \begin{bmatrix} \left(\frac{\partial g_1}{\partial z_1}\right)_{s_1,s_2,z_j} & \cdots & \left(\frac{\partial g_1}{\partial z_{NC}}\right)_{s_1,s_2,z_j} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \left(\frac{\partial g_{NC}}{\partial z_1}\right)_{s_1,s_2,z_j} & \cdots & \left(\frac{\partial g_{NC}}{\partial N_{L,NC}}\right)_{s_1,s_2,z_j} & 0 & 0 \\ \left(\frac{\partial c_1}{\partial z_1}\right)_{s_1,s_2,z_j} & \cdots & \left(\frac{\partial c_1}{\partial z_{NC}}\right)_{s_1,s_2,z_j} & \left(\frac{\partial c_1}{\partial s_1}\right)_{s_2,\mathbf{z}} & \left(\frac{\partial c_1}{\partial s_2}\right)_{s_1,\mathbf{z}} \\ \left(\frac{\partial c_2}{\partial z_1}\right)_{s_1,s_2,N_j} & \cdots & \left(\frac{\partial c_2}{\partial z_{NC}}\right)_{s_1,s_2,N_j} & \left(\frac{\partial c_2}{\partial s_1}\right)_{s_2,\mathbf{z}} & \left(\frac{\partial c_2}{\partial s_2}\right)_{s_1,\mathbf{z}} \end{bmatrix}, \quad j \in S_{NC,p} \tag{A.74}$$

$$\mathbf{N_2} = \begin{bmatrix} \left(\frac{\partial z_1}{\partial N_{z_1}}\right)_{N_{z_j}} & \cdots & \left(\frac{\partial z_1}{\partial N_{z_{NC}}}\right)_{N_{z_j}} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \left(\frac{\partial z_{NC}}{\partial N_{z_1}}\right)_{N_{z_j}} & \cdots & \left(\frac{\partial z_{NC}}{\partial N_{z_{NC}}}\right)_{N_{z_j}} & 0 & 0 \\ \left(\frac{\partial s_1}{\partial N_{z_1}}\right)_{N_{z_j}} & \cdots & \left(\frac{\partial s_1}{\partial N_{z_{NC}}}\right)_{N_{z_j}} & 1 & 0 \\ \left(\frac{\partial s_2}{\partial N_{z_1}}\right)_{N_{z_j}} & \cdots & \left(\frac{\partial s_2}{\partial N_{z_{NC}}}\right)_{N_{z_j}} & 0 & 1 \end{bmatrix}, \quad j \in S_{NC,p} \tag{A.75}$$

Sub-matrices in the linearization

For reference the matrices $\mathbf{M_1}$, $\mathbf{M_2}$, and $\mathbf{N_2}$ are defined to have the following sub-matrices.

$$\mathbf{M_1} = \begin{bmatrix} \mathbf{M_{1,g}} & \mathbf{g_T} & \mathbf{g_P} \\ \mathbf{c^T_{1,N_z}} & c_{1,T} & c_{1,P} \\ \mathbf{c^T_{2,N_z}} & c_{2,T} & c_{2,P} \end{bmatrix} \tag{A.76}$$

$$\mathbf{M_2} = \begin{bmatrix} \mathbf{M_{2,g}} & \mathbf{0} & \mathbf{0} \\ \mathbf{c^T_{1,l,N_z}} & c_{11} & c_{12} \\ \mathbf{c^T_{2,l,N_z}} & c_{21} & c_{22} \end{bmatrix} \tag{A.77}$$

$$\mathbf{N_2} = \begin{bmatrix} \mathbf{N_{2,g}} & \mathbf{0} & \mathbf{0} \\ \mathbf{s_1^T} & 1 & 0 \\ \mathbf{s_2^T} & 0 & 1 \end{bmatrix} \tag{A.78}$$

A.3.1 The matrix entries

The $\mathbf{M_1}$ and $\mathbf{M_2}$ matrice entries are derived below. First the differentials for Gibbs functions are derived. Thereafter the flash specifications are differentiated. Both specification sets given in Table A.1 are differentiated. If one of the flash specifications is P or T, the entire system must be reduced accordingly,

Temperature partial differentials of the Gibbs functions

The relation $d\mathbf{N_V} = -d\mathbf{N_L}$ is used.

$$\begin{aligned}\left(\frac{\partial g_i}{\partial lnT}\right)_{P,\mathbf{N_z}} =& \left(\frac{1}{N_{V,i}} + \frac{1}{N_{L,i}}\right)\left(\frac{\partial N_{V,i}}{\partial lnT}\right)_{P,N_{z,j}} \\ & - \left(\frac{1}{N_V} + \frac{1}{N_L}\right)\sum_l \left(\frac{\partial N_{V,l}}{\partial lnT}\right)_{P,N_{z,j}} \\ +\sum_l \Bigg(\left(\frac{\partial ln\ f_{V,i}}{\partial N_{V,l}}\right)_{T,P,N_{V,k}} & + \left(\frac{\partial ln\ f_{L,i}}{\partial N_{L,l}}\right)_{T,P,N_{V,k}}\Bigg)\left(\frac{\partial N_{V,l}}{\partial lnT}\right)_{P,\mathbf{N_z}} \\ & + \left(\frac{\partial ln\ f_{V,i}}{\partial lnT}\right)_{P,N_{V,l}} - \left(\frac{\partial ln\ f_{L,i}}{\partial lnT}\right)_{P,N_{L,l}}, \\ & i \in S_{NC}, \quad l \in S_{NC}, \quad k \in S_{NC,l}, \quad j \in S_{NC}\end{aligned} \tag{A.79}$$

This can then be assigned the sub-matrices in Equation A.76.

