Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet

Magnus Nilsen Myhre

Investigation of IDAES and optimization for a SMR process

Masteroppgave i Produktutvikling og produksjon 2-årig - Industriell produksjon

Januar 2022

NTNU
Kunnskap for en bedre verden

Magnus Nilsen Myhre

Investigation of IDAES and optimization for a SMR process

Masteroppgave i Produktutvikling og produksjon 2-årig - Industriell produksjon
Januar 2022

Norges teknisk-naturvitenskapelige universitet

NTNU
Kunnskap for en bedre verden

Investigation of IDAES and optimization for the PRICO process

Candidate number: 10006/ Name: Magnus Nilsen Myhre

Abstract

Natural gas is a resource that is getting more convenient to use as an energy resource with the ever-changing world as humans realize that the world has to move towards a more friendly attitude to the climate. Today the oil and gas industry stands tall in the competition of energy generation together with coal, and for the most part, oil has been the alpha since the first big discovery of the resource in 1969 at Ekofisk and has been one of the most severe oilfields to be extracted to this day. Natural gas has been discovered to have less C02 emissions and is therefore growing more popular to use in replacement for oil and coal in different areas. This thesis takes a look at the liquefaction of natural gas, more specifically a single mixed refrigerant process. The relatively new process software tool IDAES have been applied together with ASPEN HYSYS to model and simulate the SMR process. Lastly, an optimization of the process by the genetic algorithm *Particle swarm* has been tested to find optimum control variables to reduce energy consumption in the process.

Sammendrag

Naturgass er en ressurs som stadig blir mer beleilig å bruke som ressurs for energi med en verden som bestandig forandrer seg og med menneskeheten som innser at verden må gå mot et mer vennlig standpunkt i forhold til klimaet. I dag står olje & gass industrien høyt i konkurransen om generering av energi sammen med kull, og for det meste, har olje gass industrien vært alfaen siden den første store oppdagelsen i 1969 på Ekofisk og har vært et av de mest ekstraktede oljefeltene så langt. Naturgass har blitt oppdaget å ha lavere C02 utslipp, og har derfor blitt mer populært til å erstatte olje og kull som energikilder i forskjellige områder. Denne tesen tar en kikk på likvifisering av naturgass, mer spesifikt en singel mikset kjølemedie prosess. Den relativt nye prosess programvaren IDAES har blitt implementert sammen med ASPEN HYSYS til å modellere og simulere SMR prosessen. Til slutt har en optimalisering av prosessen ved bruk av en genetisk algoritme, *Particle swarm*, blitt testet for å finne optimale kontroll variabler for å redusere energi forbruk i prosessen.

Acknowledgement

I would like to thank my supervisor Bjørn Austbø for always being available, for being very cooperative and helpful, and for contributing of many good ideas and material for this thesis. I would also like to thank the developers of IDAES for troubleshooting in this project through their discussion forum found on Github. Lastly I would like to thank my family and friends for their support throughout my time at NTNU and for always keeping my motivation at a high level.

Contents

Abstract . . . . . **i**
Sammendrag . . . . . **ii**
Acknowledgement . . . . . **iii**
Contents . . . . . **iv**
Figures . . . . . **vi**
1 Introduction **1**
1.1 Research question and objective 2
1.2 Scope 2
1.3 Outline 2
2 Theory **4**
2.1 Natural gas 4
2.2 SMR process 5
2.3 IDAES 5
2.4 HYSYS 7
2.5 Cubic equations of state 7
2.6 Optimization 8
2.6.1 Deterministic optimization 9
2.6.2 Stochastic optimization 9
2.6.3 Particle swarm optimization 10
3 Methodology **12**
3.1 Model setup 12
3.2 Property package 15
3.2.1 Methods for calculating parameters 15
3.2.2 Units of measure 16
3.3 Read the docs 16
3.4 IDAES 17
3.4.1 Installation 17
3.4.2 General workflow 17
3.5 Python 21
3.6 Matlab & HYSYS 22
3.6.1 Optimization cases 24
3.7 Particle swarm optimization algorithm 27
4 Result **29**
4.1 PSO with S&T *Simple end point* modification 29

4.2 PSO with S&T *Simple weighted* modification 32
4.3 PSO with LNG heat exchanger 33
4.4 IDAES model 35
4.4.1 Comparison with HYSYS 38
4.4.2 Challenges along the way 39
5 Discussion and conclusion **42**
Bibliography **45**
A Unit model reports **48**
B Overflow error **52**
C Evaluation and scaling error **56**
D HYSYS coefficients **64**
E Property package configuration **65**
F IDAES python script **77**
G Matlab optimization script **84**

Figures

2.1 SMR process 5
2.2 IDAES framework [4] 6
2.3 Particles spread about space moving towards better solutions. Retrieved from [15] 11

3.1 Equivalent configuration of two shell and tube heat exchangers to represent a MHEX. 13
3.2 Multistream heat exchanger. 13
3.3 Individual Temperature-enthalpy curves for each stream in a MHEX 14
3.4 Flow chart model with modification 14
3.5 Expression for purity of benzene in an outlet of a flash unit model 19
3.6 Output levels for the IDAES logging tool 20
3.7 Output report for a heat exchanger 21
3.8 Options for the PSO 22
3.9 Decision variables, constraints and objective function 23
3.10 Spreadsheet called "Efficiency", displaying most variables . . 23
3.11 SMR process 24
3.12 Simple end point model for min T. approach, calulated at the ends 25
3.13 Simple weighted model for min T. approach, calulated at each interval 26
3.14 SMR process with a multistream heat exchanger 26

4.1 Modified IDAES model 36
4.2 Streamtable 36
4.3 Report for HE101 37
4.4 Report for HE102 38
4.5 Model in HYSYS 39
4.6 Stream table HYSYS 39

A.1 Compressor report 48
A.2 Heater/Cooler report 49

A.3 Valve101 report 49
A.4 Valve101 report 50
A.5 Splitter report 50
A.6 Valve101 report 50
A.7 HE101 report 51
A.8 HE102 report 51

D.1 Ideal gas enthalpy coefficients in HYSYS 64

Chapter 1

Introduction

As hydrocarbons generally are a great source of energy, the need for both small and large-scale processing systems are necessary to extract the most of the energy out of them in an efficient way. To make it as effective as possible one could experiment with simulation and modeling tools to find the optimal working conditions of the processing systems. One major player in the field is ASPEN HYSYS, which of from now on will be referred to as HYSYS, is probably the number one software in the market today. HYSYS needs licensing and could be costly for the average person or small businesses. The relatively new open software IDAES, short for *Institute for the Design of Advanced Energy Systems*, has come to fight for a place on the board, and the key difference between the two is that HYSYS offers an exceptional visualization and inbuilt unit models with a property package base which is extensive, while IDAES is almost completely built upon coding/programming, which in turn perhaps can give more freedom and possibility for customization. HYSYS can also be viewed as a black box system, meaning that the information coming in is transformed into some information coming out without really getting to know what happens within. HYSYS consists of a sequential modular approach where each stream and unit model is solved sequentially and uses mathematical models to predict the performance. IDAES is an equation-oriented software where a flowsheet is treated as a set of equations to be solved simultaneously and is purely mathematical. IDAES is still under development and has yet a far way to go in terms of what HYSYS can do, but it certainly has great potential to compete later.

Background

In advance of this thesis, the author had a project where IDAES was reviewed and it was found that some problems were met due to initialization and simulation of the flowsheet which was modeled. A literature study on how the Poly Refrigerant Integrated Cycle Operations (PRICO) process could be optimized was also performed and it was found that pressuriz-

ing the natural gas and changing the composition of the mixed refrigerant was important driving factors to reduce the power consumption of the compressor.
In this report, a more thoroughly try to get the model to work in IDAES is tried and implementation of a genetic algorithm called *partcle swarm optimization*, or PSO, is applied together with a hybrid function called *pattern search* to find the optimal values for the control variables of a single mixed refrigerant process, referred to as SMR from now on.

1.1 Research question and objective

The objective of the thesis will be to find out if IDAES can be a sensible alternative as a process simulation and modeling software compared to the likes of supreme software such as HYSYS by modeling an SMR process and simulating it. In particular to investigate its possibilities and limitations. The other objective of this thesis will be to optimize the SMR process with a set of given decision variables in HYSYS and make sense of the results, especially concerning the composition of the mixed refrigerant.

1.2 Scope

IDAES offers a great portion in terms of operations, to mention a few is modeling, simulation, optimization, surrogate modeling, and dynamics & control. Also, the possibility to customize property packages and unit models is an opportunity. In this thesis, the usage of IDAES is limited to modeling and simulation of a simple mixed refrigerant process to liquefy natural gas as well as implementing a configuration dictionary including the components of the natural gas and refrigerant which is utilized with an existing cubic equation of state property package in the IDAES property package library. For the optimization part of the project, the optimization is limited to the use of a particle swarm optimization algorithm with a hybrid pattern-search function for a similar single mixed refrigerant process with the use of Matlab and HYSYS.

1.3 Outline

The theory chapter will explain a bit of what natural gas is in 2.1 and how to process it to a liquefied fluid in 2.2 for an SMR process. Section 2.3 and 2.4 describe shortly how IDAES and HYSYS work and what capabilities they provide. Section 2.5 displays the theory of calculating the equations of state of the fluid utilized in the property package implemented in the IDAES and HYSYS simulation. Lastly in the chapter, section 2.6 explains what optimization is and some of the types of optimization there are, such as deterministic

and stochastic optimization. The particle swarm optimization algorithm is part of the latter. The methodology chapter explains how the model was set up in IDAES and HYSYS with the property package applied. A general workflow is included in section 3.4 where the procedure of creating a flowsheet and how to solve it is carried out in IDAES. Finally, a brief oversight of how Matlab and HYSYS have been used together with the PSO algorithm concludes this chapter. Chapter 4 displays the result gathered from the simulation in IDAES and HYSYS as well as the results from the optimization part. In chapter 5 the results are discussed and whether the objective of the thesis has been accomplished or not. Then a brief conclusion is made at the very end together with a suggestion for future work.

Chapter 2

Theory

2.1 Natural gas

Natural gas is gas that contains hydrocarbons, mostly methane and ethane, and is a result of high pressure, temperature, and fossils packed together over a long time below the ground. The gas is often found dissolved in the oil from reservoirs, but can also lay like a cap above the oil, which often will also contain some propane and butane.
In the earlier days, about the 19th century, it was mainly the crude oil and coal that were used for the industrial improvements, the reason being that the natural gas was hard to transport in large quantities. However, pipelines were built in the late 19th century, but these were restricted to only reach for about 160km because the technology at the time was not too great. It was in the 20th century, especially when world war 2 broke out, that the pipeline technology became fierce. Russia has the largest pipeline, reaching a distance of 5470km, and is shipping natural gas to big parts of Europe for energy consumption.
Natural gas can be used to heat homes, for cooking, for generation of electricity and as fuel for some vehicles [1].
According to the Center for climate and energy solutions, it is estimated that the carbon dioxide emission is half that of coal and one-third of oil and it is the largest source of electric power generation in the U.S. The transportation sector is dominated by oil, and the C02 emission from it reaches almost 28 percent of the total emission in the U.S. Natural gas is mostly used in vehicles such as buses and trucks using compressed natural gas (CNG) or LNG and if these had been used more widely it would have a big impact on the emission from this sector. [2].

2.2 SMR process

The SMR process is implemented in process systems to liquefy natural gas. Often for small-scale LNG plants, such as offshore export terminals. There are several methods to produce LNG, the SMR process is one of them. It was first developed by Black & Veatch Company. Production of LNG is very energy extensive, and the potential to increase the efficiency of the production is great. The mixture of the refrigerant is nitrogen, methane, ethane, and butane. Process unit components for the SMR process include a heat exchanger where heat is transferred between the cold SMR and the hot natural gas, a compressor where the refrigerant is compressed isentropically, and a cooler where the refrigerant is condensed, and expansion valves where the pressure is relieved and phase change occurs again. The behavior of the mixture in these components is dependent on the composition and its chemical constituency [3]. A small model of the SMR process system is shown in Figure 2.1

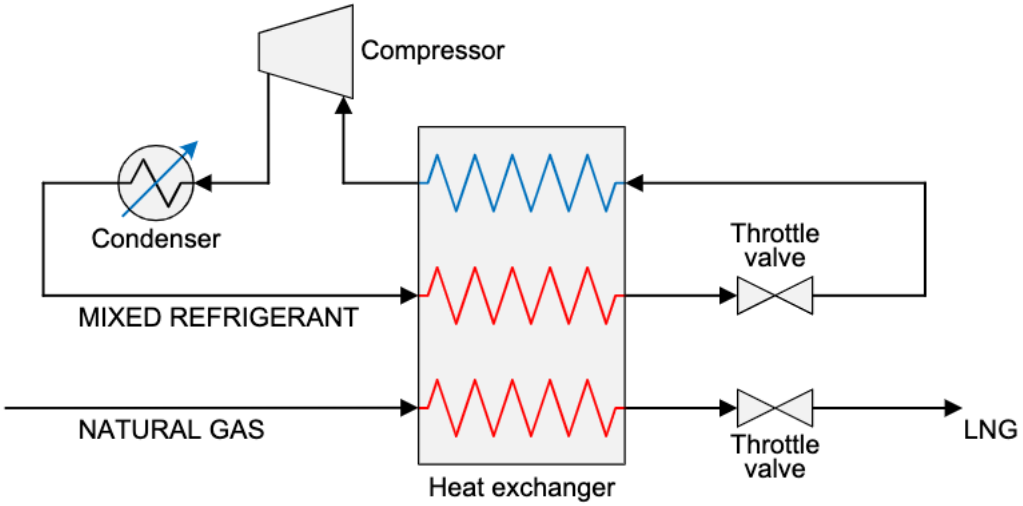


Figure 2.1: SMR process

2.3 IDAES

IDAES is short for Institute for the Design of Advanced Energy Systems and is an open-source modeling and simulation software tool, and it was developed by a team of scientists and engineers. Their vision is to design a new world energy system. The integrated IDAES platform contains advanced modeling and optimization capabilities and it uses advanced algorithms to solve complex process systems. The IDAES framework is built upon 3 main branches of computation, these are shown in figure 2.2.

The top row describes different applications that can be solved in the integrated platform of IDAES. These include surrogate modeling, conceptual design, process dynamics & control, uncertainty quantification, plant design & process optimization, materials optimization and enterprise optimization [4]. The middle row shows the principle of how the modeling framework works, mainly that one will first choose whether the model should be dynamic or steady-state, that one connects each unit of the process elements with arcs and that for each element there is a control volume. More of this will be explained more thoroughly in the next section. The last row shows the optimization solvers that are implemented in the IDAES framework, such as PYOMO, which is a python based optimization language, and ipopt which is mainly used for solving the model after it is initialized.

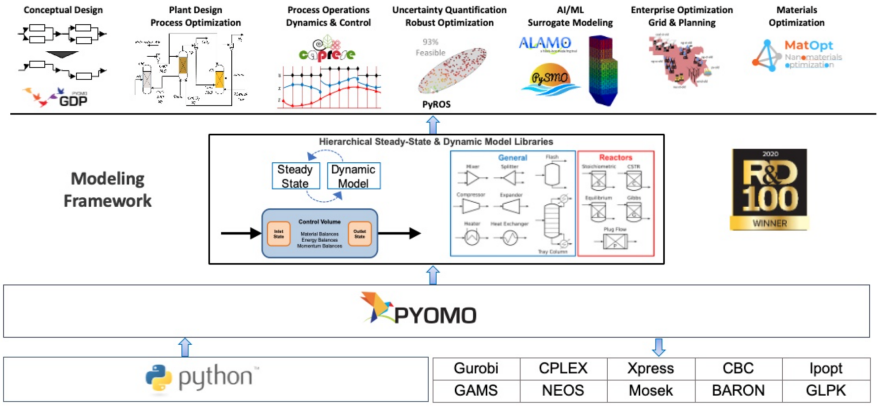


Figure 2.2: IDAES framework [4]

Optimization

An initial solution is hopefully found, and the user may now try to optimize a problem of interest. This usually includes unfixing some of the variables specified earlier, to give the solver some degrees of freedom to work with. Further one would add bounds and constraints to narrow down the solution area and finally call for the solver and check for the termination conditions as described in the last section.

Visualization of results

To conclude the typical workflow, the user may want to visualize their results. IDAES provides different visualization tools to create plots, reports, and flowsheet visualization. A flowsheet visualization can be displayed with the following code snippet

```
m.fs.visualize(m)
```

2.4 HYSYS

Aspen HYSYS is the leading Simulation and modeling software in the process industry developed by AspenTech. Its capability includes modeling small and large process systems including unit operations and can do calculations of mass balances, energy balances, vapor-liquid equilibrium, heat and mass transfer, kinetics, and more. It can do both steady and dynamic simulation and is generally used for optimization, process design and performance modeling [5][6].

2.5 Cubic equations of state

The cubic equations of state are modifications of Van der Waal´s equation of state, which correlates the pressure of a gas with temperature and density in terms of the molar volume. Wan der Waal´s equation of state is stated in Eq 2.1

$$\left(P + a\frac{1}{V_m^2}\right)(V_m - b) = RT \tag{2.1}$$

The constants a and b represents the attraction and repulsion parameter of the effective molecular volume respectively and is given by Eq 2.2 and Eq 2.5 in terms of the critical temperature and pressure with the ideal gas constant.

$$a = \frac{27(RT_c)^2}{64p_c} \tag{2.2}$$

$$b = \frac{RT_c}{8p_c} \tag{2.3}$$

Wan der Waal´s equation was formulated in 1873 and was considered to be much better than the ideal gas law, in fact, it was called the "improved ideal gas law" as it could predict the formation of a liquid phase, but based on the experimental data, is limited and does not work with higher pressures and temperatures [7].

Peng-Robinson

In 1976 Peng and Robinson developed a new two-constant equation of state where the empirical attractive constant term is modified. They proposed the following equation as of Eq 2.4. This equation can be used to predict the vapor pressure of pure substances and the ratio equilibrium of mixtures. [8]

$$
\begin{aligned}
p &= \frac{R\,T}{V_m - b} - \frac{a\,\alpha}{V_m^2 + 2bV_m - b^2} \\
a &\approx 0.45724 \frac{R^2\,T_c^2}{p_c} \\
b &\approx 0.07780 \frac{R\,T_c}{p_c} \\
\alpha &= \left(1 + \kappa\left(1 - T_r^{\frac{1}{2}}\right)\right)^2 \\
\kappa &\approx 0.37464 + 1.54226\,\omega - 0.26992\,\omega^2 \\
T_r &= \frac{T}{T_c}
\end{aligned}
\tag{2.4}
$$

Or in polynomial form

$$
\begin{aligned}
Z^3 - (1-B)Z^2 + (A - 2B - 3B^2)Z - (AB - B^2 - B^3) &= 0 \\
A &= \frac{a\,\alpha\,p}{R^2T^2} \\
B &= \frac{bp}{RT} \\
Z &= \frac{pV}{nRT}
\end{aligned}
\tag{2.5}
$$

where ω is the acentric factor of the species.