$$g_{T,i} = -\left(\frac{\partial ln\ f_{V,i}}{\partial lnT}\right)_{P,N_{V,l}} + \left(\frac{\partial ln\ f_{L,i}}{\partial lnT}\right)_{P,N_{L,l}}, \quad i \in S_{NC}, \quad l \in S_{NC} \tag{A.80}$$

Off-diagonal entries:

$$\begin{aligned}M_{1,g,i,j} =& -\left(\frac{1}{N_V} + \frac{1}{N_L}\right) + \left(\frac{\partial ln\ f_{V,i}}{\partial N_{V,j}}\right)_{T,P,N_{V,k}} \\ & + \left(\frac{\partial ln\ f_{L,i}}{\partial N_{L,j}}\right)_{T,P,N_{V,k}}, \quad i,j \in S_{NC}, j \neq i, \quad k \in S_{NC,j}\end{aligned} \tag{A.81}$$

Diagonal entries:

$$\begin{aligned}M_{1,g,i,i} =& \left(\frac{1}{N_{V,i}} + \frac{1}{N_{L,i}}\right) - \left(\frac{1}{N_V} + \frac{1}{N_L}\right) \\ & + \left(\frac{\partial ln\ f_{V,i}}{\partial N_{V,i}}\right)_{T,P,N_{V,k}} + \left(\frac{\partial ln\ f_{L,i}}{\partial N_{L,i}}\right)_{T,P,N_{V,k}}, \\ & i \in S_{NC}, \quad k \in S_{NC,i}\end{aligned} \tag{A.82}$$

Pressure partial differentials of the Gibbs functions

The relation $d\mathbf{N_V} = -d\mathbf{N_L}$ is used.

$$\begin{aligned}
\left(\frac{\partial g_i}{\partial lnP}\right)_{T,\mathbf{N_z}} =& \left(\frac{1}{N_{V,i}} + \frac{1}{N_{L,i}}\right)\left(\frac{\partial N_{V,i}}{\partial lnP}\right)_{T,N_{z,j}} \\
& - \left(\frac{1}{N_V} + \frac{1}{N_L}\right)\sum_l \left(\frac{\partial N_{V,l}}{\partial lnP}\right)_{T,N_{z,j}} \\
+\sum_l \Bigg(\left(\frac{\partial ln\ f_{V,i}}{\partial N_{V,l}}\right)_{T,P,N_{V,k}} &+ \left(\frac{\partial ln\ f_{L,i}}{\partial N_{L,l}}\right)_{T,P,N_{V,k}}\Bigg)\left(\frac{\partial N_{V,l}}{\partial lnP}\right)_{T,\mathbf{N_z}} \\
& + \left(\frac{\partial ln\ f_{V,i}}{\partial lnP}\right)_{T,N_{V,l}} - \left(\frac{\partial ln\ f_{L,i}}{\partial lnP}\right)_{T,N_{L,l}}, \\
& i \in S_{NC}, \quad l \in S_{NC}, \quad k \in S_{NC,l}, \quad j \in S_{NC}
\end{aligned} \tag{A.83}$$

The other terms restate the elements of $M_{1,g}$.

$$g_{P,i} = -\left(\frac{\partial ln\ f_{V,i}}{\partial lnP}\right)_{T,N_{V,l}} - \left(\frac{\partial ln\ f_{L,i}}{\partial lnP}\right)_{T,N_{L,l}}, \quad i \in S_{NC}, \quad l \in S_{NC} \tag{A.84}$$

Compositional partial differentials of the Gibbs functions

The relation $d\mathbf{N_V} + d\mathbf{N_L} = d\mathbf{z}$ is used.

$$\begin{aligned}
\left(\frac{\partial g_i}{\partial N_{z,j}}\right)_{T,P,N_{z,m}} =& \left(\frac{1}{N_{V,i}} + \frac{1}{N_{L,i}}\right)\left(\frac{\partial N_{V,i}}{\partial N_{z,j}}\right)_{T,P,N_{z,m}} \\
& - \left(\frac{1}{N_V} + \frac{1}{N_L}\right)\sum_l \left(\frac{\partial N_{V,l}}{\partial N_{z,j}}\right)_{T,P,N_{z,m}} \\
+\sum_l \Bigg(\left(\frac{\partial ln\ f_{V,i}}{\partial N_{V,l}}\right)_{T,P,N_{V,k}} &+ \left(\frac{\partial ln\ f_{L,i}}{\partial N_{L,l}}\right)_{T,P,N_{V,k}}\Bigg)\left(\frac{\partial N_{V,i}}{\partial N_{z,l}}\right)_{T,P,N_{z,m}} \\
& - \frac{1}{N_{L,i}}\left(\frac{\partial z_i}{\partial N_{z,j}}\right)_{N_{z,m}} \\
& - \sum_l \left(\frac{\partial ln\ f_{L,i}}{\partial N_{L,l}}\right)_{T,P,N_{V,k}}\left(\frac{\partial z_l}{\partial N_{z,j}}\right)_{N_{z,m}}, \\
& i \in S_{NC}, \quad j \in S_{NC,i}, \quad l \in S_{NC}, \\
& k \in S_{NC,l}, \quad m \in S_{NC,j}
\end{aligned} \tag{A.85}$$

Off-diagonal entries:

$$M_{2,g,i,j} = -\left(\frac{\partial ln\ f_{L,i}}{\partial N_{L,j}}\right)_{T,P,N_{V,k}} \quad i \in S_{NC}, \quad k \in S_{NC,j} \tag{A.86}$$

Diagonal entries:

$$M_{2,g,i,i} = -\frac{1}{N_{L,i}} - \left(\frac{\partial ln\ f_{L,i}}{\partial N_{L,i}}\right)_{T,P,N_{V,l}}, \quad i \in S_{NC}, \quad l \in S_{NC,i} \tag{A.87}$$