2.6 Optimization

The mission to make processes and functions of objects more effective is always something that will never stop endure. A small tweak on a geometrical surface, such as the surface of a streamlined body will have an impact on the aerodynamic properties. A change in how many products from a factory that is shipped to each warehouse may have an impact on transportation costs when delivered to the customer from the warehouse. To decide whether to buy a new automated machine, or to keep the two workers that are doing the same job, could be hard to know keeping in mind that the machine is more effective, but can potentially be a liability if it breaks down. All these examples are examples of potential optimization problems. It turns out that optimization is a great tool in engineering problems and can reduce costs and increase effectivity and functionality of different problems. The technical definition of optimization is according to the oxford English dictionary "*The action of making the best or most effective use of a situation or resource*" [9] and a "*A mathematical technique for finding a maximum or minimum*

value of a function of several variables subject to a set of constraints, as linear programming or systems analysis", according to the online dictionary, WordReference [10].
In an optimization problem, it is important to identify what is going to be optimized, which is going to be the objective function, and what variables which are to be used, which are called the decision variables. The complexity of an optimization problem grows with the number of variables, which may have an impact on the time it takes to solve the problem. Another important aspect of an optimization problem is the range with of the decision variables are allowed to work with or the constraints which are to be included if any. To sum up, an optimization problem would normally look like this as of Eq 2.6, where g_i and h_j represent the inequality and equality constraints respectively, of the problem.

$$\begin{aligned} min/max \;\; & f(x) \\ & subject \;\; to \\ g_i(x) \leq 0 \;\; & i = 1, ..., m \\ h_j(x) = 0 \;\; & j = 1, ..., p \end{aligned} \tag{2.6}$$

Many different optimization algorithms possess distinct methods and unique abilities to solve problems. Some of those are presented in the next section.

2.6.1 Deterministic optimization

Deterministic optimization, also called mathematical programming is used extensively in mathematics. It relies heavily on linear algebra as they are frequently used to compute gradients and Hessians, which is the slope and a square matrix with second partial derivatives, respectably [11]. There are upsides and downsides to deterministic optimization. One positive is that the convergence to a solution is highly faster than that of a stochastic algorithm, which is based on randomness. It needs fewer function evaluations to reach a solution. As deterministic optimization is a rigorous mathematical algorithm not involving any stochastic elements in it, the result is unique and unambiguous. The algorithm will look for stationary points and the optimal solution may be either a local optimum or a global optimum. A downside of deterministic optimization is that it requires a complete defined model and rigorous software which is very hard to build as all dependencies need to be formulated rigorously as well. [11]

2.6.2 Stochastic optimization

Randomness may be the most precise definition of stochastic optimization. The search procedure will contain randomness and a solution may be found rather quickly or can take a longer time based on how close the random configurations are to a feasible solution. There are many branches in stochastic

optimization, some of these are simulated annealing, particle swarm optimization, game theory-based optimization, evolutionary algorithms, and genetic algorithms [11]. The main source of inspiration for these methods is nature itself. By observing nature concerning biology, physics, and geology, a formulation for these methods can be brought to life and implemented into the algorithms in a simplified manner. The majority of these algorithms are population-based where a population can be taken as a set of samples that evolves to convergence. Implementation of rules for the evolution is set and will include randomness and ruled by the inspiration from the implemented natural observations. Collective behavior of the population may be based on animal or insect behavior and tries to mimic this. Terminology for this collective behavior is called swarm intelligence [11]. An advantage of stochastic optimization is that it will continue to search beyond a local optimum as the algorithm includes randomness. A disadvantage of stochastic optimization is that it will continue to search and won't stop if there are no limitations set, so it may be hard to know when a solution is good enough within a reasonable time [11].

2.6.3 Particle swarm optimization

The particle swarm optimization algorithm was created by Kennedy and Eberhart in 1995. They describe it as a simple algorithm for many optimization problems and a unique concept of the method is that there are many solutions through hyperspace, and acceleration happens towards better solutions. They further describe that the success of the particle swarm optimization lies in the particle's ability to hurtle past their target.
The stochastic aspect of the algorithm comes from what is called "craziness", which has the effect of making the particles spread beyond a clustered solution that tends to attract most of the particles. In this way, it will spread beyond these points and look for other solutions. Particle swarm optimization can be related to artificial life, and most notably to bird flocking, fish schooling, and swarm theory. The behavior of the algorithm can be compared with evolutionary and genetic programming [12]. Figure 2.3 shows the particles at an initial position moving towards solutions.

The algorithm is partly inspired by the work of Heppner and Grenander [13], which researched how bird flocking works, especially the synchronization of the birds and finding the leader of such flocks. Experimental research on finding those leaders have not yet found any success. They proposed a bird flocking simulation in which the synchronization of movement was set by some rules by a stochastic algorithm that simulated realistic flock behavior. Similarly in the paper of Reynolds [14], such a simulation was also made where the path of the individual bird was calculated. The birds were taken as particles and the simulated motion of the flock was created by a

distributed model of behavior, similar to that of a natural flock.

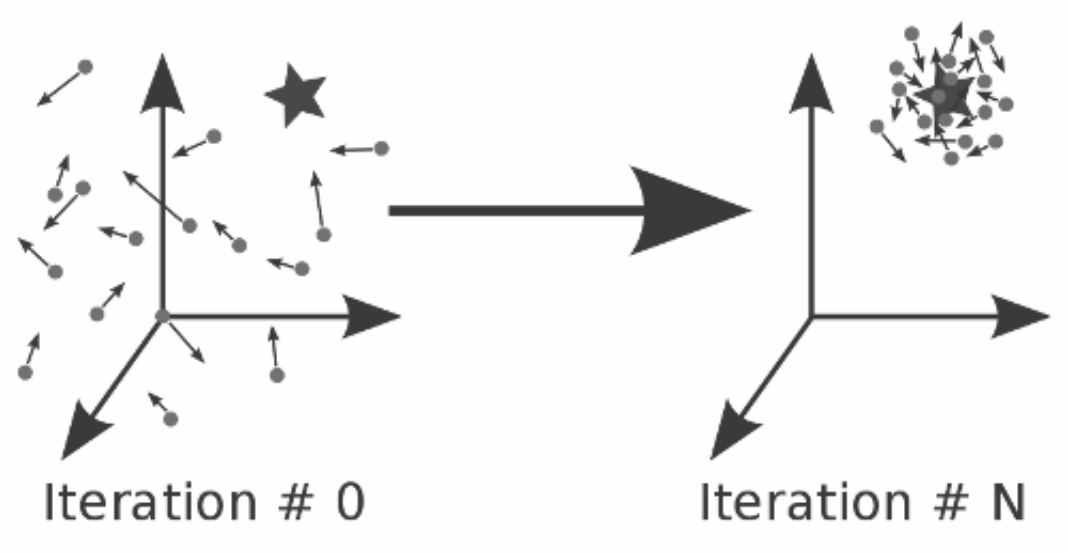


Figure 2.3: Particles spread about space moving towards better solutions. Retrieved from [15]

Chapter 3

Methodology

This chapter will describe the methodology for this thesis regarding model setup in IDAES and HYSYS and the optimization for the SMR process using the particle swarm optimization algorithm.

3.1 Model setup

The SMR process was to be modeled, and a simple process system was achieved by the author's supervisor. The model is based on Figure 2.1 but had to be modified for it to work in IDAES. It had to be modified since the IDAES unit model library did not include a multistream heat exchanger (MHEX) as what HYSYS did, where it is named as an LNG heat exchanger. An MHEX consist of 2 or more hot/cold streams and is typical in the form of a plate, fin, or spiral wound configuration, however, there are also multistream shell and tube heat exchangers. IDAES do have shell and tube heat exchangers with two streams in their unit model library and has been used in some of the tutorials they have provided.
With this in mind, a solution could be to replace the MHEX with a configuration of two-stream shell and tube heat exchangers to represent a similar case as with the MHEX. It should be noted that it does not replicate the MHEX exactly, but is intended as a possible case scenario. In an article of Rao et al. [16], it was proposed an operational optimization of processes with MHEX´s where it was developed a predictive model for the heat exchangers followed by an optimization procedure. It was said that MHEXs could be compared with black-box systems, as the MHEXs usually have complex geometries and unique designs. A non-linear programming model was implemented to synthesize a similar network of two-stream heat exchangers to best represent an MHEX. Results from the article are shown in Fig 3.1 where an equivalent configuration of two shell and tube heat exchangers represent a three-stream MHEX shown in Fig 3.2. The idea is to split the refrigerant into the two heat exchangers and the molar flow rate can be adjusted to each heat exchanger to possibly get a low minimum temperature

approach for both the exchangers.

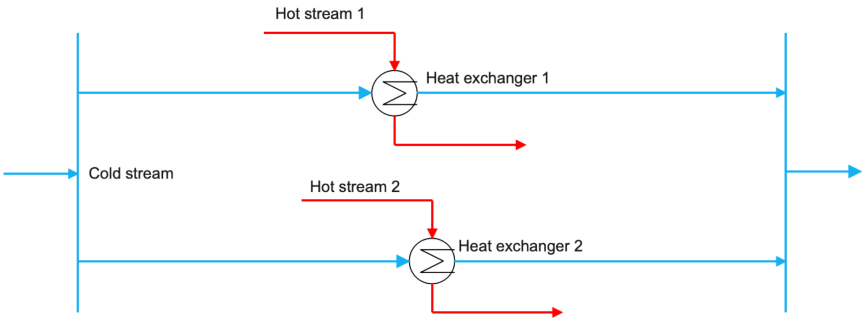


Figure 3.1: Equivalent configuration of two shell and tube heat exchangers to represent a MHEX.

Figure 3.2: Multistream heat exchanger.

This system of two shell and tube heat exchangers does not represent all of the heat transfer parameters of an MHEX because the composite curves would look different when there are only two streams compared to several streams. For example, the individual temperature-enthalpy curves for each stream in an MHEX could look like as displayed in Fig 3.3 where the black stipulated curve represents the effectively hot composite curve.
It is not possible to implement an effective composite curve for a system of two shell and tube heat exchangers as the flow is separated and not working within the same unit. However, for simplification, assuming that it does not implicate errors to a high degree, this configuration has been established in the model of the SMR process to be built in IDAES for the intention of learning purposes and to have a case to work with.
The final modification of the model will look like as of Figure 3.4, and this is what has been simulated in HYSYS and IDAES except that there is a breaking point in the flow streams into the Compressor and one of the heat exchangers in the IDAES model, the reason being that it did not work well to simulate when there was a loop involved. This will be explained in more detail in section 4.4

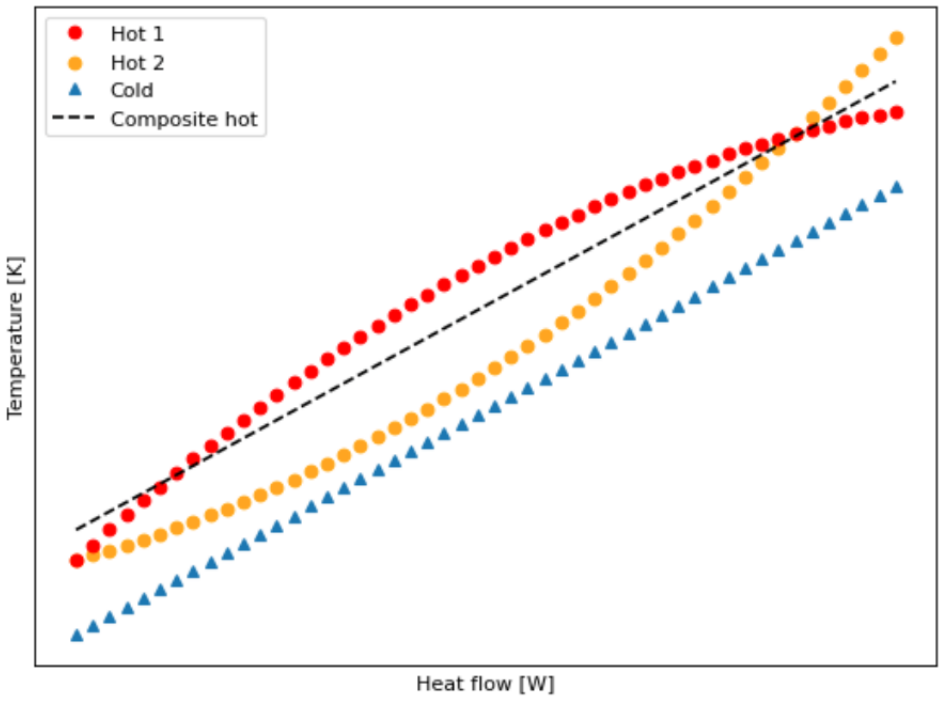


Figure 3.3: Individual Temperature-enthalpy curves for each stream in a MHEX

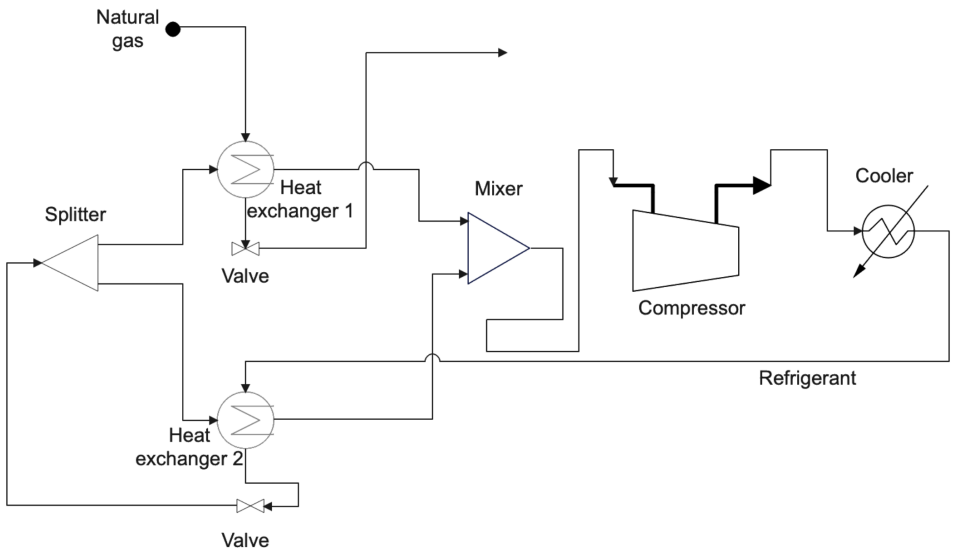


Figure 3.4: Flow chart model with modification

3.2 Property package

The property package used for simulation in the SMR process model has included a natural gas package consisting of components of nitrogen, methane, ethane, propane, and n-butane. The hot natural gas consists of all of these components while the mixed refrigerant consists of the same except for n-butane. Appendix E displays the complete configuration dictionary in the property package. The method for the equations of state has been the Peng-Robinson cubic equations of state. The state variables in the property package have been pressure, temperature, molar flow, and component molar fraction. Another aspect that needed to be specified in the package for the IDAES software was the methods to be used for the calculation of different parameters, those parameters are listed in Table 3.1.

Table 3.1: Methods and references for parameter calculations

	Parameters	Method	Reference
0	Molar liquid density	Perrys	[17]
1	Ideal liquid molar enthalpy	Perrys	[17]
2	Ideal liquid molar entropy	Perrys	[17]
3	Ideal liquid molar heat capacity	Perrys	[17]
4	Ideal gas molar enthalpy	RPP4	[18]
5	Ideal gas molar entropy	RPP4	[18]
6	Ideal gas molar heat capacity	RPP4	[18]
7	Saturation pressure	RPP5	[19]

3.2.1 Methods for calculating parameters

Listed below are all the equations corresponding to the methods utilized by the different references which are implemented in the IDAES software.

Molar liquid density:

$$\rho_{liq} = \frac{C_1}{C_2^{1+(1-\frac{T}{C_3})^{C_4}}} \qquad (3.1)$$
$$\rho_{liq} = C_1 + C_2 \times T + C_3 \times T^2 + C_4 \times T^3$$

Ideal liquid molar enthalpy:

$$h_{\text{liq}} - h_{\text{liq ref}} = C_1 \times (T - T_{ref}) + \frac{C_2}{2} \times (T^2 - T_{ref}^2) + \frac{C_3}{3} \times (T^3 - T_{ref}^3) + \frac{C_4}{4} \times (T^4 - T_{ref}^4) + \frac{C_5}{5} \times (T^5 - T_{ref}^5) + \Delta h_{\text{form, Liq}} \qquad (3.2)$$

Ideal liquid molar entropy:

$$s_{\text{liq}} - s_{\text{liq ref}} = C_1 \times ln(T/T_{ref}) + C_2 \times (T - T_{ref}) + \frac{C_3}{2} \times (T^2 - T_{ref}^2) + \frac{C_4}{3} \times (T^3 - T_{ref}^3) + \frac{C_5}{4} \times (T^4 - T_{ref}^4) + s_{\text{form, Liq}} \tag{3.3}$$

Ideal liquid molar heat capacity:

$$c_{\text{p liq}} = C_1 + C_2 \times T + C_3 \times T^2 + C_4 \times T^3 + C_5 \times T^4 \tag{3.4}$$

Ideal gas molar enthalpy:

$$h_{\text{ig}} - h_{\text{ig ref}} = A \times (T - T_{ref}) + \frac{B}{2} \times (T^2 - T_{ref}^2) + \frac{C}{3} \times (T^3 - T_{ref}^3) + \frac{D}{4} \times (T^4 - T_{ref}^4) + \Delta h_{\text{form, Vap}} \tag{3.5}$$

Ideal gas molar entropy:

$$s_{\text{ig}} = A \times ln(T/T_{ref}) + B \times (T - T_{ref}) + \frac{C}{2} \times (T^2 - T_{ref}^2) + \frac{D}{3} \times (T^3 - T_{ref}^3) + s_{\text{form, Vap}} \tag{3.6}$$

Ideal gas molar heat capacity:

$$c_{\text{p ig}} = A + B \times T + C \times T^2 + D \times T^3 \tag{3.7}$$

Saturation pressure:

$$Log(P_{sat}) = A - \frac{B}{T + C} \tag{3.8}$$

3.2.2 Units of measure

All units must be consistent when working with a large number of expressions and equations. The property package must consist of metadata defining the units of measure for the quantities including time, length, mass, amount of substance, and temperature. All units in the property package are in the SI international system format for units. The units are defined by PYOMO´s unit container and are all based on the base units listed above.

3.3 Read the docs

Read the docs is the documentation provided by IDAES and contains large amounts of information on most of the concepts that the software is containing. It is an important tool to benefit from when trying to understand

how the equations, unit models, property packages, simulation, initialization, optimization, syntax, and raw code, and much more, are working. As the IDAES software is constantly improving, so does the read the docs library. Also, the tutorials and examples provided by IDAES, which is also part of the "*read the docs*", have frequently been used to get to know how the syntax of the coding should be carried out in different process system problems. The resulting model is heavily based upon these examples.

3.4 IDAES

3.4.1 Installation

First of all, one needs to install the programming language that IDAES is based upon, namely Python. This best works in the Jupyter notebook, which is a programming environment that works in correlation with Python. The reason is that all of the tutorials and examples provided by IDAES are shown in the Jupyter notebook. The author has installed the ANACONDA navigator app, which is a library of self-chosen programming environments, including the Jupyter notebook. Once Python is installed, the IDAES module can be found on GitHub, which is an open-source community that shares results of coding and software produced from coding, and is installed by using the general installation pseudo-code "pip install idaes-pse" in the command window. It is advisable to install the IDAES module in a folder that is easy to find, such as on the desktop of the computer.