Temperature differentials of the flash specifications functions

$$\left(\frac{\partial r_{T,(U,V)}}{\partial lnT}\right)_{P,\mathbf{N_z}} = -\frac{1}{R}\left(\frac{\partial H}{\partial T}\right)_{P,\mathbf{N_z}} = \left(\frac{\partial r_{T,(H,P)}}{\partial lnT}\right)_{P,\mathbf{N_z}} \tag{A.88}$$

Writing out the differential, and using the relations developed in Section A.2.1.

$$\begin{aligned} -\frac{1}{R}\left(\frac{\partial H}{\partial T}\right)_{P,\mathbf{N_z}} &= -\frac{1}{RT}\nabla_{\mathbf{N_V}} \cdot H \left(\frac{\partial \mathbf{N_V}}{\partial lnT}\right)_{P,\mathbf{N_z}} - \frac{1}{R}\left(\frac{\partial H}{\partial T}\right)_{P,\mathbf{N_V}} \\ &= \mathbf{g_T}^T \left(\frac{\partial \mathbf{N_V}}{\partial lnT}\right)_{P,\mathbf{N_z}} + E_{TT} \end{aligned} \tag{A.89}$$

$$\begin{aligned} \left(\frac{\partial r_{P,(U,V)}}{\partial lnT}\right)_{P,\mathbf{N_z}} &= \frac{P}{R}\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{N_z}} \\ &= \frac{P}{RT}\nabla_{\mathbf{N_V}} \cdot V \left(\frac{\partial \mathbf{N_V}}{\partial lnT}\right)_{P,\mathbf{N_z}} + \frac{P}{R}\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{N_V}} \\ &= \mathbf{g_P}^T \left(\frac{\partial \mathbf{N_V}}{\partial lnT}\right)_{P,\mathbf{N_z}} + E_{TP} \end{aligned} \tag{A.90}$$

Pressure differentials of the flash specifications functions

The pressure differential of the HP-system is not needed, since the pressure is known.

$$\begin{aligned} \left(\frac{\partial r_{T,(U,V)}}{\partial lnP}\right)_{T,\mathbf{N_z}} &= \frac{1}{RT}\left(\frac{\partial \left(PV^{spec} - H\right)}{\partial lnP}\right)_{T,\mathbf{N_z}} \\ &= -\frac{1}{RT}\nabla_{\mathbf{N_V}} \cdot H \left(\frac{\partial \mathbf{N_V}}{\partial lnP}\right)_{T,\mathbf{N_z}} + \frac{P}{R}\left(\frac{\partial V}{\partial T}\right)_{P,\mathbf{N_V}} \\ &= \mathbf{g_T}^T \left(\frac{\partial \mathbf{N_V}}{\partial lnP}\right)_{T,\mathbf{N_z}} + E_{TP} \end{aligned} \tag{A.91}$$

$$\begin{aligned} \left(\frac{\partial r_{P,(U,V)}}{\partial lnP}\right)_{T,\mathbf{N_z}} &= \frac{P^2}{RT}\left(\frac{\partial V}{\partial P}\right)_{T,\mathbf{N_z}} \\ &= \frac{P}{RT}\nabla_{\mathbf{N_V}} \cdot V \left(\frac{\partial \mathbf{N_V}}{\partial lnP}\right)_{T,\mathbf{N_z}} + \frac{P^2}{RT}\left(\frac{\partial V}{\partial P}\right)_{T,\mathbf{N_V}} \\ &= \mathbf{g_P}^T \left(\frac{\partial \mathbf{N_V}}{\partial lnP}\right)_{T,\mathbf{N_z}} + E_{PP} \end{aligned} \tag{A.92}$$

Compositional differentials of the flash specifications functions

$$\begin{aligned}\left(\frac{\partial r_{T,(U,V)}}{\partial N_{z,i}}\right)_{T,P,N_{z,j}} &= -\frac{1}{RT}\left(\frac{\partial H}{\partial N_{z,i}}\right)_{T,P,N_{z,j}} \\ &= \left(\frac{\partial r_{T,(H,P)}}{\partial N_{z,i}}\right)_{T,P,N_{z,j}}, \\ &i \in S_{NC}, \quad j \in S_{NC,i}\end{aligned} \tag{A.93}$$

$$\begin{aligned}-\frac{1}{RT}\left(\frac{\partial H}{\partial N_{z,i}}\right)_{T,P,N_{z,j}} &= -\frac{1}{RT}\nabla_{\mathbf{N_V}}\cdot H\left(\frac{\partial \mathbf{N_V}}{\partial N_{z,i}}\right)_{T,P,N_{z,j}} \\ &= \mathbf{g_T}^T\left(\frac{\partial \mathbf{N_V}}{\partial N_{z,i}}\right)_{T,P,N_{z,j}} \\ &\quad -\frac{(1-w)}{RT}\nabla_{\mathbf{N_L}}\cdot h_L\left(\frac{\partial \mathbf{z}}{\partial N_{z,i}}\right)_{N_{z,j}}, \\ &i \in S_{NC}, \quad j \in S_{NC,i}\end{aligned} \tag{A.94}$$

From this it is seen:

$$\mathbf{c^T_{1,l,N_z}} = -\frac{(1-w)}{RT}\nabla_{\mathbf{N_L}}\cdot h_L \tag{A.95}$$

where $\mathbf{c^T_{1,l,N_z}}$ is a row vector of the $\mathbf{M_2}$ matrix, as defined in Equation A.77.

$$\begin{aligned}\left(\frac{\partial r_{P,(U,V)}}{\partial N_{z,i}}\right)_{T,P,N_{z,j}} &= \frac{P}{RT}\left(\frac{\partial V}{\partial N_{z,i}}\right)_{T,P,N_{z,j}} \\ &= \frac{P}{RT}\nabla_{\mathbf{N_V}}\cdot V\left(\frac{\partial \mathbf{N_V}}{\partial N_{z,i}}\right)_{T,P,N_{z,j}} \\ &= \mathbf{g_P}^T\left(\frac{\partial \mathbf{N_V}}{\partial N_{z,i}}\right)_{T,P,N_{z,j}} \\ &\quad -\frac{(1-w)P}{RT\rho_L^2}\nabla_{\mathbf{N_L}}\cdot \rho_L\left(\frac{\partial \mathbf{z}}{\partial N_{z,i}}\right)_{T,P,N_{z,j}}\end{aligned} \tag{A.96}$$

From this it is seen:

$$\mathbf{c^T_{2,l,N_z}} = -\frac{(1-w)P}{RT\rho_L^2}\nabla_{\mathbf{N_L}}\cdot \rho_L \tag{A.97}$$

where $\mathbf{c^T_{2,l,N_z}}$ is a row vector of the $\mathbf{M_2}$ matrix, as defined in Equation A.77.

The differentiation of the flash specifications functions with respect to the specifications

$$c_{11} = \left(\frac{\partial r_{T,(U,V)}}{\partial U^{spec}}\right)_{s_2=V^{spec},\mathbf{N_z}} = \frac{1}{RT} = \left(\frac{\partial r_{T,(H,P)}}{\partial H^{spec}}\right)_{s_2=P,\mathbf{N_z}} \tag{A.98}$$

$$c_{12} = \left(\frac{\partial r_{T,(U,V)}}{\partial V^{spec}}\right)_{s_1=U^{spec},\mathbf{N_z}} = \frac{P}{RT} \tag{A.99}$$

$$c_{21} = \left(\frac{\partial r_{P,(U,V)}}{\partial U^{spec}}\right)_{s_2=V^{spec},\mathbf{N_z}} = 0 \tag{A.100}$$

$$c_{22} = \left(\frac{\partial r_{P,(U,V)}}{\partial V^{spec}}\right)_{s_2=U^{spec},\mathbf{N_z}} = -\frac{P}{RT} \tag{A.101}$$

The differentiation of the flash specifications with respect to the overall holdup

These entries are found in $\mathbf{N_2}$.

$$s_{1,i} = \left(\frac{\partial U^{spec}}{\partial N_{z_i}}\right)_{s_2=V^{spec},N_{z_j}} = \left(\frac{\partial H^{spec}}{\partial N_{z_i}}\right)_{s_2=P,N_{z_j}} = 0 \tag{A.102}$$

$$s_{2,i} = \left(\frac{\partial V^{spec}}{\partial N_{z_i}}\right)_{s_1=U^{spec},N_{z_j}} = V^{spec} \tag{A.103}$$

Appendix B

Mathematics

This chapter defines some mathematical relations used in the thesis.

B.1 Linear implicit integration

The 1-stage Rosenbrock method will here be derived, and it will be shown that the method is A-stable.

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \Delta t\ \dot{\mathbf{x}}_{\mathbf{k+1}} \tag{B.1}$$

First order approximation of the differential at time k+1.

$$\dot{\mathbf{x}}_{\mathbf{k+1}} \approx \dot{\mathbf{x}}_{\mathbf{k}} + \nabla_x \dot{\mathbf{x}}_{\mathbf{k}} \Delta \mathbf{x} \tag{B.2}$$

Defining:

$$\mathbf{J} = \nabla_x \dot{\mathbf{x}}_{\mathbf{k}}{}^T \tag{B.3}$$

Combining:

$$\begin{gathered} \mathbf{x_{k+1}} \approx \mathbf{x_k} + \Delta t\ \dot{\mathbf{x}}_{\mathbf{k}} + \Delta t\ \mathbf{J}\Delta\mathbf{x} \\ (\mathbf{I} - \Delta t\ \mathbf{J})\Delta\mathbf{x} \approx \Delta t\ \dot{\mathbf{x}}_{\mathbf{k}} \\ \Delta\mathbf{x} \approx (\mathbf{I} - \Delta t\ \mathbf{J})^{-1}\Delta t\ \dot{\mathbf{x}}_{\mathbf{k}} \end{gathered} \tag{B.4}$$

Stability requirements

Formulate a general test case:

$$\dot{\mathbf{x}} = \begin{bmatrix} \lambda_{11} & \dots & \dots & \dots & \lambda_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \lambda_{jj} & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \lambda_{n1} & \dots & \dots & \dots & \lambda_{nn} \end{bmatrix} \mathbf{x} = \mathbf{Jx} \tag{B.5}$$

The eigenvalue matrix of the Jacobian:

$$\mathbf{J} = \mathbf{M}^{-1}\mathbf{\Lambda}\mathbf{M} \tag{B.6}$$

The integration formula:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \Delta t\ (\mathbf{I} - \Delta t\ \mathbf{J})^{-1}\dot{\mathbf{x}}_\mathbf{k} \tag{B.7}$$

$$\mathbf{x_{k+1}} = (\mathbf{I} + \Delta t\ (\mathbf{I} - \Delta t\ \mathbf{J})^{-1}\mathbf{J})\mathbf{x_k} \tag{B.8}$$

$$\mathbf{x_{k+1}} = (\mathbf{I} + \Delta t\ \mathbf{M}^{-1}(\mathbf{I} - \Delta t\ \boldsymbol{\Lambda})^{-1}\boldsymbol{\Lambda}\mathbf{M})\mathbf{x_k} \tag{B.9}$$

We have generally:

$$Eig[\mathbf{I} + \mathbf{A}] = \mathbf{I} + Eig[\mathbf{A}] \tag{B.10}$$

The eigenvalues of $\mathbf{J}$ are negative.

$$\begin{aligned} Eig_i[\Delta t\ \mathbf{M}^{-1}(\mathbf{I} - \Delta t\ \boldsymbol{\Lambda})^{-1}\boldsymbol{\Lambda}\mathbf{M})] &= Eig_i[\Delta t\ (\mathbf{I} - \Delta t\ \boldsymbol{\Lambda})^{-1}\boldsymbol{\Lambda})] \\ &= \tfrac{\Delta t\ \Lambda_{i,i}}{1-\Delta t\ \Lambda_{i,i}}, \quad \text{for } i\epsilon\{1..n\} \end{aligned} \tag{B.11}$$