3.4.2 General workflow

Now as the software is installed, one is ready to start with the fun part, to set up the model and solve it. The general workflow of how to achieve this will be described in the following subsections. This is a general procedure referenced from the documentation of the IDAES "read the docs" [20], which was briefly mentioned in section 3.3.

Importing modules

IDAES contains many code modules that work together to achieve the functions that are desired since this will greatly reduce the amount of coding compared to if one would write all the code in the same code cell block. IDAES is based on object-oriented classes. This means that most of the objects must be imported from the modules, where all components are defined. These components include PYOMO environment components, PYOMO network components, IDAES flowsheet block, the property packages of interest, unit models, data visualization tools, and external user components if desired.

Building the model

Once all the modules are imported, the building of the model can start. This is done by using the environmental component from PYOMO, `ConcreteModel()`. As the model has been produced, the next step is to add a flowsheet. This is done by using the IDAES flowsheet block, `FlowsheetBlock()`. To define the module as steady or dynamic, one will enter "dynamic": True/False in the parenthesis of the `FlowsheetBlock()`. Many commands are based on keys in dictionaries which are more described in detail later.

Property package

The next step is to add a property package to the model, this can either be customized, in which the user must define all state variables & expressions, methods of calculation such as the Peng Robinson equation of state, methods of calculation for the parameters included in property package, constants for calculation of parameters based on what methods is used, units of measure and other. Or one could simply use the already produced collections of property packages defined by IDAES if the user finds this sufficient enough. The property package will then be defined in its own module, and it can be imported to the flowsheet by using a general parameter block.

Unit models

Unit models can now be added to the model by importing the desired modules in which they are defined as components. Some of the unit models of IDAES consist of Heat exchanger, Pressurechanger(Compressor, Valve, Turbine), Reactor, Mixer, Separator, and Flash columns. However, if the user is interested, it is possible to create your own customized unit model, but this requires some extensive coding ability. An example of implementation of the heat exchanger is given with the following code

```
m.fs.My\_heat\_exchanger = HeatExchanger()
```

The property packages must also be assigned to the unit model in the form of a key in the paranthesis. For example

```
m.fs.My_heat_exchanger = HeatExchanger(default=
{"property_package": My_property_package})
```

Connect unit models

As the unit models are defined in the model, the next step is to connect them. This is done by using Arcs, which is a PYOMO type of object that is lines with control volumes connecting each unit model. This means that

whatever flows into the arc is the same flowing out of the arc. For example, connecting the heat exchanger shell outlet to a valve is done by

```
m.fs.S01 = Arc(source=m.fs.My_heat_exchanger.shell_outlet,
destination=m.fs.My_Valve.inlet)
```

Expanding the arcs

An important task is to expand the arcs after they are connected with the unit models. The purpose of expanding the arcs is to produce the constraints that the arcs are to hold, such as control volume properties. The expansion of the arcs is done by using `TransformationFactory`, which is a component found in the PYOMO module. An example of a code snippet fulfilling this is

```
TransformationFactory("network_expand_arcs").apply_to.m
```

where `m` is the name of the model.

Adding variables, constraints and objectives

Now the user can add the desired variables, constraints, and objectives that one wishes to implement into the model. This is simply done by writing code snippets and assigning them to the unit models or variables as one sees fit. An example of an expression calculating the purity of benzene in the outlet of a flash unit model is given in Figure 3.5

In [17]:

```
m.fs.purity = Expression(
        expr=m.fs.F102.vap_outlet.flow_mol_phase_comp[0, "Vap", "benzene"] /
        (m.fs.F102.vap_outlet.flow_mol_phase_comp[0, "Vap", "benzene"]
         + m.fs.F102.vap_outlet.flow_mol_phase_comp[0, "Vap", "toluene"]))
```

Figure 3.5: Expression for purity of benzene in an outlet of a flash unit model

Scaling of the model

Scaling of the model is important to achieve a reliable and efficient solution from the solvers. If the user decides to write their constraints or add variables to the model, it is general good advice to scale these. PYOMO provides a tool which is called `Scaling_factor`. The scaling factor is multiplied by the variables or constraints they scale, for example, a pressure in Pa usually in the order of 10^6, is multiplied by 10^{-5}. IDAES does have inbuilt scaling factors in most of the unit model scripts, so that scaling occurs automatically. The user should anyway be aware of this if it is desired to add their variables and constraints to the model.

Specification of the model

It is time to specify the model by fixing the variables. The variables that need to be fixed is depending on the application but usually include the feed state variables. It is advisable to fully specify the model to prepare for initialization so that there are no degrees of freedom, but it is not necessary for all situations. However, it is safer to do so, as the model will be well defined when it is passed on to the solver.

Initialization

The next step is to initialize the model. All unit models have inbuilt initialization routines and are executed by the following code snippet for a heat exchanger

```
m.fs.My_heat_exchanger.initialize(outlvl=idaeslog.INFO)
```

Here the `outlvl` is the information output from the IDAES logging tool and can be set to 8 different levels based on how much information the user would like to see, here it is set to INFO. Figure 3.6 shows all the different output levels for the IDAES logging tool.

Constant Name	Value	Name	Log Method
CRITICAL	50	CRITICAL	`critial()`
ERROR	40	ERROR	`error()`, `exception()`
WARNING	30	WARNING	`warning()`
INFO_LOW	21	INFO	`info_low()`
INFO	20	INFO	`info()`
INFO_HIGH	19	INFO	`info_high()`
DEBUG	10	DEBUG	`debug()`
NOTSET	0	NOTSET	–

Figure 3.6: Output levels for the IDAES logging tool

Now if the model contains a whole system of connected unit models, one should use a sequential modular approach to initialize the model. Here the unit models are initialized sequentially where the state of the outlet of a unit model is passed on to the inlet of the next unit model. PYOMO provides a sequential decomposition tool, which handles the initialization of the flowsheet based on a heuristic method.

Solving of the model

As the model has been initialized, the preparation for the solver is ready and a feasible solution is passed on to the solver. The solver will iterate until it finds an optimal solution for the process. To solve the model, the PYOMO environment component which was imported in the start, `SolverFactory` is utilized. An example code snippet would look like

```
Solver = SolverFactory("solver_name")
results = solver.solve(m)
```

To verify that the solver has found a feasible solution, it is appropriate to check this by using the command

```
print(results.solver.termination_condition)
```

If everything is fine, a report can be printed out to see the results. An example of such a report for solving a heat exchanger is given in Figure 3.7

```
==========================================================================================
Unit : fs.heat_exchanger                                                    Time: 0.0
------------------------------------------------------------------------------------------
    Unit Performance

    Variables:

    Key            : Value      : Fixed : Bounds
           HX Area :     200.26 : False : (0, None)
    HX Coefficient :     500.00 :  True : (0, None)
         Heat Duty : 4.4423e+06 : False : (None, None)

    Expressions:

    Key             : Value
    Delta T Driving : 44.365
         Delta T In : 78.730
        Delta T Out : 10.000

------------------------------------------------------------------------------------------
    Stream Table
                                Hot Inlet Hot Outlet Cold Inlet Cold Outlet
    Molar Flow (mol/s)              100      100.00        -             -
    Mass Flow (kg/s)              1.8015     1.8015        -             -
    T (K)                         450.00     360.00        -             -
    P (Pa)                        101325  1.0132e+05       -             -
    Vapor Fraction                1.0000     0.0000        -             -
    Molar Enthalpy (J/mol) Vap    50977.     47698.        -             -
    Molar Enthalpy (J/mol) Liq    13489.     6554.3        -             -
    Total Molar Flowrate               -          -      250        250.00
    Total Mole Fraction benzene        -          -  0.40000       0.40000
    Total Mole Fraction toluene        -          -  0.60000       0.60000
    Temperature                        -          -      350        371.27
    Pressure                           -          -   101325    1.0132e+05
==========================================================================================
```

Figure 3.7: Output report for a heat exchanger

3.5 Python

The Python coding language has been extensively used in correlation to the IDAES software, which is based on this language. Jupyter notebook have

been used as the environment in Python, which displays the coding interactively and clearly. All of the tutorials and examples provided by IDAES are also given in the Jupyter Notebook environment, which further strengthens this choice of environment to use when coding. Appendix F shows the complete code for the IDAES model that was made by the python environment.

3.6 Matlab & HYSYS

HYSYS has been an important tool for the verification and validation of results found in the IDAES model. HYSYS has also been exploited to do the PSO together with Matlab where a global optimization toolbox has been utilized. A script was written in Matlab including the objective function & constraints and the run-function itself with all the parameters necessary, such as the size of the swarm, tolerance, maximum iterations as well as the initial swarm matrix. Figure 3.8 shows all the options that were set for the algorithm. The swarm size was set to be 100, as this was a reasonable amount to give relatively good results and effective time management. Stall iterations are iterations giving the same function value as the iteration before. Some nice visuals on the trend of the function value were also established.

```
x0 = [0.3 0.2 0.7 1 0.2 6500 4.5 13.5]; %nN2, nCh4, nC2H6, nC4H10, FR1,
% Molflow, LP, HP

LB =[ 1e-4; 1e-4; 1e-4; 1e-4; 0.1; 2000; 1; 10];
UB =[ 1; 1; 1; 1; 0.9; 10000; 5; 25];

options = optimoptions('particleswarm')
options = optimoptions('particleswarm','SwarmSize',100);
options = optimoptions(options,'FunctionTolerance', 1e-6);
options = optimoptions(options,'MaxIterations', 100);
options = optimoptions(options,'MaxStallIterations', 20); % default=20
options = optimoptions(options,'Display', 'iter');
options = optimoptions(options,'PlotFcn',@pswplotbestf);
options = optimoptions(options,"HybridFcn","patternsearch");
options.InitialSwarmMatrix = x0;
```

Figure 3.8: Options for the PSO

The decision variables, constraints and the objective function is shown in Figure 3.9 and were defined based on a spreadsheet produced in HYSYS. The spreadsheet is displayed in Figure 3.10. "x" represents the swarm vector, consisting of all the decision variables. See Appendix G for the complete Matlab script.
All the values found from the PSO were sent to this spreadsheet and exported into the HYSYS model. A nice feature of the spreadsheet is that it gives an oversight of all the desired variables and can be monitored during the optimization.

```
% variables

hysolver.Cansolve=0;

hyf.Operations.Item('Efficiency').Cell('D2').CellValue = x(1);
hyf.Operations.Item('Efficiency').Cell('D3').CellValue = x(2);
hyf.Operations.Item('Efficiency').Cell('D4').CellValue = x(3);
hyf.Operations.Item('Efficiency').Cell('D6').CellValue = x(4);
hyf.Operations.Item('Efficiency').Cell('B13').CellValue = x(5);
hyf.Operations.Item('Efficiency').Cell('D18').CellValue = x(6);
hyf.Operations.Item('Efficiency').Cell('B8').CellValue = x(7);
hyf.Operations.Item('Efficiency').Cell('B9').CellValue = x(8);
hysolver.Cansolve=1;

% constraints

try
    c(:,1)=hyf.Operations.Item('Efficiency').Cell('B17').CellValue;
    c(:,2) =hyf.Operations.Item('Efficiency').Cell('G9').CellValue;
    c(:,3) =hyf.Operations.Item('Efficiency').Cell('G11').CellValue;
    Power =hyf.Operations.Item('Efficiency').Cell('G4').CellValue;
```

Figure 3.9: Decision variables, constraints and objective function

	A	B	C	D	E	F	G
1	**Variables**	**NG**			**MR**	**Result**	
2	**nN2**	**3.700e-003**	*0.0037*	**0.2659**	*0.1016*	**Compressor**	2.229e+004 kW
3	**nCH4**	**0.9589**	*0.9589*	**0.6083**	*0.2323*	**Cooler**	3.594e+004 kW
4	**nC2H6**	**2.960e-002**	*0.0296*	**0.8901**	*0.3400*	**Total work Comp**	*2.229e+004 kW*
5	**nC3H8**	**7.200e-003**	*0.0072*	**0.0000**	*0.0000*		
6	**nC4H10**	**6.000e-004**	*0.0006*	**0.8540**	*0.3262*		
7				*2.618*	*1.000*		
8	**LP**	**2.471**	*2.471 bar*			**E-100, UA**	5.942e+006 kJ/C-h
9	**HP**	**10.68**	*10.68 bar*			**E-100, Min.T**	3.000 C
10	**T,in**	**293.1**	*293.1 K*			**E-101, UA**	6.437e+007 kJ/C-h
11	**P,in**	**60.00 bar**	*60.00 bar*			**E-101, Min.T**	3.000 C
12							
13	**Flow ratio 1**	**0.1501**	*0.1501*				
14	**Flow ratio 2**	0.8499	*0.8499*				
15	**Tin**	266.9 K					
16	**Tdew**	266.9 K	**Pdew**	2.471 bar	*2.471 bar*		
17	**dTdew**	*6.568e-002 K*					
18	**Molarflow**	**1000 gmole/s**	*1000 gmole/s*	**5198**	*5198 gmole/s*		

Figure 3.10: Spreadsheet called "Efficiency", displaying most variables

3.6.1 Optimization cases

Three cases were optimized with the PSO algorithm. The first two are related to the model as of in Figure 3.11. Here the first case was simulated with the *simple end point* model for the heat exchanger and the second with the *simple weighted* model. These models represent the number of intervals which was used to calculate the minimum temperature approach for the heat exchangers. The simple end point model only calculate the min T. approach at the ends and the simple weighted model calculated the min T. approach at the set intervals, which in this case were 100. The difference between the two models is given in Figure 3.12 and Figure 3.13.

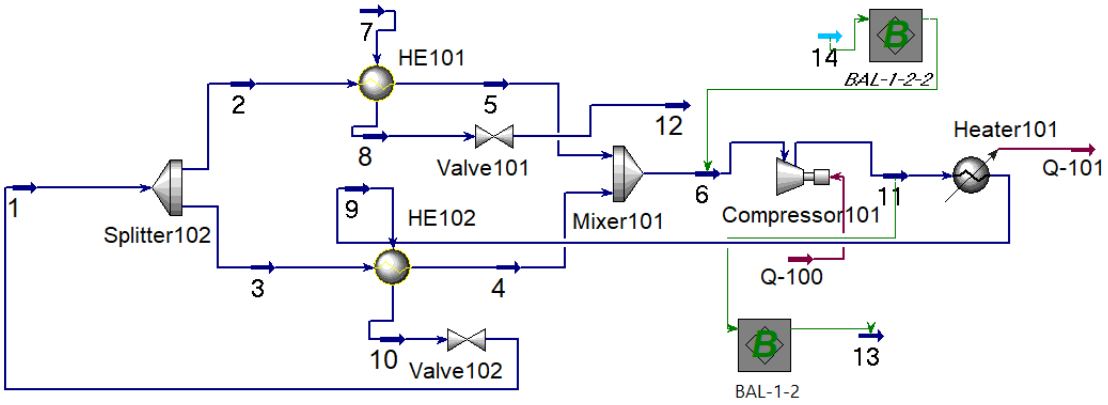


Figure 3.11: SMR process

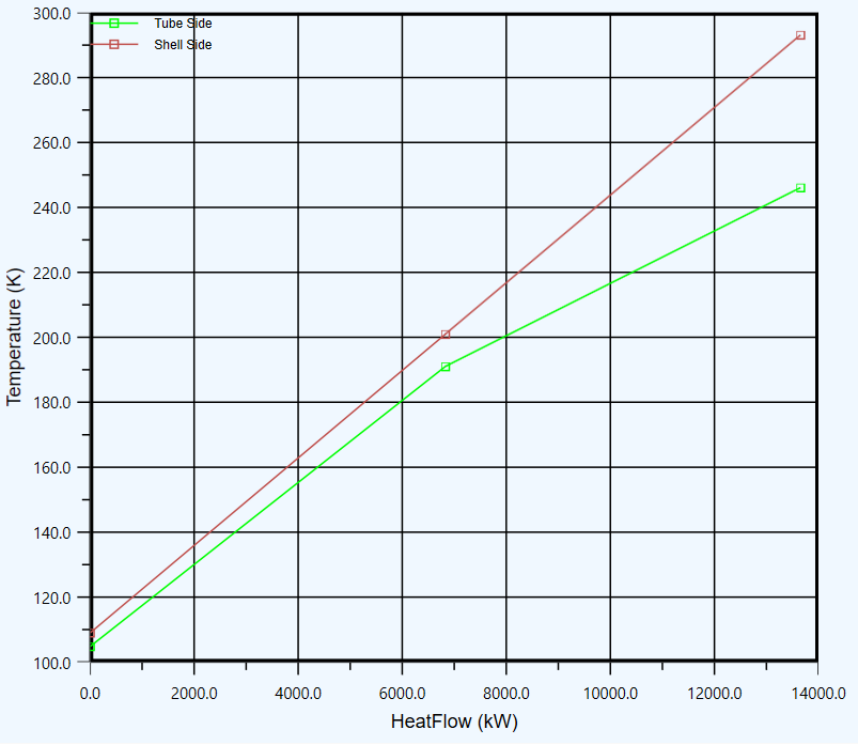


Figure 3.12: Simple end point model for min T. approach, calulated at the ends

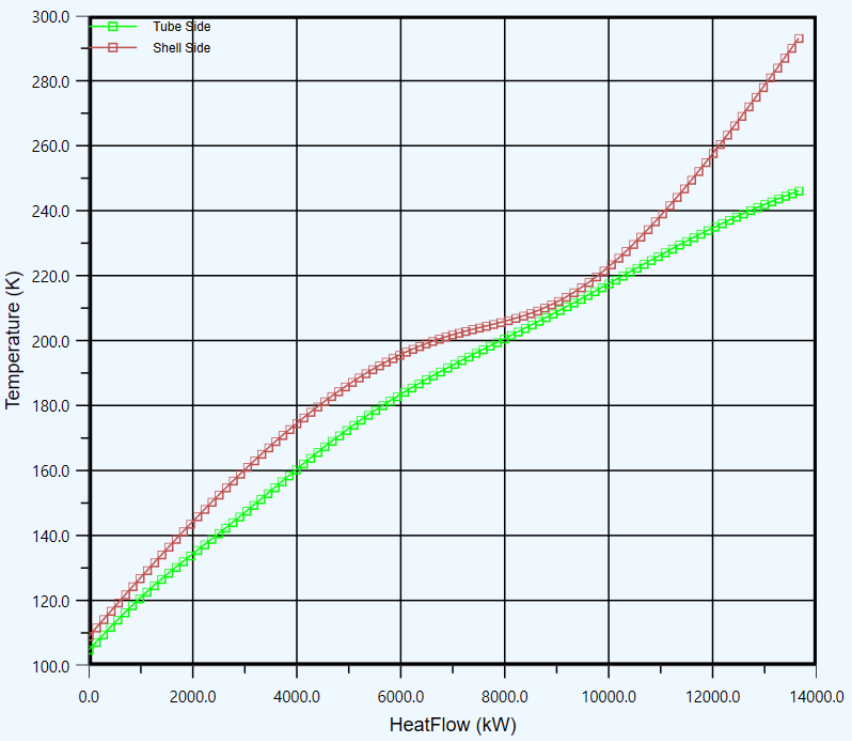


Figure 3.13: Simple weighted model for min T. approach, calulated at each interval

The last case to be optimized was the same SMR process, but instead of the two shell and tube heat exchangers, an MHEX was modeled. Figure 3.14 displays the model. Here the simple weighted model for the MHEX was chosen with a set of 100 intervals.