Under the assumptions presented, we have:

$$\lim_{\Delta t\ \to\infty}\left(\tfrac{\Delta t\ \Lambda_{ii}}{1-\Delta t\ \Lambda_{ii}}\right) = -1 \tag{B.12}$$

The real part of the eigenvalue lies within the following bounds:

$$-1 < Re\left(\tfrac{\Delta t\ \Lambda_{i,i}}{1-\Delta t\ \Lambda_{i,i}}\right) \leq 0, \quad \text{for } i\epsilon\{1..n\}, \quad \Delta t \geq 0 \tag{B.13}$$

Stability condition:

$$\mid Re(Eig_i[(\mathbf{I} + \Delta t\ \mathbf{M}^{-1}(\mathbf{I} - \Delta t\ \boldsymbol{\Lambda})^{-1}\boldsymbol{\Lambda}\mathbf{M})]) \mid < 1 \tag{B.14}$$

The left-hand side of Equation B.14 will lie between zero and one. The linear implicit integration is A-stable.

B.2 Fully implicit Euler integration

The implicit Euler scheme is given in Equation B.15.

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \Delta t\ \dot{\mathbf{x}}_\mathbf{k+1}, \qquad t_{k+1} = t_k + \Delta t \tag{B.15}$$

To solve this system an iterative scheme has to be used. An NR algorithm with line search is chosen. The function to be solved has the following form:

$$\mathbf{F}(\mathbf{x^i_{k+1}}) = \mathbf{x^i_{k+1}} - \mathbf{x_k} - \Delta t\ \dot{\mathbf{x}}^\mathbf{i}_\mathbf{k+1}, \qquad \dot{\mathbf{x}}^\mathbf{i}_\mathbf{k+1} = \dot{\mathbf{x}}_\mathbf{k+1}(\mathbf{x^i_{k+1}}) \tag{B.16}$$

Need to solve $\mathbf{F}(\mathbf{x^i_{k+1}}) = 0$. The iterative scheme then becomes.

$$\mathbf{p^i} = -\nabla\mathbf{F}^{-1}\mathbf{F} = -(\mathbf{I} - \Delta t\mathbf{J})^{-1}\mathbf{F} \tag{B.17}$$

$$\mathbf{x^{i+1}_{k+1}} = \mathbf{x^i_{k+1}} + \alpha\mathbf{p^i}, \qquad \alpha \in \langle 0, 1] \tag{B.18}$$

A line search is executed to determine a suitable value for the α parameter, giving a sufficient decrease in some function norm.

B.3 Matrix algebra

To solve the linear equation sets that emerge during integration, and otherwise, a matrix template was written and fronted to Lapack. The Fortran 77 version 3.0 of Lapack, Anderson et al. (2000), is used.

In the same manner a sparse matrix template, fronting the sparse BLAS function library shipped with SparseLib++ v1.5d, (Dongarra, Lumsdaine, and Pozo (1996)). To solve sparse systems the function templates of IML++ v1.2a, Dongarra et al. (1995), are used. The preferred sparse solver method was GMRES, Saad (2003), with an ILU(0) reconditioner. The IML++ template was modified to work with the sparse matrix template.

B.4 General relations

Definition of a q-quadratic convergence given by Nocedal and Wright (1999).

$$\frac{\left\| x_{k+1} - x^* \right\|}{\left\| x_k - x^* \right\|^2} \leq M, \quad \text{for all k sufficiently large} \tag{B.19}$$

Here, M is a bounded positive constant, x^* is the solution vector of x, x_k is the k-th NR iteration result for x. The $\left\| \ldots \right\|$ is an arbitrary vector norm.

The partial differential operator nabla is defined as follows:

$$\nabla_{\mathbf{x}} = \left[\begin{array}{ccc} \frac{\partial}{\partial x_1} & \cdots & \frac{\partial}{\partial x_n} \end{array} \right]^T \tag{B.20}$$

The Jacobian, $\mathbf{J}$, of a general function vector, $\mathbf{F} = \mathbf{F}(\mathbf{x})$ is defined by Equation B.21.

$$\mathbf{J} = \mathbf{F}'(\mathbf{x}) = \nabla_x \mathbf{F}(\mathbf{x})^T \tag{B.21}$$

The difference operator, Δ, is defined in Equation B.22.

$$\Delta_{a,b} f = f_a - f_b \tag{B.22}$$

The backward difference polynomial interpolation operator, Γ, is defined through Equation B.23. The operator is applied on a function $f = f(x)$. The subscript index, n, is refers to x_n, that is; $f_n = f(x_n)$. For the variable x, the following holds: $\cdots < x_{n-2} < x_{n-1} < x_n < x_{n+1} < \ldots$.

$$\Gamma^0 f_n = f_n, \qquad \Gamma^{j+1} f_n = \Gamma^j f_n - \Gamma^j f_{n-1} \tag{B.23}$$

Definition of the space time, given by Fogler (1999).

$$\tau = \frac{V\rho_{inlet}}{W_{inlet}} \tag{B.24}$$

Definition of the compression factor, given by Atkins (1994).

$$Z = \frac{P}{\rho RT} \tag{B.25}$$

Appendix C

Flowsheet for the LNG Plant

Figure C.1 shows the MFCP process split up into six sub-processes interacting with each other. The six sub-processes are given in Figures C.3 through C.8. The sub-processes are identified through the figure text, and the numbering on Figure C.1, (SP 1 = Sub Process 1, etc.).