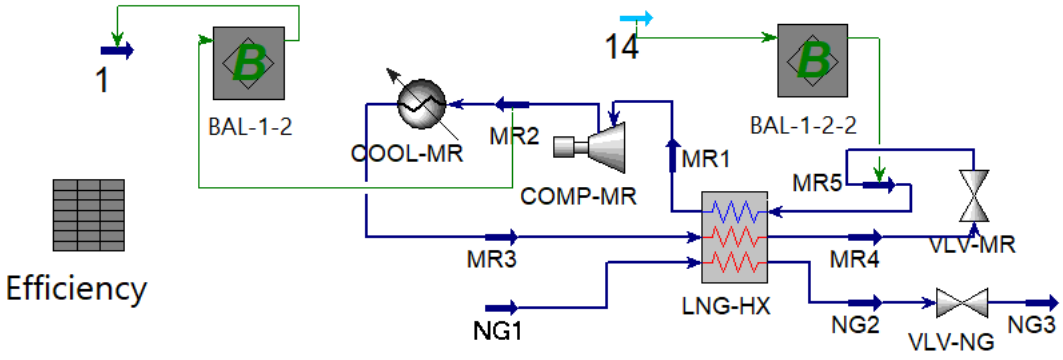


Figure 3.14: SMR process with a multistream heat exchanger

3.7 Particle swarm optimization algorithm

The particle swarm optimization algorithm has been utilized to optimize the operational conditions in the HYSYS models for the three cases mentioned in the last section. The methodology of implementation of this model has been inspired by son et al. [21]. The objective function has been to minimize the power output from the main power-consuming component in the SMR process, namely the compressor unit. To do this a set of decision variables has been determined and also provided is a set of lower and upper bounds of these variables. In addition, some constraints are set and defined to be that the inlet temperature to the compressor must be higher than that of the dew temperature of the fluid and that the minimum temperature approach for each heat exchanger is not less than 3 kelvin such that there would be no liquid in the compressor and a realistic performance could be achieved. The decision variables with bounds are displayed in Table 3.2. More precise they are the mole fraction of each component, low and high pressure, the molar flow rate and the flow ratio asserted to the heat exchangers from the splitter unit. The flow ratio variable only applies to the cases with the s&t heat exchangers. It should be noted that the molar fraction of the total composition cannot exceed 1, and the lower and upper bounds for each of the components is normalized when sent to HYSYS for simulation. The initial swarm matrix can contain an optional value for each variable within the bounds, these were chosen randomly.

Table 3.2: Decision variables with lower and upper bounds

	Decision variables	Lower bound	Upper bound
0	nN2	0.0001	1
1	nCH4	0.0001	1
2	nC2H6	0.0001	1
3	nC4H10	0.0001	1
4	LP	1	5
5	HP	10	25
6	Molar flowrate	2000	10000
7	FR1	0.1	0.9

Pattern search

The general particle swarm solution will find a relatively coarse solution. To find even better solutions, a hybrid function was inserted into the algorithm, where the usual particle swarm simulation routine will be followed by another function of choice, in this case, a *patternsearch* function was added to refine the solution even further. It is a numerical optimization algorithm that does not require any gradients and can be employed on non -continuous

functions, which fits well with the problem at hand.

Chapter 4

Result

In this chapter, the results from the particle swarm optimization are presented for the different modifications and the resulting IDAES model. Firstly the results where the minimum temperature approach is defined by the *simple end point* method in HYSYS, are given in 4.1. To follow, the simulations where the minimum temperature approach defined by the *simple weighted* method is given in section 4.2. Finally, the epilogue of the optimization part of this project is given in section 4.3 where the configuration with the LNG heat exchanger is optimized by the PSO algorithm. The cyan color in the following tables marks the best solution found.
At last the IDAES model is presented in section 4.4, with a short explanation of how the results were obtained including some challenges which were met along the way. Also included is a comparison with HYSYS for the similar IDAES model.

4.1 PSO with S&T *Simple end point* modification

Results from the particle swarm optimization where the end-to-end temperature difference of the heat exchangers is defined for the calculation of the minimum temperature approach are given in Table 4.1, where all the decision variables and the power output are listed from all 20 simulations. Table 4.2 shows the corresponding constraints. It can be seen that simulation # 3, has the lowest power output and hence is the best result. By taking the variables found from this best solution, a new simulation was completed to further improve this configuration. The refined solutions are given in Table 4.3 with the corresponding constraints in Table 4.4.

Table 4.1: Resulting decision variables & power output from simulation

#	nN2	nCH4	nC2H6	nC4H10	FR	Molarflow	LP	HP	Power
	[-]	[-]	[-]	[-]	[-]	[mol/s]	[bar]	[bar]	[kW]
0	0.3048	0.1785	0.1694	0.3473	0.1561	3965.81	5	10.0006	8220.96
1	0.2378	0.0098	0.3951	0.3573	0.2162	2455.5	4.8902	12.3817	6736.07
2	0.3648	0.2359	0	0.3993	0.2034	3123.2	3.9717	10	8789.7
3	0.1506	0.033	0.3942	0.4221	0.2402	2015.42	4.3256	10.1273	5028.77
4	0.2239	0.0175	0.4506	0.3079	0.1174	4656.13	4.9999	10.0004	9420.47
5	0.2114	0.058	0.2945	0.4361	0.2559	2000	3.4028	10.493	6783.55
6	0.2737	0.0004	0.3242	0.4018	0.2435	2167.07	4.7028	11.2966	5640.07
7	0.5099	0.0128	0.023	0.4543	0.2875	2154.58	4.2748	10.5396	5938.85
8	0.4438	0.1067	0.0253	0.4242	0.2433	2591.49	4.2655	10.1721	6837.92
9	0.149	0.1822	0.3263	0.3425	0.1458	3696.1	5	10.0019	7491
10	0.3521	0	0.2936	0.3542	0.1702	3400.96	5	10.1052	7083.8
11	0.13	0.1908	0.2749	0.4043	0.2111	2415.13	4.7284	10.0409	5359.6
12	0.1532	0.1944	0.2213	0.4311	0.2341	2194.69	4.3741	10	5378.86
13	0.2227	0.0001	0.4484	0.3289	0.1567	3397.1	4.9908	10.3161	7284.67
14	0.2405	0.1269	0.1901	0.4425	0.2448	2153.45	4.0687	10	5773.65
15	0.1563	0.1075	0.3748	0.3614	0.1744	2998.94	4.9999	10	6071.68
16	0.3026	0.0177	0.3264	0.3533	0.1673	3357.15	4.9905	10.0564	6928.59
17	0.1353	0.0192	0.4222	0.4233	0.2385	2000.3	4.0961	10.0453	5241.14
18	0.122	0.0112	0.4842	0.3826	0.2402	2006.65	4.859	11.7779	5178.33
19	0.4221	0	0.2104	0.3674	0.1885	3371.84	4.7393	10	7510.65

Table 4.2: Resulting constraints from simulation

#	> Dew T	Min Temp. Approach HE1	Min Temp. Approach HE2
	[K]	[K]	[K]
0	1.7012	3.979	3.341
1	0.0096	4.1647	4.1647
2	2.2255	5.5262	5.5262
3	0.3183	3.0091	3.0091
4	2.0481	3.0001	3.0001
5	4.25	4.0716	4.0716
6	0.0022	3.1381	3.2249
7	0.0018	3.3569	3.0588
8	0.0791	4.552	4.552
9	0.1242	3.6099	3.6099
10	0.2061	2.9871	3.0001
11	0.0022	3.0012	3.017
12	0.0012	3.7072	3.0053
13	3.8667	3	3
14	0.0009	3.1278	3.8615
15	0.1481	3.1376	3.1376
16	0.0599	3	3
17	0.0232	3.0002	3.0002
18	0.0017	3.0009	3.0042
19	0.0016	3.2198	3.0052

Table 4.3: Refinement of solutions

#	nN2	nCH4	nC2H6	nC4H10	FR	Molarflow	LP	HP	Power
	[-]	[-]	[-]	[-]	[-]	[mol/s]	[bar]	[bar]	[kW]
0	0.1504	0.0333	0.3936	0.4227	0.2402	2015.34	4.3368	10.1252	5005.46
1	0.1505	0.0332	0.3933	0.423	0.2404	2012.57	4.3361	10.1264	5000.95
2	0.1501	0.0337	0.3934	0.4227	0.2405	2011.75	4.3402	10.1257	4995.31
3	0.1509	0.033	0.3931	0.423	0.2401	2014.92	4.3285	10.119	5012.17
4	0.1502	0.0331	0.3936	0.4232	0.2422	2007.41	4.3334	10.1272	4994.32

Table 4.4: Resulting constraints from refinement

#	> Dew T	Min Temp. Approach HE1	Min Temp. Approach HE2
	[K]	[K]	[K]
0	0.0005	3	3
1	0.0113	3	3
2	0.1366	3	3
3	0	3.0001	3.0001
4	0.1209	3	3

It seems in this case that a higher composition of ethane and n-butane is favored and the amount of methane is nearly 5 times lower than that of nitrogen. A relatively low molar flow in the system just above 2000 mol/s is also observed. Noticeably is also the remarking low power output produced, which is likely due to the configuration of the method for calculation of the minimum temperature approach. This result could be expected, and in the next two sections where the method is set to be the *simple weighted* method for both cases, we can expect a higher power output as this is perhaps more precise and displays a higher degree of realism.

4.2 PSO with S&T *Simple weighted* modification

Provided here are the results from ten simulations with the *simple weighted* method for calculation of the minimum temperature approach. The result yields quite a difference from the result in the previous section with the *simple end point* method. From Table 4.5 simulation #0 gives the best result. It seems that the composition of the SMR in simulation #0 is more balanced between the three heavier hydrocarbons and prefers lesser nitrogen. Combined with a relatively low LP and high HP with a flow ratio that exceeds most of the other simulations. Table 4.6 shows the related constraints from the simulation.

Table 4.5: Resulting decision variables from simulation

#	nN2	nCH4	nC2H6	nC4H10	Flow ratio	Molarflow	LP	HP	Power
	[-]	[-]	[-]	[-]	[-]	[mol/s]	[bar]	[bar]	[kW]
0	0.0888	0.3181	0.3053	0.2878	0.2028	3766.92	2.5498	17.5441	22071.4
1	0.1294	0.2976	0.3545	0.2186	0.1487	5395.65	4.4244	19.454	24093.9
2	0.1361	0.2919	0.3353	0.2366	0.12	6921.61	4.8489	14.8743	22821.7
3	0.1387	0.2623	0.3853	0.2137	0.1105	7474.11	4.9259	15.9047	26034
4	0.2074	0.2676	0.2799	0.2451	0.1116	7557.05	4.4519	14.2837	26111.4
5	0.0728	0.1811	0.3945	0.3516	0.2303	2990.43	1.1813	13.3352	22344.9
6	0.2667	0.2092	0.3172	0.2069	0.131	6513.06	5	20.4384	27609.4
7	0.1352	0.3226	0.2937	0.2485	0.1	8376.54	4.9854	12.855	23231.6
8	0.2078	0.2291	0.2319	0.3312	0.18	4462.83	1.6619	13.515	28177.9
9	0.1016	0.2323	0.34	0.3262	0.1501	5198.03	2.4715	10.6805	22289.2

Table 4.6: Resulting constraints from simulation

#	> Dew T	Min Temp. Approach HE1	Min Temp. Approach HE2
	[K]	[K]	[K]
0	2.2599	3.0011	3.0009
1	2.8172	3.0003	3.0034
2	0.0487	3.0005	3.0409
3	3.5624	3.0005	3.0021
4	0.0016	3.0006	3.0007
5	10.3559	3.0007	3.0004
6	0.0025	3.0089	3.0002
7	0.0001	3	3.0002
8	0.0025	3.0002	3.0017
9	0.0656	3	3

4.3 PSO with LNG heat exchanger

This section is the last of the optimization part and displays the simulations with the case for the LNG heat exchanger. According to Table 4.7 simulation # 1, 3, and 5 gives the best result with a power output just below 17000kW. It looks like the composition of the SMR is relatively close to each other with a slight change in the molar flow. The low pressure and high pressure also seem to be rather close when compared. As usual, the resulting constraints follow in Table 4.8.

Table 4.7: Resulting decision variables & power output from simulation

#	nN2	nCH4	nC2H6	nC4H10	Molarflow	LP	HP	Power
	[-]	[-]	[-]	[-]	[mol/s]	[bar]	[bar]	[kW]
0	0.1104	0.2525	0.3546	0.2825	3275.98	3.2982	18.529	17106.7
1	0.1042	0.2648	0.3542	0.2767	3343.76	3.1798	15.8316	16967.1
2	0.1136	0.2635	0.3636	0.2593	3270.97	3.5543	18.4432	17031.7
3	0.1047	0.2627	0.3553	0.2774	3218.17	3.1069	16.4152	16986.3
4	0.0848	0.2409	0.3433	0.331	2698.84	1.8866	14.1025	17469
5	0.1088	0.2695	0.3573	0.2644	3658.26	3.6059	15.8217	16967.5
6	0.1316	0.2665	0.3765	0.2254	3441.04	4.4794	21.8315	17338.4
7	0.1264	0.2745	0.3628	0.2363	4148.29	4.4466	16.7536	17277
8	0.1159	0.2631	0.3649	0.2561	3521.04	3.8435	18.27	17040.5
9	0.09	0.2488	0.3383	0.323	3055.02	2.2965	14.0224	17207.7
10	0.1329	0.2755	0.3638	0.2278	4305.06	4.7405	17.2272	17429
11	0.1228	0.271	0.369	0.2372	3682.1	4.2033	18.1211	17105.9
12	0.1302	0.2626	0.3641	0.2431	4299.95	4.6401	17.8523	17439.1
13	0.1021	0.2516	0.3507	0.2956	2892.15	2.6687	17.6565	17179.7
14	0.106	0.2679	0.3466	0.2795	4001.12	3.5412	14.4352	17128.1
15	0.1159	0.2707	0.3611	0.2523	3789.52	3.9377	16.5452	17059
16	0.0758	0.2421	0.3325	0.3497	3099.08	1.7886	10.6578	17741.6
17	0.0891	0.2527	0.3307	0.3275	3489.48	2.4905	12.7119	17236.3
18	0.0724	0.2155	0.3165	0.3957	2536.18	1.2919	13.7672	17988
19	0.1022	0.2608	0.3501	0.287	3356.9	3.0612	15.5718	17000

Table 4.8: Resulting constraints from simulation

#	> Dew T	Min Temp. Approach HE
	[K]	[K]
0	2.1204	3
1	15.6815	3
2	14.2706	3
3	15.9114	3
4	20.5937	3
5	14.224	3
6	15.0516	3
7	15.0909	3
8	9.9252	3
9	11.958	3
10	14.9811	3
11	17.1046	3
12	3.2885	3
13	11.9353	3
14	7.4161	3
15	14.1072	3
16	23.4657	3
17	6.2145	3
18	1.758	3
19	11.5594	3

4.4 IDAES model

It was quite challenging to achieve a converged solution in the IDAES model with the given initial values. First, the initialization had to converge, so it could be sent to the solver. For the initialization to converge, the need to set good initial values for the different variables in the model was important. Once the solver had found an optimal solution, it was necessary to unfix different variables to get the values for the variables that were desired. Especially the temperature for the shell outlets of the heat exchangers had to be low enough so that all the fluid through these had become liquid. Figure 4.1 displays the resulting model and Figure 4.2 gives an oversight of the state variables and the composition of the fluid at each arc in a stream table. Both figures are created by the IDAES visualization tool.

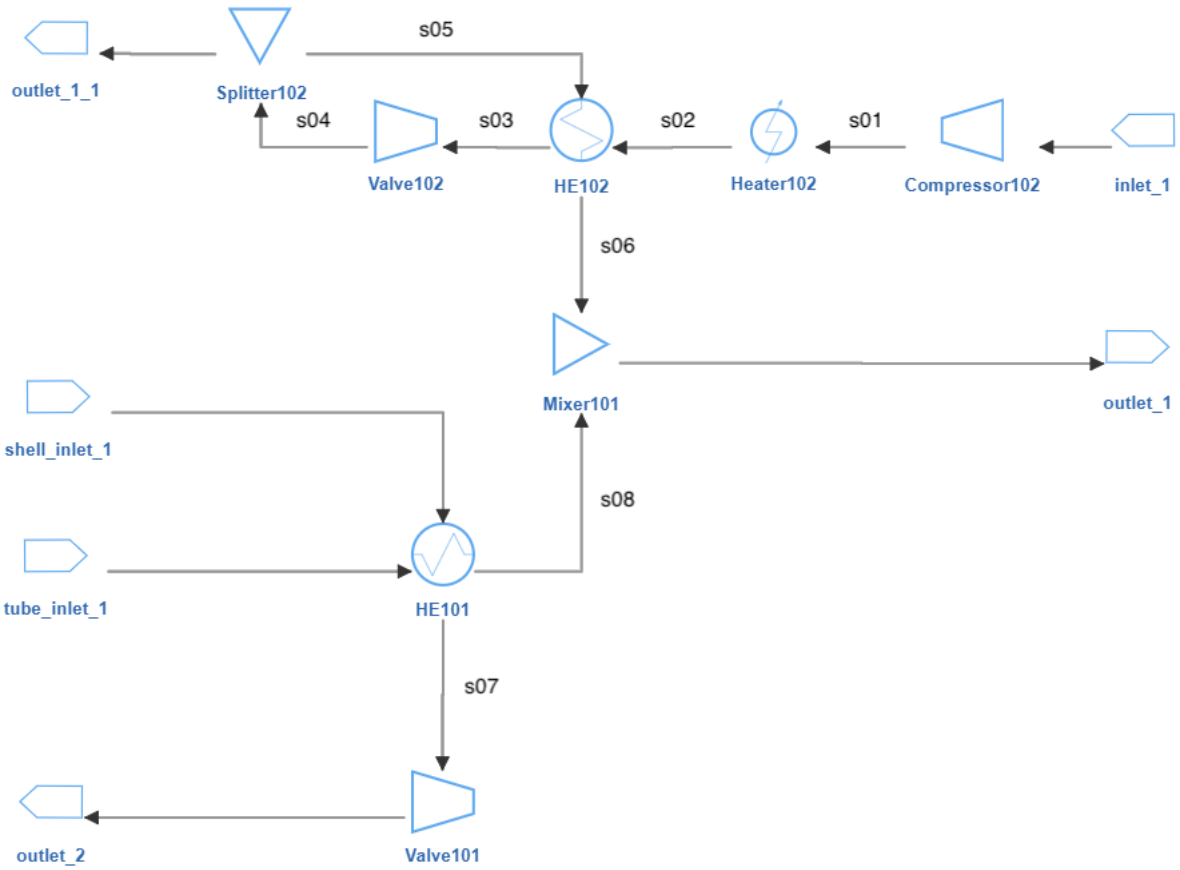


Figure 4.1: Modified IDAES model

Variable		s01	s02	s03	s04	s05	s06	s07	s08
Total Molar Flowrate	mol/s	3440	3440	3440	3440	3268.22927	3268.22927	1000	1711.59305
Total Mole Fraction nitrogen	–	0.0959	0.0959	0.0959	0.0959	0.0959	0.0959	0.0037	0.0959
Total Mole Fraction methane	–	0.2704	0.2704	0.2704	0.2704	0.2704	0.2704	0.9589	0.2704
Total Mole Fraction ethane	–	0.3603	0.3603	0.3603	0.3603	0.3603	0.3603	0.0296	0.3603
Total Mole Fraction nbutane	–	0.2734	0.2734	0.2734	0.2734	0.2734	0.2734	0.0006	0.2734
Temperature	K	369.90924	293	112	111.88967	111.88967	265.13553	112	189.20482
Pressure	kg/(m s^2)	1550000	1550000	1550000	350000	350000	350000	3000000	350000
Total Mole Fraction propane	–	-	-	-	-	-	-	0.0072	-

Figure 4.2: Streamtable

The model had to be modified several times for achieving a final result, which is not perfect, but still resembles a valid solution. Referring back to Figure 3.4, it can be seen that the final model looks differently. The reason it had to be modified is a result of many factors. A summary of the challenges met, and how they were dealt with follows in the next section. A short report of both the heat exchangers is given in Figure 4.3 and Figure 4.4. A report for all the unit models in the system can be found in Appendix A.

```
==========================================================================================
Unit : fs.HE101                                                            Time: 0.0
------------------------------------------------------------------------------------------
    Unit Performance

    Variables:

    Key            : Value      : Fixed : Bounds
           HX Area :     944.08 : False : (0, None)
    HX Coefficient :     944.08 : False : (0, None)
         Heat Duty : 1.3492e+07 : False : (None, None)

    Expressions:

    Key             : Value
    Delta T Driving :  15.137
         Delta T In :  103.80
        Delta T Out : 0.11000

------------------------------------------------------------------------------------------
    Stream Table
                                 Hot Inlet  Hot Outlet Cold Inlet Cold Outlet
    Total Molar Flowrate            1000.0      1000.0     1711.6      1711.6
    Total Mole Fraction nitrogen 0.0037000   0.0037000   0.095900    0.095900
    Total Mole Fraction methane    0.95890     0.95890    0.27040     0.27040
    Total Mole Fraction ethane    0.029600    0.029600    0.36030     0.36030
    Total Mole Fraction propane  0.0072000   0.0072000          -           -
    Total Mole Fraction nbutane 0.00060000  0.00060000    0.27340     0.27340
    Temperature                     293.00      112.00     111.89      189.20
    Pressure                   3.0000e+06  3.0000e+06 3.5000e+05  3.5000e+05
==========================================================================================
```

Figure 4.3: Report for HE101

```
==========================================================================================
Unit : fs.HE102                                                              Time: 0.0
------------------------------------------------------------------------------------------
    Unit Performance

    Variables:

    Key            : Value      : Fixed : Bounds
           HX Area :     3687.6 : False : (0, None)
    HX Coefficient :     3687.6 : False : (0, None)
         Heat Duty : 6.8226e+07 : False : (None, None)

    Expressions:

    Key             : Value
    Delta T Driving :  5.0173
         Delta T In :  27.864
        Delta T Out : 0.11033

------------------------------------------------------------------------------------------
    Stream Table
                                Hot Inlet  Hot Outlet  Cold Inlet  Cold Outlet
    Total Molar Flowrate           3440.0      3440.0      3268.2      3268.2
    Total Mole Fraction nitrogen  0.095900    0.095900    0.095900    0.095900
    Total Mole Fraction methane    0.27040     0.27040     0.27040     0.27040
    Total Mole Fraction ethane     0.36030     0.36030     0.36030     0.36030
    Total Mole Fraction nbutane    0.27340     0.27340     0.27340     0.27340
    Temperature                     293.00      112.00      111.89      265.14
    Pressure                   1.5500e+06  1.5500e+06  3.5000e+05  3.5000e+05
==========================================================================================
```

Figure 4.4: Report for HE102

4.4.1 Comparison with HYSYS

To validate the results a similar model was constructed in HYSYS with the same specifications. Figure 4.5 shows the model and Figure 4.6 shows the corresponding stream table. The bold blue numbers represent the variables that were specified in the model. In an ideal model, stream 12 is supposed to be stream 1, and stream 14 is supposed to be stream 10, but as a closed configuration did not work in IDAES this configuration had been made.
Some values resembles what is found in IDAES, such as the temperature in the outlet of the compressor and the tube outlet streams from the heat exchangers. Hysys find that the temperature in the tube inlet of HE102 should be around 109K but IDAES find it to be around 111.8K. The outlet temperature for the mixer unit found in IDAES is according to the report in Figure A.6 245K and in HYSYS about 246.7K, which is not too far off. The temperature of valve101 from the report in FigureA.3 shows an increase in temperature between the outlet and inlet of about 1.25K while in HYSYS a decrease of about 0.5K.

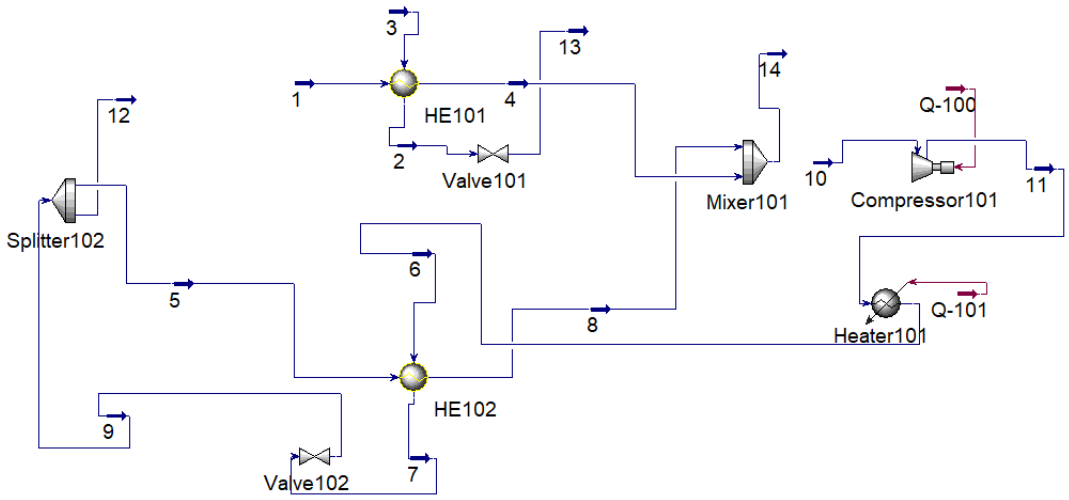


Figure 4.5: Model in HYSYS

Case - Material Stream														
Name	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Pressure [bar]	**3.5000**	30.0000	**30.0000**	3.5000	3.5000	15.5000	15.5000	3.5000	**3.5000**	**3.5000**	**15.5000**	3.5000	**1.0500**	3.5000
Temperature [K]	**111.8900**	**112.0000**	**293.0000**	192.0014	109.0757	**293.0000**	**112.0000**	265.3554	109.0757	**287.0000**	370.0313	109.0757	111.5249	246.7076
Vapor / Phase Fraction	0.0650	0.0000	1.0000	0.4364	0.0564	0.7874	0.0000	0.9171	0.0564	1.0000	1.0000	0.0564	0.0110	0.7606

Figure 4.6: Stream table HYSYS

4.4.2 Challenges along the way

Here are some of the more troublesome challenges encountered with the model listed and a short explanation of how they were dealt with are presented.

- Degrees of freedom
- Initialization
- Overflow
- Solving and refinement of solution

Degrees of freedom

As the model was built up corresponding to Figure 3.4, the flowchart was created and arcs connecting all the unit models were established. When expanding the arcs, which is a very important task due to fixing the constraints related to each unit model and physical properties within each arc, the degrees of freedom were too few to be able to get the model initialized due to under specification of the model. The reason was that model was in a closed-loop, and a breakpoint had to be implemented. This breakpoint was chosen to be between the Mixer unit and the Compressor unit. Once done, the degrees of freedom increased and the model could be specified.

Initialization

As the model had been specified the initialization procedure could take form. This was done by the *SequentialDecomposition* tool, which initializes each unit model in a heuristic manner based on the arc connections. The initialization went smoothly up until the heat exchangers, where an error occurred. This was yet again a degree of freedom error that turned out to be caused by specifying the outlet temperatures in the shell outlet of the heat exchangers. Specifying outlet temperatures of the heat exchanger can only be done after the model is solved by unfixing some of the degrees of freedom of the heat exchanger, such as the overall heat transfer coefficient or area, and fixing the outlet temperature followed by utilizing the solver.

Overflow

The Wegstein method in the initialization procedure performs when there is a loop involved, which was still the case, as the shell outlet of heat exchanger HE102 was indirectly connected to the tube inlet. This loop will be initialized several times until it converges. As the initialization had run several times, an *overflow* error occurred, see Appendix B for the output error. This is due to a mathematical error occurring in the calculations, which was not fully understood by the author. To overcome this error a new breakpoint was tried and implemented by unfixing the original arc connecting the Splitter102 to the tube inlet of heat exchanger HE101. This seemed to fix the problem, but it comes with a slight fallacy. Namely that the mass balance in the Compressor101 is not the same as the mass balance in the Mixer101.
On the other hand, this solved issues related to scaling of the heat exchangers, as there was trouble getting them to converge when inputting a given flow ratio, see Appendix C for an example of an output of such an error. The Splitter102 would now assign a flow ratio to the tube inlet of HE102 that was appropriate to solve the heat exchanger.

Solving and refinement of solution

At this point the model looks as in Figure 4.1, the initialization can converge to a solution and the solver found the optimal solution. However, the temperatures in the shell outlets of both heat exchangers were not the desired ones and needed to be fixed. This was done by unfixing the heat transfer coefficient and the inlet flow ratio at the tube inlets of the heat exchangers, followed by fixing the temperature shell outlets of the heat exchangers. The difference in value between the previous value for the shell outlet temperature and the newly set outlet temperature could not be high, for the model to converge when sent to the solver. This resulted in repeating this procedure until the desired temperature was reached. To avoid multiple initializations and solving routines when doing this refinement, a .json file was created

and put in the same path as the model. This file includes previous results for initialization and solving, and resulted in quick solving steps when refining the shell outlet temperatures.

Chapter 5

Discussion and conclusion

The IDAES model offered some nice results in regards to the production of LNG in the end, but still included some small inconsistencies from the HYSYS model. There is not a single answer to why this is but is due to several factors the author believes. First off is the property package which had to be created. The package included lots of parameters that had to be defined by given values in different literature, and some of these are not the same as defined in HYSYS. Figure D.1 is an example of the coefficients used to calculate the ideal gas enthalpy in HYSYS, which differs from that of the found values in the package applied in IDAES, see Appendix E. The reason is that HYSYS uses different methods to calculate properties than that of IDAES for some properties. The complexity of HYSYS is more distinguished in its way than IDAES complexity is in its own. HYSYS relies on inbuilt property packages and brute force calculations with distinct optimization methods and a rather simple way of setting up a model and solving it. The unit models in HYSYS have a tremendous amount of design inputs and customization of different settings. IDAES relies almost everything on customized coding and object-oriented formulation. The degree of complexity related to the unit models in IDAES and design parameters does not come close to that of HYSYS. The methodology in the procedure of solving the model in IDAES is unique in one perspective as asserting desired values for variables in some unit models can only be done after a general valid solution has been obtained from the specified inputs, and the user must solve the model several times until the preferred result occurs. In HYSYS the assertion of values for variables for the majority of the unit models is independent of first finding a generic solution for the model.
From the results of the optimization of the different HYSYS models, it was interesting to see the difference in the power output from each of the models. Perhaps the most interesting results came from the simulations done with the *simple weighted* method for the heat exchangers as these offered the most realistic outcomes. However, the result from the *simple end point* method for the heat exchangers can be used to explore the main factors con-

tributing to the best solution based on the decision variables found. It should also be mentioned that if the results from the *simple end point* method were put into the model with the *simple weighted* method, it would appear as an infeasible solution. Maybe more exclusively the best results in this configuration yielded on average a lesser mole fraction of methane than that of the other two configurations. In these, it was quite similar distributions of the heavier elements for the best solutions. It should be noted that all the values found from the optimization most likely do not correspond to global minimums, but firmly local minimums. If the population of the swarm had been increased as well as the maximum iterations, the algorithm would find even better solutions, the dilemma becomes of how much time should be assigned to achieve a good enough result.

Conclusion

The IDAES framework offers a good way to simulate and model a process system and provides the user with a high portion of customizability for creating their unit models and property packages as well as methods to calculate different functions and variables. IDAES also offers a variety of libraries of defined unit models and some property packages that could be changed or edited if desired. Perhaps the most noteworthy downside of the IDAES software as it stands today is that it is not generic enough to reach out to all possible clientele within the industry. It requires good knowledge of coding and the use of correct syntax in different operations. They do have tutorials provided, but once the user is to set up their own model everything doesn't necessarily work just as planned. Errors occurring without a proper formulation of what is causing it is not clear and this leads to troubleshooting. However, a discussion forum on troubleshooting is present on the IDAES GitHub page, which has been utilized for some of the troubleshooting in this thesis. IDAES is still a relatively new software, and version 2.0 has just been released. In that perspective, it has the potential to reach great heights in the coming future in its goals to become one of the leading process and modeling software in the industry.

Optimization is an outstanding tool to find the best configuration for many problems. Learning to understand how nature works and in this thesis, particularly how birds interact with each other and a small implementation of craziness or perhaps more formally said, stochastic features, forms the basis of an optimization algorithm. By studying these fields, the genetic algorithm can be implemented in many engineering problems to solve difficult nonlinear functions. The particle swarm optimization showed that the power output from the compressors depended on the composition of the fluid, the pressure ratio of the compressor, and the flow rate of the fluid. From the results, it can be concluded that higher mole fractions of the heavier com-

ponents reduce the power output and that the distribution between the mole fractions is almost universal.

Future work

Future work should consist of investigating further how to solve complete closed-loop systems in IDAES, as this was what caused a relatively sizeable amount of trouble in the model. Also interesting would be to create a multistream heat exchanger by utilizing the tools IDAES provides to accomplish this. Perhaps by borrowing some inspiration from how the LNG heat exchanger is built and designed in HYSYS. IDAES provides some information regarding this in the documentation and as for the raw code, it would be possible to look at how the existing unit models are built in the IDAES library. It would be interesting to also include optimization within IDAES, by utilizing the different optimization solvers they offer.
Another aspect of future work could include the use of different genetic algorithms to optimize the SMR process or other liquefaction processes of natural gas, followed by a comparison of the results to find the most suited algorithm that best comes close to a global optimum.

Bibliography

[1] J. P. Riva, G. I. Atwater, L. H. Solomon, J. E. Carruthers and A. Waddams, *Natural gas*, en. [Online]. Available: https://www.britannica.com/science/natural-gas.

[2] C. for climate and energy solutions, *Natural gas*, en. [Online]. Available: https://www.c2es.org/content/natural-gas/.

[3] T. Morosuk, S. Tesch, A. Hiemann, G. Tsatsaronis and N. Bin Omar, 'Evaluation of the PRICO liquefaction process using exergy-based methods,' en, *Journal of Natural Gas Science and Engineering*, vol. 27, pp. 23–31, Nov. 2015, ISSN: 18755100. DOI: 10.1016/j.jngse.2015.02.007. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1875510015000657 (visited on 22/05/2022).

[4] IDAES, *IDAES Integrated platform*, en. [Online]. Available: https://idaes.org/idaes-integrated-platform/.

[5] A. TECH, *Hyprotech: Simulation software for industry*, en. [Online]. Available: https://web.archive.org/web/20140920080106/http://www.ucalgary.ca/community/research/hyprotech.

[6] Wikipedia, *Aspen HYSYS*, dn. [Online]. Available: https://en.wikipedia.org/wiki/Aspen_HYSYS.

[7] Wikipedia, *Wan der Waal ´s equation of state*, en. [Online]. Available: https://en.wikipedia.org/wiki/Cubic_equations_of_state.

[8] D.-Y. Peng and D. B. Robinson, 'A New Two-Constant Equation of State,' en, *Industrial & Engineering Chemistry Fundamentals*, vol. 15, no. 1, pp. 59–64, Feb. 1976, ISSN: 0196-4313, 1541-4833. DOI: 10.1021/i160057a011. [Online]. Available: https://pubs.acs.org/doi/abs/10.1021/i160057a011 (visited on 28/11/2021).

[9] Lexico, *Oxford english dictionary*, en. [Online]. Available: https://www.lexico.com/definition/optimization.

[10] Wordreference, *Online dictionary*, en. [Online]. Available: https://www.wordreference.com/definition/optimization.

[11] M. Cavazzuti, *Optimization Methods*, en. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-31186-4 978-3-642-31187-1. DOI: 10.1007/978-3-642-31187-1. [Online]. Available: http://link.springer.com/10.1007/978-3-642-31187-1 (visited on 30/11/2021).

[12] J. Kennedy and R. Eberhart, 'Particle swarm optimization,' en, in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, Perth, WA, Australia: IEEE, 1995, pp. 1942–1948, ISBN: 978-0-7803-2768-9. DOI: 10.1109/ICNN.1995.488968. [Online]. Available: http://ieeexplore.ieee.org/document/488968/ (visited on 31/05/2022).

[13] F. Heppner and U. Grenander, *A stochastic nonlinear model for coordinate bird flocks*, en, Jan. 1990. [Online]. Available: https://www.researchgate.net/publication/216300775_A_Stochastic_Nonlinear_Model_for_Coordinate_Bird_Flocks.

[14] C. W. Reynolds, 'Flocks, Herds, and Schools: A Distributed Behavioral Model,' en, p. 21,

[15] Pagmo, *Particle swarm optimization*. [Online]. Available: https://esa.github.io/pagmo2/docs/cpp/algorithms/pso.html.

[16] H. N. Rao, S. K. Nair and I. A. Karimi, 'Operational Optimization of Processes with Multistream Heat Exchangers Using Data-Driven Predictive Modeling,' en, *Industrial & Engineering Chemistry Research*, vol. 58, no. 15, pp. 5838–5850, Apr. 2019, ISSN: 0888-5885, 1520-5045. DOI: 10.1021/acs.iecr.8b05270. [Online]. Available: https://pubs.acs.org/doi/10.1021/acs.iecr.8b05270 (visited on 01/06/2022).

[17] R. H. Perry, D. W. Green and J. O. Maloney, Eds., *Perry's chemical engineers' handbook*, en, 7th ed. New York: McGraw-Hill, 1997, ISBN: 978-0-07-049841-9.

[18] R. C. Reid, J. M. Prausnitz and B. E. Poling, *The Properties og Gases and Liquids 4th edition*, 4th ed. McGraw -Hill, inc, 1987. [Online]. Available: https://www.academia.edu/33161453/The_Properties_of_Gases_and_Liquids_4th_Edition_R_C_Reid_J_M_Prausnitz_and_B_E_Poling_.