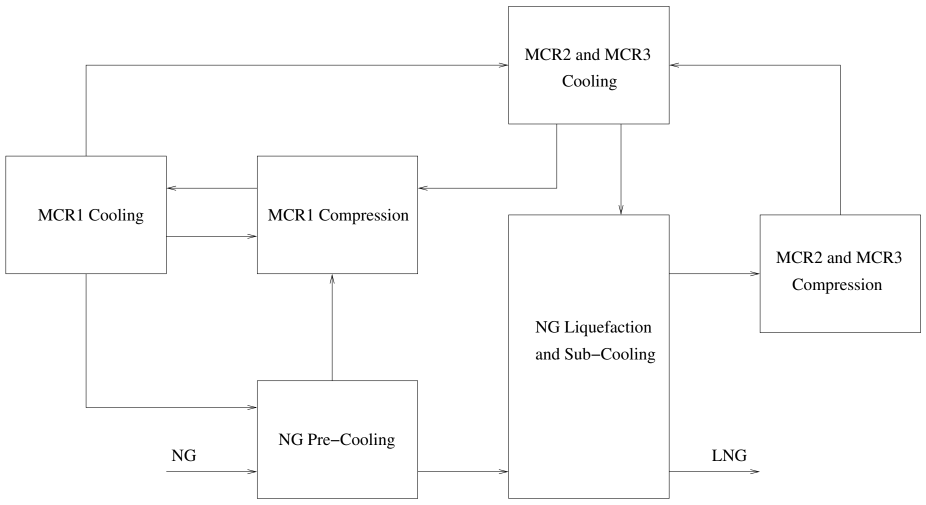


Figure C.1: Overview of the sub systems

Table C.2 show a structure map of the unit names. The positions 1 through 5 are explained in Table C.1.

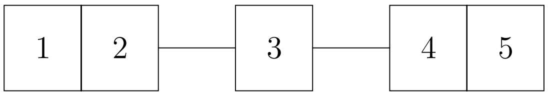


Figure C.2: N

Table C.1 show how the flowsheet units are named.

Table C.1: Name

Position	Description
1	Name prefix
2	System number
3	Unit specific code
4	Unit number
5	Name suffix

The naming is further defined in Tables C.2 through C.4.

Table C.2: Name prefix

Name	Description
N	Node
S	Source
V	Valve

Table C.3: Unit specific codes

Name	Description
CT	Liquid expander
HA	Simplified heat exchanger, constant temperature output.
HG	Heat exchanger
HX	Heat exchanger
KA	Compressor
VD	Tank - flash or node

Table C.4: Name suffix

Name	Description
SF	Static Flow
SP	Static Pressure
A or B	Two units with the same name
-I or -II	Two units with the same name

Example: **S25-138SP** : Source in system 25, unit number 138 and the node supplies a static pressure to the equation set. **25-HG-104** : System 25 heat exchanger, unit number 104.

The only exception from the naming rules is *node* **N25-LNG-SP**, using a "LNG-SP" instead of a number and a followed by "SP".

Table C.5: Controller names

Name	Description
FIC	Flow Identification and Control
LIC	Level Identification and Control
PIC	Pressure Identification and Control
TIC	Temperature Identification and Control

Table C.5 shows the meaning of the controller names.