[19] J. P. O´Connel, J. M. Prausnitz and B. E. Poling, *The Properties og Gases and Liquids 5th edition*, 5th ed. McGraw -Hill, inc, 2001. [Online]. Available: https://www.accessengineeringlibrary.com/content/book/9780070116825.

[20] IDAES, *General workflow*, en. [Online]. Available: https://idaes-pse.readthedocs.io/en/latest/how_to_guides/workflow/general.html.

[21] H. Son, B. Austbø, T. Gundersen, J. Hwang and Y. Lim, 'Techno-economic versus energy optimization of natural gas liquefaction processes with different heat exchanger technologies,' en, *Energy*, vol. 245, p. 123 232, Apr. 2022, ISSN: 03605442. DOI: 10.1016/j.energy.2022.123232. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0360544222001359 (visited on 01/06/2022).

Appendix A

Unit model reports

```
==========================================================================================
Unit : fs.Compressor102                                                      Time: 0.0
------------------------------------------------------------------------------------------
    Unit Performance

    Variables:

    Key                    : Value      : Fixed : Bounds
    Isentropic Efficiency :    0.80000 :  True : (None, None)
          Mechanical Work : 1.6128e+07 : False : (None, None)
          Pressure Change : 1.2000e+06 : False : (None, None)
           Pressure Ratio :     4.4286 : False : (None, None)

------------------------------------------------------------------------------------------
    Stream Table
                                  Inlet     Outlet
    Total Molar Flowrate           3440.0     3440.0
    Total Mole Fraction nitrogen  0.095900   0.095900
    Total Mole Fraction methane    0.27040    0.27040
    Total Mole Fraction ethane     0.36030    0.36030
    Total Mole Fraction nbutane    0.27340    0.27340
    Temperature                     287.00     369.91
    Pressure                   3.5000e+05 1.5500e+06
==========================================================================================
```

Figure A.1: Compressor report

```
==========================================================================================
Unit : fs.Heater102                                                          Time: 0.0
------------------------------------------------------------------------------------------
    Unit Performance

    Variables:

    Key       : Value       : Fixed : Bounds
    Heat Duty : -3.0136e+07 : False : (None, None)

------------------------------------------------------------------------------------------
    Stream Table
                                  Inlet     Outlet
    Total Molar Flowrate           3440.0     3440.0
    Total Mole Fraction nitrogen  0.095900   0.095900
    Total Mole Fraction methane    0.27040    0.27040
    Total Mole Fraction ethane     0.36030    0.36030
    Total Mole Fraction nbutane    0.27340    0.27340
    Temperature                     369.91     293.00
    Pressure                   1.5500e+06 1.5500e+06
==========================================================================================
```

Figure A.2: Heater/Cooler report

```
==========================================================================================
Unit : fs.Valve101                                                           Time: 0.0
------------------------------------------------------------------------------------------
    Unit Performance

    Variables:

    Key             : Value       : Fixed : Bounds
    Mechanical Work : -5.3030e-21 : False : (None, None)
    Pressure Change : -2.8500e+06 :  True : (None, None)
     Pressure Ratio :    0.050000 : False : (None, None)

------------------------------------------------------------------------------------------
    Stream Table
                                  Inlet     Outlet
    Total Molar Flowrate           1000.0     1000.0
    Total Mole Fraction nitrogen 0.0037000  0.0037000
    Total Mole Fraction methane    0.95890    0.95890
    Total Mole Fraction ethane    0.029600   0.029600
    Total Mole Fraction propane  0.0072000  0.0072000
    Total Mole Fraction nbutane 0.00060000 0.00060000
    Temperature                     112.00     113.25
    Pressure                   3.0000e+06 1.5000e+05
==========================================================================================
```

Figure A.3: Valve101 report

```
==========================================================================================
Unit : fs.Valve102                                                           Time: 0.0
------------------------------------------------------------------------------------------
    Unit Performance

    Variables:

    Key             : Value       : Fixed : Bounds
    Mechanical Work : -2.0418e-21 : False : (None, None)
    Pressure Change : -1.2000e+06 :  True : (None, None)
     Pressure Ratio :     0.22581 : False : (None, None)

------------------------------------------------------------------------------------------
    Stream Table
                                    Inlet     Outlet
    Total Molar Flowrate             3440.0     3440.0
    Total Mole Fraction nitrogen   0.095900   0.095900
    Total Mole Fraction methane     0.27040    0.27040
    Total Mole Fraction ethane      0.36030    0.36030
    Total Mole Fraction nbutane     0.27340    0.27340
    Temperature                      112.00     111.89
    Pressure                     1.5500e+06 3.5000e+05
==========================================================================================
```

Figure A.4: Valve101 report

```
==========================================================================================
Unit : fs.Splitter102                                                        Time: 0.0
------------------------------------------------------------------------------------------
    Unit Performance

    Variables:

    Key                            : Value   : Fixed : Bounds
    Split Fraction [('outlet_2',)] : 0.95021 : False : (None, None)

------------------------------------------------------------------------------------------
    Stream Table
                                    Inlet
    Pressure                    3.5000e+05
    Temperature                     111.89
    Total Molar Flowrate             3440.0
    Total Mole Fraction ethane      0.36030
    Total Mole Fraction methane     0.27040
    Total Mole Fraction nbutane     0.27340
    Total Mole Fraction nitrogen   0.095900
==========================================================================================
```

Figure A.5: Splitter report

```
==========================================================================================
Unit : fs.Mixer101                                                           Time: 0.0
------------------------------------------------------------------------------------------
    Stream Table
                                   inlet_1    inlet_2    Outlet
    Total Molar Flowrate             3268.2     1711.6     4979.8
    Total Mole Fraction nitrogen   0.095900   0.095900   0.095900
    Total Mole Fraction methane     0.27040    0.27040    0.27040
    Total Mole Fraction ethane      0.36030    0.36030    0.36030
    Total Mole Fraction nbutane     0.27340    0.27340    0.27340
    Temperature                      265.14     189.20     245.82
    Pressure                    3.5000e+05 3.5000e+05 3.5000e+05
==========================================================================================
```

Figure A.6: Valve101 report

```
==========================================================================================
Unit : fs.HE101                                                            Time: 0.0
------------------------------------------------------------------------------------------
    Unit Performance

    Variables:

    Key            : Value      : Fixed : Bounds
           HX Area :     944.08 : False : (0, None)
    HX Coefficient :     944.08 : False : (0, None)
         Heat Duty : 1.3492e+07 : False : (None, None)

    Expressions:

    Key             : Value
    Delta T Driving :  15.137
         Delta T In :  103.80
        Delta T Out : 0.11000

------------------------------------------------------------------------------------------
    Stream Table
                                Hot Inlet  Hot Outlet Cold Inlet Cold Outlet
    Total Molar Flowrate           1000.0     1000.0     1711.6      1711.6
    Total Mole Fraction nitrogen 0.0037000  0.0037000  0.095900    0.095900
    Total Mole Fraction methane    0.95890    0.95890    0.27040     0.27040
    Total Mole Fraction ethane    0.029600   0.029600    0.36030     0.36030
    Total Mole Fraction propane  0.0072000  0.0072000          -           -
    Total Mole Fraction nbutane 0.00060000 0.00060000    0.27340     0.27340
    Temperature                    293.00     112.00     111.89      189.20
    Pressure                   3.0000e+06 3.0000e+06 3.5000e+05  3.5000e+05
==========================================================================================
```

Figure A.7: HE101 report

```
==========================================================================================
Unit : fs.HE102                                                            Time: 0.0
------------------------------------------------------------------------------------------
    Unit Performance

    Variables:

    Key            : Value      : Fixed : Bounds
           HX Area :     3687.6 : False : (0, None)
    HX Coefficient :     3687.6 : False : (0, None)
         Heat Duty : 6.8226e+07 : False : (None, None)

    Expressions:

    Key             : Value
    Delta T Driving :  5.0173
         Delta T In :  27.864
        Delta T Out : 0.11033

------------------------------------------------------------------------------------------
    Stream Table
                                Hot Inlet  Hot Outlet  Cold Inlet  Cold Outlet
    Total Molar Flowrate           3440.0     3440.0      3268.2      3268.2
    Total Mole Fraction nitrogen  0.095900   0.095900    0.095900    0.095900
    Total Mole Fraction methane    0.27040    0.27040     0.27040     0.27040
    Total Mole Fraction ethane     0.36030    0.36030     0.36030     0.36030
    Total Mole Fraction nbutane    0.27340    0.27340     0.27340     0.27340
    Temperature                    293.00     112.00      111.89      265.14
    Pressure                   1.5500e+06 1.5500e+06  3.5000e+05  3.5000e+05
==========================================================================================
```

Figure A.8: HE102 report

Appendix B

Overflow error

```
---------------------------------------------------------------------
----
OverflowError                             Traceback (most recent call l
ast)
~\AppData\Local\Temp/ipykernel_19748/845197630.py in <module>
     39 def function(unit):
     40         unit.initialize(outlvl=idaeslog.DEBUG)
---> 41 seq.run(m, function)
     42

~\Anaconda3\envs\my-idaes-env\lib\site-packages\pyomo\network\decomposi
tion.py in run(self, model, function)
    284
    285         try:
--> 286             return self._run_impl(model, function)
    287         finally:
    288             # Cleanup

~\Anaconda3\envs\my-idaes-env\lib\site-packages\pyomo\network\decomposi
tion.py in _run_impl(self, model, function)
    343                     kwds["accel_min"] = self.options["accel_min
"]
    344                     kwds["accel_max"] = self.options["accel_max
"]
--> 345                     self.solve_tear_wegstein(**kwds)
    346
    347                 else:

~\Anaconda3\envs\my-idaes-env\lib\site-packages\pyomo\network\foqus_gra
ph.py in solve_tear_wegstein(self, G, order, function, tears, outEdges,
iterLim, tol, tol_type, report_diffs, accel_min, accel_max)
    211
    212             logger.info("Running Wegstein iteration %s" % iterc
ount)
--> 213             self.run_order(G, order, function, ignore)
    214
    215             gofx = self.generate_gofx(G, tears)

~\Anaconda3\envs\my-idaes-env\lib\site-packages\pyomo\network\decomposi
tion.py in run_order(self, G, order, function, ignore, use_guesses)
    391                     self.load_values(port, default, fixed_ins,
use_guesses)
    392
--> 393                 function(unit)
    394
    395                 # free the inputs that were not already fixed

~\AppData\Local\Temp/ipykernel_19748/845197630.py in function(unit)
     38
     39 def function(unit):
---> 40         unit.initialize(outlvl=idaeslog.DEBUG)
     41 seq.run(m, function)
     42

~\Anaconda3\envs\my-idaes-env\lib\site-packages\idaes\generic_models\un
it_models\heat_exchanger.py in initialize(self, state_args_1, state_arg
s_2, outlvl, solver, optarg, duty)
```

```
    552         init_log.info_high("Initialization Step 1a (hot side) C
omplete.")
    553
--> 554         flags2 = cold_side.initialize(
    555             outlvl=outlvl, optarg=optarg, solver=solver, state_
args=state_args_2
    556         )

~\Anaconda3\envs\my-idaes-env\lib\site-packages\idaes\core\control_volu
me0d.py in initialize(blk, state_args, outlvl, optarg, solver, hold_sta
te)
   1365
   1366         # Initialize state blocks
-> 1367         in_flags = blk.properties_in.initialize(
   1368             outlvl=outlvl,
   1369             optarg=optarg,

~\Anaconda3\envs\my-idaes-env\lib\site-packages\idaes\generic_models\pr
operties\core\generic\generic_property.py in initialize(blk, state_args
, state_vars_fixed, hold_state, outlvl, solver, optarg)
   1340                         c.activate()
   1341                         for p, j in blk[k].params._phase_compon
ent_set:
-> 1342                             calculate_variable_from_constraint(
   1343                                 blk[k].log_mole_frac_phase_comp
[p,j],
   1344                                 blk[k].log_mole_frac_phase_comp
_eqn[p,j])

~\Anaconda3\envs\my-idaes-env\lib\site-packages\pyomo\util\calc_var_val
ue.py in calculate_variable_from_constraint(variable, constraint, eps,
iterlim, linesearch, alpha_min)
    143         if slope:
    144             variable.set_value(-intercept/slope, skip_validatio
n=True)
--> 145             body_val = value(body, exception=False)
    146             if body_val is not None and abs(body_val - upper) <
eps:
    147                 # Re-set the variable value to trigger any warn
ings WRT

pyomo\core\expr\numvalue.pyx in pyomo.core.expr.numvalue.value()

pyomo\core\expr\numeric_expr.pyx in pyomo.core.expr.numeric_expr.Expres
sionBase.__call__()

~\Anaconda3\envs\my-idaes-env\lib\site-packages\pyomo\core\expr\visitor
.py in evaluate_expression(exp, exception, constant)
    890         visitor = _EvaluationVisitor(exception=exception)
    891     try:
--> 892         return visitor.dfs_postorder_stack(exp)
    893
    894     except ( TemplateExpressionError, ValueError, TypeError,

~\Anaconda3\envs\my-idaes-env\lib\site-packages\pyomo\core\expr\visitor
.py in dfs_postorder_stack(self, node)
    589             # Process the current node
    590             #
```

```
--> 591                 ans = self.visit(_obj, _result)
    592                 if _stack:
    593                     #

~\Anaconda3\envs\my-idaes-env\lib\site-packages\pyomo\core\expr\visitor
.py in visit(self, node, values)
    785     def visit(self, node, values):
    786         """ Visit nodes that have been expanded """
--> 787         return node._apply_operation(values)
    788
    789     def visiting_potential_leaf(self, node):

pyomo\core\expr\numeric_expr.pyx in pyomo.core.expr.numeric_expr.UnaryF
unctionExpression._apply_operation()

OverflowError: math range error
```

Appendix C

Evaluation and scaling error

```
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: This program contains Ipo
pt, a library for large-scale nonlinear optimization.
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: Ipopt is released as open
source code under the Eclipse Public License (EPL).
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: For more information visi
t http://projects.coin-or.org/Ipopt
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: This version of Ipopt was
compiled from source code available at
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: https://github.com/IDAES/
Ipopt as part of the Institute for the Design of
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: Advanced Energy Systems P
rocess Systems Engineering Framework (IDAES PSE
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: Framework) Copyright (c)
2018-2019. See https://github.com/IDAES/idaes-pse.
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: This version of Ipopt was
compiled using HSL, a collection of Fortran codes
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: for large-scale scientifi
c computation.  All technical papers, sales and
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: publicity material result
ing from use of the HSL codes within IPOPT must
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: contain the following ack
nowledgement:
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: HSL, a collection of Fort
ran codes for large-scale scientific
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: computation. See http://w
ww.hsl.rl.ac.uk.
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: ************************
************************************************
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: This is Ipopt version 3.1
3.2, running with linear solver ma27.
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: Number of nonzeros in equ
ality constraint Jacobian...:     1127
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: Number of nonzeros in ine
quality constraint Jacobian.:        0
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: Number of nonzeros in Lag
rangian Hessian.............:      572
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: Total number of variables
............................:      214
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: variables with only lower
bounds:       45
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: variables with lower and
upper bounds:      121
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: variables with only upper
bounds:       36
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: Total number of equality
constraints.................:      213
```

```
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: Total number of inequalit
y constraints...............:        0
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: inequality constraints wi
th only lower bounds:        0
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: inequality constraints wi
th lower and upper bounds:        0
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: inequality constraints wi
th only upper bounds:        0
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: iter    objective    inf_
pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
2022-06-16 16:22:10 [DEBUG] idaes.solve.fs.HE101: 0  0.0000000e+00 3.28e+08
1.00e+00  -1.0 0.00e+00    -  0.00e+00 0.00e+00   0
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: 1  0.0000000e+00 1.62e+08
2.47e+01  -1.0 3.79e+06    -  6.86e-01 4.95e-01h  2
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: 2  0.0000000e+00 1.13e+08
4.48e+04  -1.0 2.08e+06    -  6.90e-01 2.40e-01h  3
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: 3  0.0000000e+00 9.90e+07
4.44e+04  -1.0 2.58e+06    -  3.13e-01 6.05e-02h  5
```

```
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: 4  0.0000000e+00 1.44e+07
4.37e+06  -1.0 8.92e+02  -4.0 8.32e-01 9.69e-01h  1
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: 5  0.0000000e+00 1.01e+07
4.10e+06  -1.0 9.84e+05    -  1.01e-01 6.19e-02h  5
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: 6  0.0000000e+00 1.01e+07
4.10e+06  -1.0 7.00e+02  -4.5 3.43e-01 1.18e-04h  1
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
```

```
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: 7  0.0000000e+00 8.07e+06
4.07e+06  -1.0 9.25e+05    -  8.51e-02 7.74e-03h  8
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
```

```
2022-06-16 16:22:11 [DEBUG] idaes.solve.fs.HE101: 8  0.0000000e+00 6.27e+06
4.06e+06  -1.0 9.98e+05    -  1.94e-05 9.67e-04f 11
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
```

```
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Error in an AMPL evaluati
on. Run with "halt_on_ampl_error yes" to see details.
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: Warning: Cutting back alp
ha due to evaluation error
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: 9  0.0000000e+00 5.74e+06
4.06e+06  -1.0 1.35e+06  -4.1 5.69e-04 4.24e-05h 15
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: iter    objective    inf_
pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
2022-06-16 16:22:12 [DEBUG] idaes.solve.fs.HE101: 10  0.0000000e+00 5.48e+0
6 4.06e+06  -1.0 2.04e+05  -2.7 8.99e-01 9.71e-05h  1
```