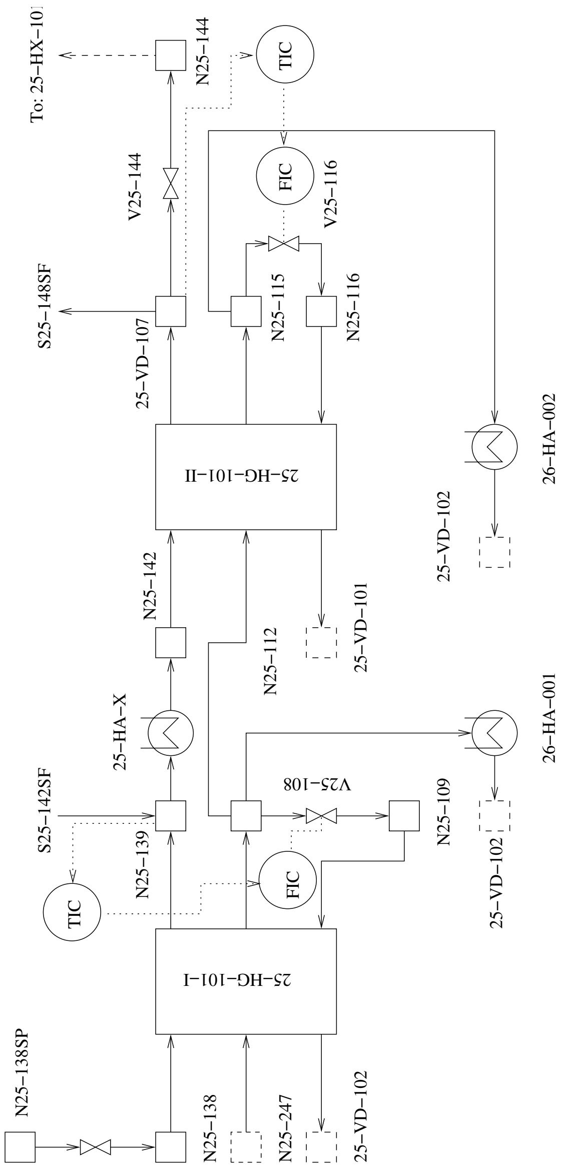


Figure C.3: NG Pre-Cooling - SP 1

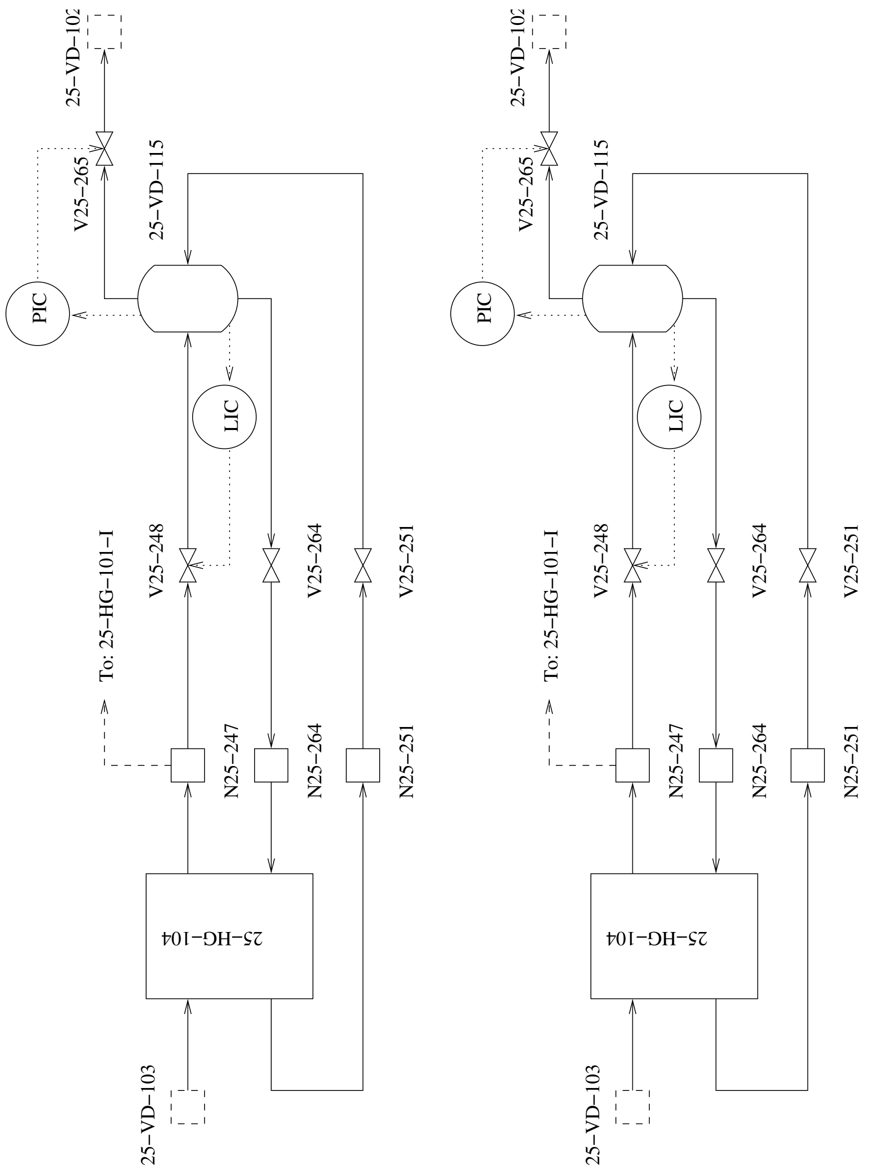


Figure C.4: MCR1 Cooling - SP 2

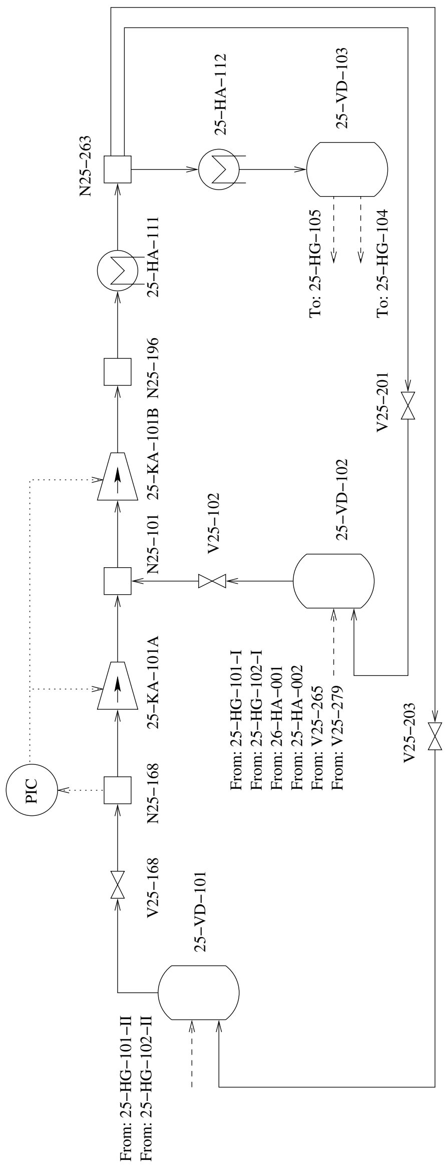


Figure C.5: MCR1 Compression - SP 3

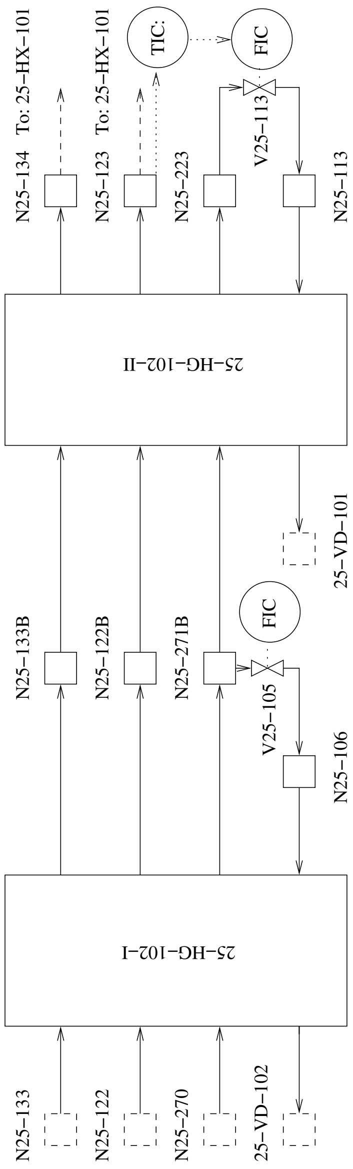


Figure C.6: MCR2 and MCR3 Cooling - SP 4

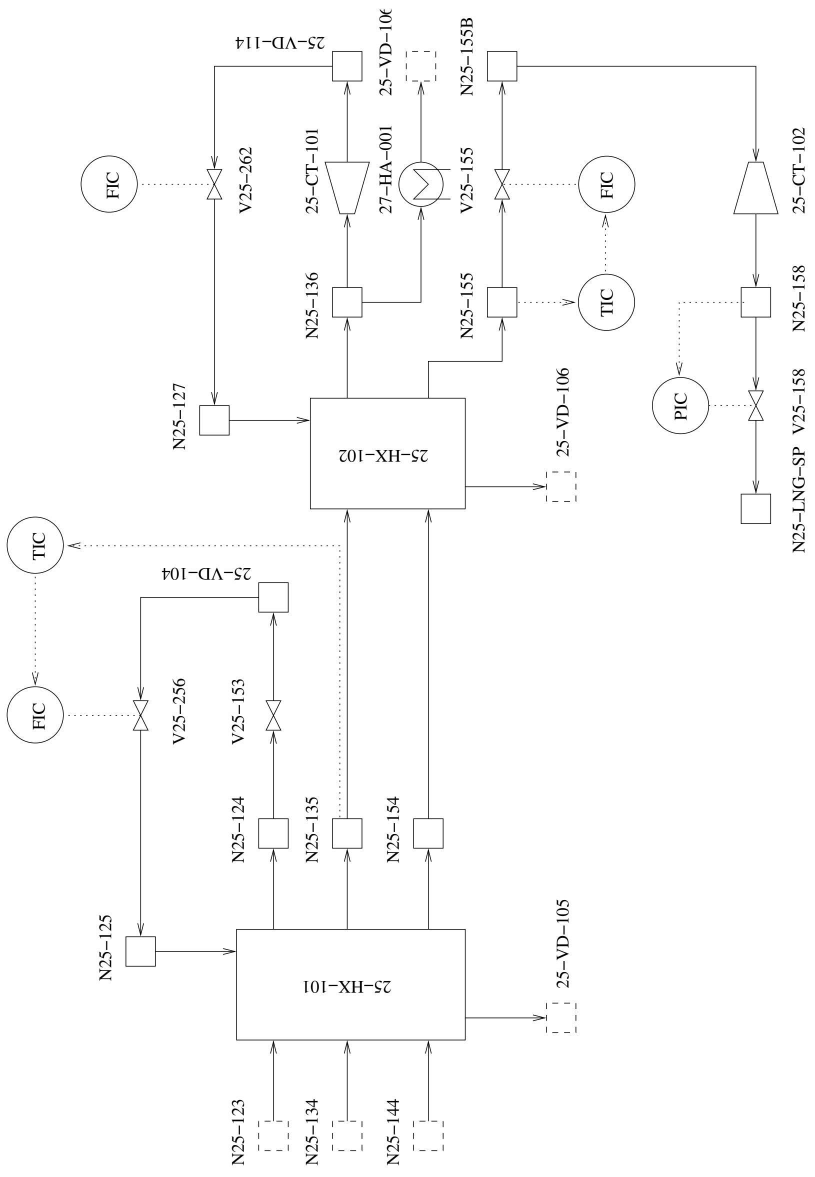


Figure C.7: NG Liquefaction and Sub-Cooling - SP 5

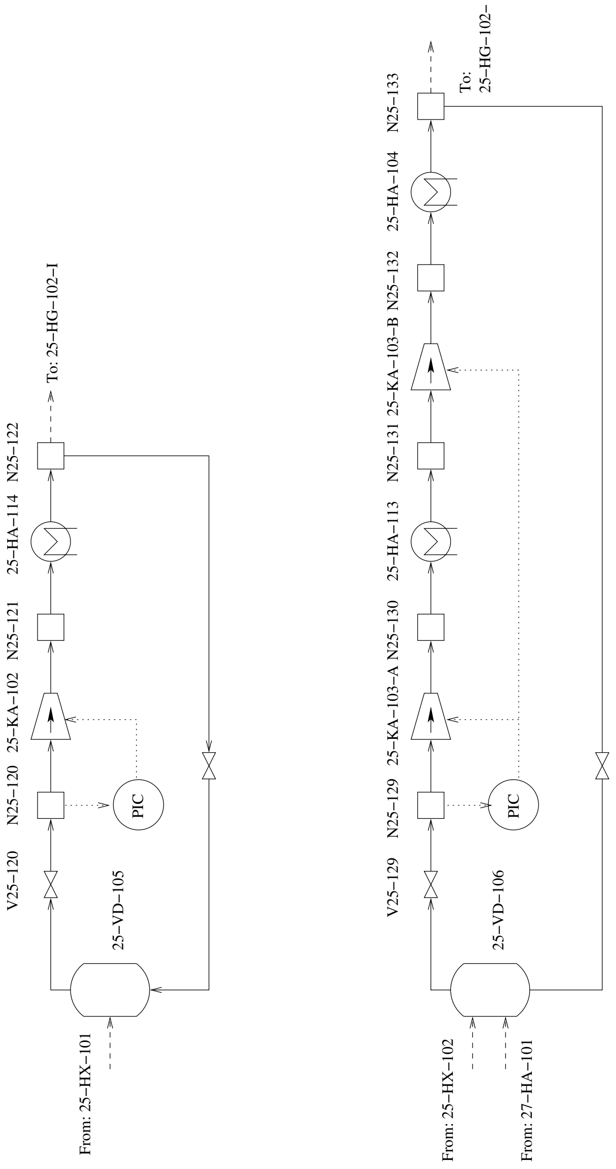


Figure C.8: MCR2 and MCR3 Compression - SP 6