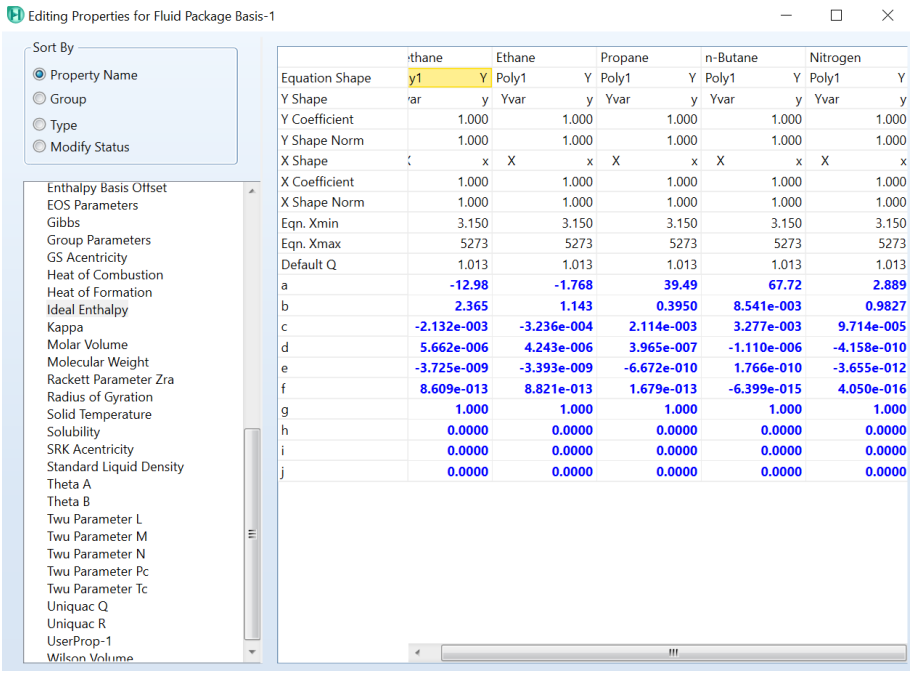
Last 10 iterations

```
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 351r 0.0000000e+00 2.43e+
00 9.05e+02  -7.0 1.68e+04    -  2.10e-01 1.31e-03h  1
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 352r 0.0000000e+00 2.43e+
00 8.95e+02  -7.0 4.87e+01    -  5.10e-04 7.23e-04h  1
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 353r 0.0000000e+00 2.39e+
00 1.42e+03  -7.0 6.07e+00    -  7.43e-01 1.53e-02h  1
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 354r 0.0000000e+00 1.75e+
00 1.04e+03  -7.0 5.98e+00    -  3.60e-01 2.54e-01h  1
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 355r 0.0000000e+00 1.75e+
00 1.04e+03  -7.0 4.49e+00    -  0.00e+00 6.23e-08R  2
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 356r 0.0000000e+00 4.03e-
01 5.31e+02  -7.0 4.78e+00    -  5.51e-01 4.89e-01f  1
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
```

```
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 357r 0.0000000e+00 3.99e-
01 5.26e+02  -7.0 2.66e+00    -  7.69e-03 9.48e-03f  1
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 358r 0.0000000e+00 1.04e-
01 2.98e+02  -7.0 2.30e+00    -  1.02e-01 6.63e-01f  1
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 359r 0.0000000e+00 2.06e-
02 1.63e+02  -7.0 5.77e-01    -  4.94e-01 8.08e-01f  1
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: iter    objective    inf_
pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 360r 0.0000000e+00 3.70e-
03 5.30e+00  -7.0 5.02e-02    -  1.00e+00 9.43e-01f  1
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: Scaling factors are inval
id - setting them all to 1.
2022-06-16 16:23:05 [DEBUG] idaes.solve.fs.HE101: 361r 0.0000000e+00 3.70e-
03 2.41e-05  -7.0 5.61e-03    -  1.00e+00 1.00e+00f  1
```

Appendix D

HYSYS coefficients



	ethane	Ethane	Propane	n-Butane	Nitrogen
Equation Shape	y1 Y	Poly1 Y	Poly1 Y	Poly1 Y	Poly1 Y
Y Shape	ar y	Yvar y	Yvar y	Yvar y	Yvar y
Y Coefficient	1.000	1.000	1.000	1.000	1.000
Y Shape Norm	1.000	1.000	1.000	1.000	1.000
X Shape	x	X x	X x	X x	X x
X Coefficient	1.000	1.000	1.000	1.000	1.000
X Shape Norm	1.000	1.000	1.000	1.000	1.000
Eqn. Xmin	3.150	3.150	3.150	3.150	3.150
Eqn. Xmax	5273	5273	5273	5273	5273
Default Q	1.013	1.013	1.013	1.013	1.013
a	**-12.98**	**-1.768**	**39.49**	**67.72**	**2.889**
b	**2.365**	**1.143**	**0.3950**	**8.541e-003**	**0.9827**
c	**-2.132e-003**	**-3.236e-004**	**2.114e-003**	**3.277e-003**	**9.714e-005**
d	**5.662e-006**	**4.243e-006**	**3.965e-007**	**-1.110e-006**	**-4.158e-010**
e	**-3.725e-009**	**-3.393e-009**	**-6.672e-010**	**1.766e-010**	**-3.655e-012**
f	**8.609e-013**	**8.821e-013**	**1.679e-013**	**-6.399e-015**	**4.050e-016**
g	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
h	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
i	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
j	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**

Figure D.1: Ideal gas enthalpy coefficients in HYSYS

Appendix E

Property package configuration

"""
Hydrocarbon processing phase equilibrium package using Peng-Robinson EoS.

Example property package using the Generic Property Package Framework.
This example shows how to set up a property package to do hydrocarbon
processing phase equilibrium in the generic framework using Peng-Robinson
equation along with methods drawn from the pre-built IDAES property libraries.

The example includes the dictionary named configuration contains parameters
for calculating VLE phase equilibrium and properties for hydrocarbon processing.
"""

# Import Python libraries
import logging

# Import Pyomo units
from pyomo.environ import units as pyunits

Import IDAES cores

```
from idaes.core import LiquidPhase, VaporPhase, Component
from idaes.core.phases import PhaseType as PT
from idaes.generic_models.properties.core.state_definitions import FTPx
from idaes.generic_models.properties.core.eos.ceos import Cubic, CubicType
from idaes.generic_models.properties.core.phase_equil import SmoothVLE
from idaes.generic_models.properties.core.phase_equil.bubble_dew import \
      LogBubbleDew
from idaes.generic_models.properties.core.phase_equil.forms import log_fugacity

from idaes.generic_models.properties.core.pure import Perrys
from idaes.generic_models.properties.core.pure import RPP4
from idaes.generic_models.properties.core.pure import RPP5

# Set up logger
_log = logging.getLogger(__name__)



# ---------------------------------------------------------------------
# Configuration dictionary for a Peng Robinson natural gas system

# Data Sources:
# [1] The Properties of Gases and Liquids (1987)
#     4th edition, Chemical Engineering Series - Robert C. Reid
# [2] Perry's Chemical Engineers' Handbook 7th Ed.
#     Converted to J/mol.K, mol/m^3
# [3] Engineering Toolbox, https://www.engineeringtoolbox.com
#     Retrieved 15th september, 2020
# [4] The Properties of Gases and Liquids (2001)
#     5th edition, Chemical Engineering Series - Robert C. Reid

configuration = {
```

```
# Specifying components
"components": {
    "nitrogen": {"type": Component,
                "elemental_composition": {'N': 2, 'C': 0},
                "dens_mol_liq_comp": Perrys,
                "enth_mol_liq_comp": Perrys,
                "enth_mol_ig_comp": RPP4,
                "entr_mol_ig_comp": RPP4,
                "pressure_sat_comp": RPP5,
                "phase_equilibrium_form": {("Vap", "Liq"): log_fugacity},
                "parameter_data": {
                   "mw": (28.0135E-3, pyunits.kg/pyunits.mol),  # [1]
                   "pressure_crit": (34e5, pyunits.Pa),  # [1]
                   "temperature_crit": (126.2, pyunits.K),  # [1]
                   "omega": 0.04,  # [1]
                   "dens_mol_liq_comp_coeff": {
                     "1": (3.2091, pyunits.kmol*pyunits.m**-3),  # [2] pg. 2-98
                     "2": (0.2861, None),
                     "3": (126.2, pyunits.K),
                     "4": (0.2966, None)},
                   "cp_mol_ig_comp_coeff": {
                      "A": (3.115E1,
                           pyunits.J/pyunits.mol/pyunits.K),  # [1]
                      "B": (-1.357E-2,
                           pyunits.J/pyunits.mol/pyunits.K**2),
                      "C": (2.680E-5,
                           pyunits.J/pyunits.mol/pyunits.K**3),
                      "D": (-1.168E-8,
                           pyunits.J/pyunits.mol/pyunits.K**4)},
                   "cp_mol_liq_comp_coeff": {
                     "1": (2.8197e5, pyunits.J*pyunits.kmol**-1*pyunits.K**-1),  # [2]
```

```
                "2": (-1.2281e4, pyunits.J*pyunits.kmol**-1*pyunits.K**-2),
                "3": (2.4800e2, pyunits.J*pyunits.kmol**-1*pyunits.K**-3),
                "4": (-2.2182, pyunits.J*pyunits.kmol**-1*pyunits.K**-4),
                "5": (7.4902e-3, pyunits.J*pyunits.kmol**-1*pyunits.K**-5)},
            "enth_mol_form_liq_comp_ref": (
              0, pyunits.J/pyunits.mol), # [3]
            "enth_mol_form_vap_comp_ref": (
               0.0, pyunits.J/pyunits.mol),  # [3]
            "entr_mol_form_vap_comp_ref": (
               191.6, pyunits.J/pyunits.mol/pyunits.K),  # [3]
            "pressure_sat_comp_coeff": {
               "A": (3.61947, None),  # [4]
               "B": (255.68, pyunits.K),
               "C": (266.55, pyunits.K)}}},

"methane": {"type": Component,
        "elemental_composition": {'H': 4, 'C': 1},
        "dens_mol_liq_comp": Perrys,
        "enth_mol_liq_comp": Perrys,
        "enth_mol_ig_comp": RPP4,
        "entr_mol_ig_comp": RPP4,
        "pressure_sat_comp": RPP5,
        "phase_equilibrium_form": {("Vap", "Liq"): log_fugacity},
        "parameter_data": {
          "mw": (16.043E-3, pyunits.kg/pyunits.mol),  # [1]
          "pressure_crit": (46e5, pyunits.Pa),  # [1]
          "temperature_crit": (190.4, pyunits.K),  # [1]
          "omega": 0.011,
          "dens_mol_liq_comp_coeff": {
            "1": (2.9214, pyunits.kmol*pyunits.m**-3),  # [2] pg. 2-98
            "2": (0.28976, None),
```

```
            "3": (190.56, pyunits.K),
            "4": (0.28881, None)},
        "cp_mol_ig_comp_coeff": {
            "A": (1.925e1, pyunits.J/pyunits.mol/pyunits.K), # [1]
            "B": (5.213e-2, pyunits.J/pyunits.mol/pyunits.K**2),
            "C": (1.197e-5, pyunits.J/pyunits.mol/pyunits.K**3),
            "D": (-1.132e-8, pyunits.J/pyunits.mol/pyunits.K**4)},
        "cp_mol_liq_comp_coeff": {
            "1": (6.5708e1, pyunits.J*pyunits.kmol**-1*pyunits.K**-1),  # [2]
            "2": (3.8883e4, pyunits.J*pyunits.kmol**-1*pyunits.K**-2),
            "3": (-2.5795e2, pyunits.J*pyunits.kmol**-1*pyunits.K**-3),
            "4": (6.1407e2, pyunits.J*pyunits.kmol**-1*pyunits.K**-4),
            "5": (0, pyunits.J*pyunits.kmol**-1*pyunits.K**-5)},
        "enth_mol_form_liq_comp_ref": (
            0, pyunits.J/pyunits.mol), # [3]
        "entr_mol_form_vap_comp_ref": (
            186, pyunits.J/pyunits.mol/pyunits.K), # [3]
        "enth_mol_form_vap_comp_ref": (
            -75000, pyunits.J/pyunits.mol), # [3]
        "pressure_sat_comp_coeff": {
            "A": (3.76870, None),  # [4]
            "B": (395.7440, pyunits.K),
            "C": (266.681, pyunits.K)}}},

"ethane": {"type": Component,
        "elemental_composition": {'H': 6, 'C': 2},
        "dens_mol_liq_comp": Perrys,
        "enth_mol_liq_comp": Perrys,
        "enth_mol_ig_comp": RPP4,
        "entr_mol_ig_comp": RPP4,
        "pressure_sat_comp": RPP5,
```

```
"phase_equilibrium_form": {("Vap", "Liq"): log_fugacity},
"parameter_data": {
   "mw": (30.070E-3, pyunits.kg/pyunits.mol),  # [1]
   "pressure_crit": (48.8e5, pyunits.Pa),  # [1]
   "temperature_crit": (305.4, pyunits.K),  # [1]
   "omega": 0.099,
   "dens_mol_liq_comp_coeff": {
      "1": (1.9122, pyunits.kmol*pyunits.m**-3),  # [2] pg. 2-98
      "2": (0.27937, None),
      "3": (305.32, pyunits.K),
      "4": (0.29187, None)},
   "cp_mol_ig_comp_coeff": {
      "A": (5.409e0, pyunits.J/pyunits.mol/pyunits.K), # [1]
      "B": (1.781e-1, pyunits.J/pyunits.mol/pyunits.K**2),
      "C": (-6.938e-5, pyunits.J/pyunits.mol/pyunits.K**3),
      "D": (8.713e-9, pyunits.J/pyunits.mol/pyunits.K**4)},
   "cp_mol_liq_comp_coeff": {
      "1": (4.4009e1, pyunits.J*pyunits.kmol**-1*pyunits.K**-1),  # [2]
      "2": (8.9718e4, pyunits.J*pyunits.kmol**-1*pyunits.K**-2),
      "3": (9.1877e2, pyunits.J*pyunits.kmol**-1*pyunits.K**-3),
      "4": (-1.886e3, pyunits.J*pyunits.kmol**-1*pyunits.K**-4),
      "5": (0, pyunits.J*pyunits.kmol**-1*pyunits.K**-5)},
   "enth_mol_form_liq_comp_ref": (
      0, pyunits.J/pyunits.mol),  # [3]
   "entr_mol_form_vap_comp_ref": (
      229.2, pyunits.J/pyunits.mol/pyunits.K), # [3]
   "enth_mol_form_vap_comp_ref": (
      -84000, pyunits.J/pyunits.mol),  # [3]
   "pressure_sat_comp_coeff": {"A": (3.95405, None),  # [4]
                               "B": (663.720, pyunits.K),
                               "C": (256.681, pyunits.K)}}},
```

```
"propane": {"type": Component,
        "elemental_composition": {'H': 8, 'C': 3},
        "dens_mol_liq_comp": Perrys,
        "enth_mol_liq_comp": Perrys,
        "enth_mol_ig_comp": RPP4,
        "entr_mol_ig_comp": RPP4,
        "pressure_sat_comp": RPP5,
        "phase_equilibrium_form": {("Vap", "Liq"): log_fugacity},
        "parameter_data": {
          "mw": (44.094E-3, pyunits.kg/pyunits.mol),  # [1]
          "pressure_crit": (42.5e5, pyunits.Pa),  # [1]
          "temperature_crit": (369.8, pyunits.K),  # [1]
          "omega": 0.153, # [1]
          "dens_mol_liq_comp_coeff": {
            "1": (1.3757, pyunits.kmol*pyunits.m**-3),  # [2] pg. 2-98
            "2": (0.27453, None),
            "3": (369.83, pyunits.K),
            "4": (0.29359, None)},
          "cp_mol_ig_comp_coeff": {
            "A": (-4.224e0, pyunits.J/pyunits.mol/pyunits.K), # [1]
            "B": (3.063e-1, pyunits.J/pyunits.mol/pyunits.K**2),
            "C": (-1.586e-4, pyunits.J/pyunits.mol/pyunits.K**3),
            "D": (3.215e-8, pyunits.J/pyunits.mol/pyunits.K**4)},
          "cp_mol_liq_comp_coeff": {
            "1": (6.2983e1, pyunits.J*pyunits.kmol**-1*pyunits.K**-1),  # [2]
            "2": (1.1363e5, pyunits.J*pyunits.kmol**-1*pyunits.K**-2),
            "3": (6.3321e2, pyunits.J*pyunits.kmol**-1*pyunits.K**-3),
            "4": (-8.7346e2, pyunits.J*pyunits.kmol**-1*pyunits.K**-4),
            "5": (0, pyunits.J*pyunits.kmol**-1*pyunits.K**-5)},
          "enth_mol_form_liq_comp_ref": (
```

```
            0.0, pyunits.J/pyunits.mol),  # [3]
        "entr_mol_form_vap_comp_ref": (
            270.3, pyunits.J/pyunits.mol/pyunits.K), # [3]
        "enth_mol_form_vap_comp_ref": (
            -103800, pyunits.J/pyunits.mol),  # [3]
        "pressure_sat_comp_coeff": {"A": (3.92828, None),  # [4]
                                    "B": (803.9970, pyunits.K),
                                    "C": (247.040, pyunits.K)}}},
"nbutane": {"type": Component,
        "elemental_composition": {'H': 10, 'C': 4},
        "dens_mol_liq_comp": Perrys,
        "enth_mol_liq_comp": Perrys,
        "enth_mol_ig_comp": RPP4,
        "entr_mol_ig_comp": RPP4,
        "pressure_sat_comp": RPP5,
        "phase_equilibrium_form": {("Vap", "Liq"): log_fugacity},
        "parameter_data": {
            "mw": (58.124E-3, pyunits.kg/pyunits.mol),  # [1]
            "pressure_crit": (38.0e5, pyunits.Pa),  # [1]
            "temperature_crit": (425.2, pyunits.K),  # [1]
            "omega": 0.199,
            "dens_mol_liq_comp_coeff": {
                "1": (1.0677, pyunits.kmol*pyunits.m**-3),  # [2] pg. 2-98
                "2": (0.27188, None),
                "3": (425.12, pyunits.K),
                "4": (0.28688, None)},
            "cp_mol_ig_comp_coeff": {
                "A": (9.487e0, pyunits.J/pyunits.mol/pyunits.K), # [1]
                "B": (3.313e-1, pyunits.J/pyunits.mol/pyunits.K**2),
                "C": (-1.108e-4, pyunits.J/pyunits.mol/pyunits.K**3),
                "D": (-2.822e-9, pyunits.J/pyunits.mol/pyunits.K**4)},
```

```
                "cp_mol_liq_comp_coeff": {
                   "1": (6.473e1, pyunits.J*pyunits.kmol**-1*pyunits.K**-1),  # [2]
                   "2": (1.6184e5, pyunits.J*pyunits.kmol**-1*pyunits.K**-2),
                   "3": (9.8341e2, pyunits.J*pyunits.kmol**-1*pyunits.K**-3),
                   "4": (-1.4315e3, pyunits.J*pyunits.kmol**-1*pyunits.K**-4),
                   "5": (0, pyunits.J*pyunits.kmol**-1*pyunits.K**-5)},
                "enth_mol_form_liq_comp_ref": (
                   0, pyunits.J/pyunits.mol),  # [3]
                "entr_mol_form_vap_comp_ref": (
                   310, pyunits.J/pyunits.mol/pyunits.K), # [3]
                "enth_mol_form_vap_comp_ref": (
                   -125700, pyunits.J/pyunits.mol),  # [3]
                "pressure_sat_comp_coeff": {"A": (3.93266, None),  # [4]
                                            "B": (935.7730, pyunits.K),
                                            "C": (238.789, pyunits.K)}}}},



# Specifying phases
"phases":  {"Liq": {"type": LiquidPhase,
                "equation_of_state": Cubic,
                "equation_of_state_options": {
                   "type": CubicType.PR}},
           "Vap": {"type": VaporPhase,
                "equation_of_state": Cubic,
                "equation_of_state_options": {
                   "type": CubicType.PR}}},


# Set base units of measurement
"base_units": {"time": pyunits.s,
               "length": pyunits.m,
               "mass": pyunits.kg,
```

```
            "amount": pyunits.mol,
            "temperature": pyunits.K},

# Specifying state definition
"state_definition": FTPx,
"state_bounds": {"flow_mol": (0, 3000, 7000, pyunits.mol/pyunits.s),
              "temperature": (100, 300, 500, pyunits.K),
              "pressure": (1e5, 20e5, 85e5, pyunits.Pa)},
"pressure_ref": (101325, pyunits.Pa),
"temperature_ref": (298.15, pyunits.K),



# Defining phase equilibria
"phases_in_equilibrium": [("Vap", "Liq")],
"phase_equilibrium_state": {("Vap", "Liq"): SmoothVLE},
"bubble_dew_method": LogBubbleDew,
"parameter_data": {"PR_kappa": {("nitrogen", "nitrogen"): 0.000,
                                ("nitrogen", "methane"): 0.000,
                                ("nitrogen", "ethane"): 0.000,
                                ("nitrogen", "propane"): 0.000,
                                ("nitrogen", "nbutane"): 0.000,
                                ("methane", "nitrogen"): 0.000,
                                ("methane", "methane"): 0.000,
                                ("methane", "ethane"): 0.000,
                                ("methane", "propane"): 0.000,
                                ("methane", "nbutane"): 0.000,
                                ("ethane", "nitrogen"): 0.000,
                                ("ethane", "methane"): 0.000,
                                ("ethane", "ethane"): 0.000,
                                ("ethane", "propane"): 0.000,
                                ("ethane", "nbutane"): 0.000,
```

("propane", "nitrogen"): 0.000,

("propane", "methane"): 0.000,

("propane", "ethane"): 0.000,

("propane", "propane"): 0.000,

("propane", "nbutane"): 0.000,

("nbutane", "nitrogen"): 0.000,

("nbutane", "methane"): 0.000,

("nbutane", "ethane"): 0.000,

("nbutane", "propane"): 0.000,

("nbutane", "nbutane"): 0.000}}}

Appendix F

IDAES python script

In [ ]:
```
from pyomo.environ import (Constraint,
                           Var,
                           ConcreteModel,
                           Expression,
                           Objective,
                           SolverFactory,
                           TransformationFactory,
                           value)
from pyomo.network import Arc, SequentialDecomposition
from idaes.core import FlowsheetBlock
from idaes.generic_models.unit_models import (Compressor,
                                      Mixer, MomentumMixingType,
                                      Separator as Splitter,
                                      Heater,
                                      HeatExchanger,
                                      Valve,
                                      Feed,
                                           PressureChanger)
from idaes.generic_models.unit_models.pressure_changer import ThermodynamicAs
from idaes.core.util.model_statistics import degrees_of_freedom
from idaes.generic_models.unit_models.heat_exchanger import delta_temperature

# Import idaes logger to set output levels
import idaes.logger as idaeslog
from idaes.generic_models.properties.core.examples.HC_PR2 import configuratic
from idaes.generic_models.properties.core.examples.HC_PR1 import configuratic
from idaes.generic_models.properties.core.generic.generic_property import (
        GenericParameterBlock)
```

In [ ]:
```
m = ConcreteModel()
m.fs = FlowsheetBlock(default={"dynamic": False})
```

In [ ]:
```
m.fs.properties = GenericParameterBlock(default=configuration1)
m.fs.propertiesNG = GenericParameterBlock(default=configuration)
```

In [ ]:

```
m.fs.Compressor102 = Compressor(default={"dynamic": False,
             "property_package": m.fs.properties})
             #"has_phase_equilibrium": True})
m.fs.Valve102 = PressureChanger(default={"property_package": m.fs.properties,
                                         "compressor": False,
                                         "thermodynamic_assumption": Thermody
m.fs.Valve101 = PressureChanger(default={"property_package": m.fs.propertiesN
                                         "compressor": False,
                                         "thermodynamic_assumption": Thermody

m.fs.Heater102 = Heater(default={"property_package": m.fs.properties,
                                 "has_pressure_change": False})

m.fs.HE102 = HeatExchanger(default={
        "delta_temperature_callback":delta_temperature_lmtd_callback,
        "shell":{"property_package": m.fs.properties,
                 "has_phase_eqillibrium:": True,
                 "has_pressure_change": False,
                 "material_balance_type": "useDefault"},
        "tube":{"property_package": m.fs.properties,
                "has_phase_eqillibrium:": True,
                "has_pressure_change": False,
                "material_balance_type": "useDefault"}})

m.fs.HE101 = HeatExchanger(default={
        "delta_temperature_callback":delta_temperature_lmtd_callback,
        "shell":{"property_package": m.fs.propertiesNG,
                 "has_phase_eqillibrium:": True,
                 "has_pressure_change": False,
                 "material_balance_type": "useDefault"},
        "tube":{"property_package": m.fs.properties,
                "has_phase_eqillibrium:": True,
                "has_pressure_change": False,
                "material_balance_type": "useDefault"}})


m.fs.Splitter102 = Splitter(default={"property_package": m.fs.properties,
                                     "num_outlets": 2,
                                     "ideal_separation": False})

m.fs.Mixer101 = Mixer(default={"property_package": m.fs.properties,
                               "num_inlets":2})

m.fs.s01 = Arc(source=m.fs.Compressor102.outlet, destination=m.fs.Heater102.i
m.fs.s02 = Arc(source=m.fs.Heater102.outlet, destination=m.fs.HE102.shell_inl
m.fs.s03 = Arc(source=m.fs.HE102.shell_outlet, destination=m.fs.Valve102.inle
m.fs.s04 = Arc(source=m.fs.Valve102.outlet, destination=m.fs.Splitter102.inle
m.fs.s05 = Arc(source=m.fs.Splitter102.outlet_2, destination=m.fs.HE102.tube_
m.fs.s06 = Arc(source=m.fs.HE102.tube_outlet, destination=m.fs.Mixer101.inlet
m.fs.s07 = Arc(source=m.fs.HE101.shell_outlet, destination=m.fs.Valve101.inle
m.fs.s08 = Arc(source=m.fs.HE101.tube_outlet, destination=m.fs.Mixer101.inlet


TransformationFactory("network.expand_arcs").apply_to(m)
```

In [ ]:
```
DOF_initial = degrees_of_freedom(m)
print("The initial DOF is {0}".format(DOF_initial))
```

In [ ]:
```
#Compressor101
m.fs.Compressor102.inlet.flow_mol[0].fix(3440)
m.fs.Compressor102.inlet.pressure[0].fix(3.5e5)
m.fs.Compressor102.outlet.pressure[0].fix(15.5e5)
m.fs.Compressor102.inlet.temperature[0].fix(287)
m.fs.Compressor102.inlet.mole_frac_comp[0,"nitrogen"].fix(0.0959)
m.fs.Compressor102.inlet.mole_frac_comp[0,"methane"].fix(0.2704)
m.fs.Compressor102.inlet.mole_frac_comp[0,"ethane"].fix(0.3603)
m.fs.Compressor102.inlet.mole_frac_comp[0,"nbutane"].fix(0.2734)
m.fs.Compressor102.efficiency_isentropic[0].fix(0.8)
m.fs.Valve102.deltaP[0].fix(-12e5)
m.fs.Valve101.deltaP[0].fix(-28.5e5)
m.fs.Heater102.outlet.temperature[0].fix(293)
#m.fs.HE102.overall_heat_transfer_coefficient[0].fix(135000) #W/m2/K


m.fs.HE101.shell_inlet.temperature.fix(293)
m.fs.HE101.shell_inlet.pressure.fix(30e5)
m.fs.HE101.shell_inlet.flow_mol.fix(1000)
m.fs.HE101.shell_inlet.mole_frac_comp[0,"nitrogen"].fix(0.0037)
m.fs.HE101.shell_inlet.mole_frac_comp[0,"methane"].fix(0.9589)
m.fs.HE101.shell_inlet.mole_frac_comp[0,"ethane"].fix(0.0296)
m.fs.HE101.shell_inlet.mole_frac_comp[0,"nbutane"].fix(0.0006)
m.fs.HE101.shell_inlet.mole_frac_comp[0,"propane"].fix(0.0072)
#m.fs.HE101.overall_heat_transfer_coefficient[0].fix(135000)

m.fs.HE101.tube_inlet.flow_mol.fix(1000)
m.fs.HE101.tube_inlet.pressure.fix(3.5e5)
m.fs.HE101.tube_inlet.temperature.fix(106)
m.fs.HE101.tube_inlet.mole_frac_comp[0,"nitrogen"].fix(0.0959)
m.fs.HE101.tube_inlet.mole_frac_comp[0,"methane"].fix(0.2704)
m.fs.HE101.tube_inlet.mole_frac_comp[0,"ethane"].fix(0.3603)
m.fs.HE101.tube_inlet.mole_frac_comp[0,"nbutane"].fix(0.2734)
```

In [ ]:
```
DOF_final = degrees_of_freedom(m)
print("The final DOF is {0}".format(DOF_final))
assert DOF_final == 5
```

In [ ]:

```
seq = SequentialDecomposition()
seq.options.select_tear_method = "heuristic"
seq.options.tear_method = "Wegstein"
seq.options.iterLim = 2

# Using the SD tool
G = seq.create_graph(m)
heuristic_tear_set = seq.tear_set_arcs(G, method="heuristic")
order = seq.calculation_order(G)

for o in heuristic_tear_set:
    print(o.name)
print("Order of initialization")
for o in order:
    print(o[0].name)

tear_guesses = {
        "flow_mol_phase_comp": {
                (0, "Vap", "nitrogen"): 1,
                (0, "Vap", "methane"): 1,
                (0, "Vap", "ethane"): 1,
                (0, "Vap", "nbutane"): 1,
                (0, "Liq", "nitrogen"): 1e-5,
                (0, "Liq", "methane"): 1e-5,
                (0, "Liq", "ethane"): 1e-5,
                (0, "Liq", "nbutane"): 1e-5},

        "temperature": {0: 280},
        "pressure": {0: 1550000},
        "flow_mol": {0: 3440}}

# Pass the tear_guess to the SD tool
seq.set_guesses_for(m.fs.HE102.shell_outlet, tear_guesses)
```

In [ ]:

```
# Create the solver object
solver = SolverFactory('ipopt')
solver.options = {'tol': 1e-6, 'max_iter': 2000}
```

In [ ]:
```
def function(unit):
        unit.initialize(outlvl=idaeslog.DEBUG)
# Import python path
import os

# Import idaes model serializer to store initialized model
from idaes.core.util import model_serializer as ms

if not os.path.exists("HM_init_testing1.json.gz"):
    seq.run(m, function)

    # Solve the simulation using ipopt
    # Note: If the degrees of freedom = 0, we have a square problem
    solve_status = solver.solve(m, tee=True)

    ms.to_json(m, fname="HM_init_testing1.json.gz")
else:
    ms.from_json(m, fname="HM_init_testing1.json.gz")
```

In [ ]:
```
result = solver.solve(m, tee=True)
```

In [ ]:
```
ms.to_json(m, fname="HM_init_testing1.json.gz")
```

In [ ]:
```
m.fs.Compressor102.report()
m.fs.Heater102.report()
m.fs.HE102.report()
m.fs.Valve102.report()
m.fs.Splitter102.report()
m.fs.HE101.report()
m.fs.Valve101.report()
m.fs.Mixer101.report()
```

In [ ]:
```
from pyomo.opt import TerminationCondition, SolverStatus
import pytest

# Check if termination condition is optimal
assert result.solver.termination_condition == TerminationCondition.optimal
assert result.solver.status == SolverStatus.ok

assert value(m.fs.HE101.shell.properties_out[0].temperature) == pytest.approx
assert value(m.fs.HE101.tube.properties_out[0].temperature) == pytest.approx(
```

In [ ]:
```
m.fs.HE101.tube_inlet.flow_mol.unfix()
m.fs.HE101.overall_heat_transfer_coefficient[0].unfix()
m.fs.HE101.shell_outlet.temperature.fix(112)
```

In [ ]:
```
result = solver.solve(m, tee=True)

print(result)

#Display a readable report
```

In [ ]:
```
m.fs.Compressor102.report()
m.fs.Heater102.report()
m.fs.HE102.report()
m.fs.HE101.report()
m.fs.Valve102.report()
m.fs.Valve101.report()
m.fs.Separator102.report()
m.fs.Mixer101.report()
```

In [ ]:
```
m.fs.visualize("m")
```

In [ ]:
```
m.fs.display()
```

In [ ]:
```

```

Appendix G

Matlab optimization script

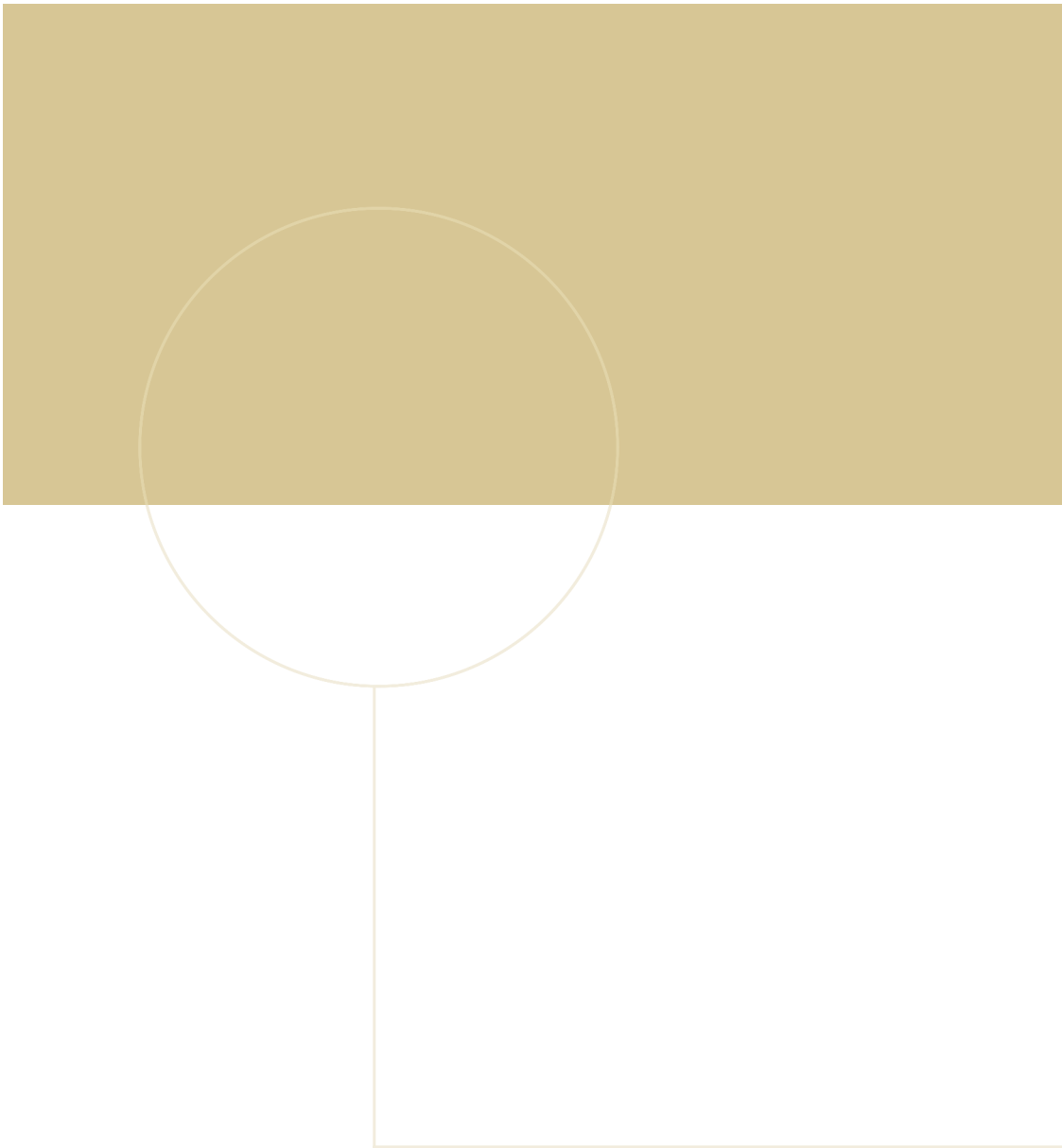
Objective function with constraints and decision variables

```
function Obj= Obj_func(x)

    global count_num
    count_num = count_num +1;
    global RESULT
    Error = 0;
    tic

    hysys = actxserver ('HYSYS.Application');
    hyCase = hysys.ActiveDocument;
    hysolver = hysys.ActiveDocument.Solver;
    hyf=hyCase.Flowsheet;


    % variables

    hysolver.Cansolve=0;

    hyf.Operations.Item('Efficiency').Cell('D2').CellValue = x(1);
    hyf.Operations.Item('Efficiency').Cell('D3').CellValue = x(2);
    hyf.Operations.Item('Efficiency').Cell('D4').CellValue = x(3);
    hyf.Operations.Item('Efficiency').Cell('D6').CellValue = x(4);
    hyf.Operations.Item('Efficiency').Cell('B13').CellValue = x(5);
    hyf.Operations.Item('Efficiency').Cell('D18').CellValue = x(6);
    hyf.Operations.Item('Efficiency').Cell('B8').CellValue = x(7);
    hyf.Operations.Item('Efficiency').Cell('B9').CellValue = x(8);
    hysolver.Cansolve=1;

    % constraints

    try
        c(:,1)=hyf.Operations.Item('Efficiency').Cell('B17').CellValue;
        c(:,2) =hyf.Operations.Item('Efficiency').Cell('G9').CellValue;
        c(:,3) =hyf.Operations.Item('Efficiency').Cell('G11').CellValue;
        Power =hyf.Operations.Item('Efficiency').Cell('G4').CellValue;


        if (c(:,1) +273.15) < 0 | c(:,2) < 3 | c(:,3) < 3
            Obj=Power+100000;
            Error = 1;
            cpu_time = toc;
            RESULT(count_num,:) = [count_num Error cpu_time x c Power Obj];
            save('RESULT','RESULT');
        else
            Obj=Power;
            Error = 0;
            cpu_time = toc;
            RESULT(count_num,:) = [count_num Error cpu_time x c Power Obj];
            save('RESULT','RESULT');

        end
```

```
    catch exeption

        Obj=10000;
        Power=10000;
        Error=2;
        cpu_time = toc;
        RESULT(count_num,:) = [count_num Error cpu_time x c Power Obj];
        save('RESULT','RESULT');
        return
    end
```

PSO run-script

```
clear all;
clc;
close all;

hysys = actxserver ('HYSYS.Application');
hyCase = hysys.ActiveDocument;
hysolver = hyCase.Solver;
hyf=hyCase.Flowsheet;

global count_num
global RESULT
RESULT = [];
count_num = 0;

x0 = [0.3 0.2 0.7 1 0.2 6500 4.5 13.5]; %nN2, nCh4, nC2H6, nC4H10, FR1,
% Molflow, LP, HP

LB =[ 1e-4; 1e-4; 1e-4; 1e-4; 0.1; 2000; 1; 10];
UB =[ 1; 1; 1; 1; 0.9; 10000; 5; 25];

options = optimoptions('particleswarm')
options = optimoptions('particleswarm','SwarmSize',100);
options = optimoptions(options,'FunctionTolerance', 1e-6);
options = optimoptions(options,'MaxIterations', 100);
options = optimoptions(options,'MaxStallIterations', 20); % default=20
options = optimoptions(options,'Display', 'iter');
options = optimoptions(options,'PlotFcn',@pswplotbestf);
options = optimoptions(options,"HybridFcn","patternsearch");
options.InitialSwarmMatrix = x0;

tic
[x,fval,exitflag,output]= particleswarm(@Obj_func,8,LB,UB,options);
toc
```



NTNU

Kunnskap for en bedre verden