

SUBPRO

SUBPRO Summer project 2022

Modelling of Offshore Blue Hydrogen Production with Carbon Storage



Figure 1: The Sleipner field in the North Sea. (Photo: Harald Pettersen / Equinor ASA)

Author: Yoonsik Oh

Supervisors: Johannes Jäschke

Co-supervisors: Evren Mert Turan

Magne Hillestad

Gro Mogseth

July 30, 2022

Contents

1	Job description	2
2	Process description	2
2.1	Flowsheet	2
3	Simulation	4
3.1	Aspen Hysys Model	4
3.1.1	Case simulation	4
3.2	Model in Python	6
3.3	Gas heated- and autothermal reformer	7
3.4	Initial complete model	8
3.5	Improved new model	11
4	Results and discussion	11
5	Future considerations	14
A	Python code	16
A.1	Defining and unpacking variables	16
A.2	Enthalpy	18
A.3	Equilibrium constant	19
A.4	Gas heated reformer	20
A.5	Autothermal reformer	21
A.6	Initial model	22
A.7	Mixing temperature	30
A.8	Prereformer	30
A.9	Post ATR temperature	31
A.10	Isothermal temperature shift reactor	32
A.11	New improved model	32
A.12	Stream summary print function	40
A.13	Error output	41
A.14	Mass balance check function	41
A.15	Composition print function	42
B	Hysys simulation with pure oxygen	46
C	Hysys simulation with 90% oxygen	50
D	Hysys simulation with 40% oxygen	54

1 Job description

The goal of this summer project was to model an offshore blue hydrogen production with carbon capture and storage (CCS) and further researching towards a net-zero hydrogen production. However, the scope for the project was to only build the necessary models relevant to the control volume which is the hydrogen production and the CCS plant. This means that this project would not go into detail of the feasibility of different methods for storing the captured carbon (CO_2). Many different methods of producing blue hydrogen ($> 95\% \text{CO}_2$ captured) was considered, but the deciding factors for choosing the final method was based on economy, efficiency and spacing. Spacing is especially important in this project since the plant is designed to be on an offshore platform, where spacing is limited and expensive to expand. Further, this project will build the basis of a possible future project, a feasibility analysis.

2 Process description

There are already many known methods to produce blue hydrogen. The main issue in this project is that the plant is being built on an offshore platform which has limited amount of space and weight and the building cost will be higher. In a typical blue hydrogen production plant, a large furnace is used which takes most of the space in a steam reforming plant. A company named *Johnson Matthey* has come up with a method of producing hydrogen that instead of using a large furnace, it instead uses a combination of a gas heated reformer and an autothermal reformer^[1]. This method also has a high hydrogen purity and low carbon emission as most of the CO_2 will be captured and stored. More details about the conversion is explained later in the report. Using this process flowsheet, a model from the software *Aspen Hysys* has been made to test out different cases, which will also be discussed later.

2.1 Flowsheet

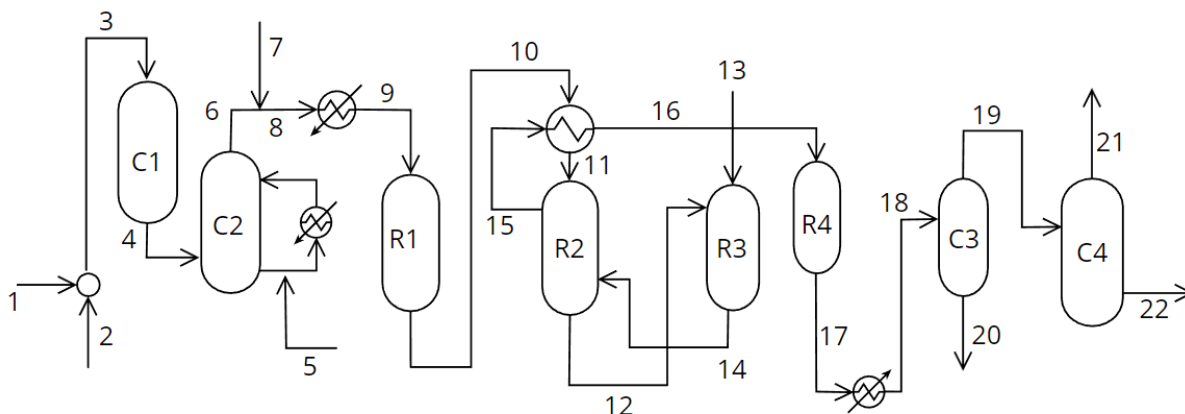


Figure 2: Flowsheet of blue hydrogen production

The complete flowsheet can be seen in fig. 2. Stream 1 is the stream of natural gas extracted from the subsea. The natural gas consists mainly of methane, but also some other organic compounds like ethane and propane. There are small trace of N_2 , O_2 , CO_2 and other small compounds. When discussing the simulated model further down, an average range of each compound and the used for the model will be shown^[2]. Stream 2 consists of hydrogen and its purpose is to turn all sulfuric compounds into H_2S . Then stream 3 containing the natural gas and H_2S will enter the first column, C1, which is the purification step. Here most of the H_2S will be captured by some pellets to be later removed. This step is important as most sulfuric compounds damages the catalysts used later in the process, which is important when considering both economy and yield.

After the purification step, stream 4 will enter the second column, C2, which is the saturator. The purpose of the saturator is to saturate the stream before the reactors with water so the reactions happening in the reactors is shifted towards right side. Here a steam to carbon ratio was set to 2.5 where carbon is mostly methane. After the saturator process, stream 6 will mix with stream 7, which will be heated before the prereforming process. The prereforming reactor R1s purpose is to remove all the heavier hydrocarbon compounds with more carbons than methane, into methane. This is done by reacting the hydrocarbons with water into CO and hydrogen, but due to the low temperature and with correct conditions, the CO and H_2 will turn back into methane and water, mainly due to the equilibrium.

Stream 10, which is the outlet stream of R1 will enter a heat exchanger which will heat up the stream from the outlet stream of the gas heated reformer (GHR), R2. In GHR the stream enters catalyst filled pipelines and will enter another reformer. This reactor is the autothermal reformer (ATR), R3 and in this reformer the equilibrium will be heavily shifted towards right due to the high temperature and the reactions happening are endothermic. The high temperature is gained by burning a feed with pure oxygen. Later on, this will be discussed as pure oxygen is hard to come by on a offshore platform. After the reactions in the ATR, the outlet stream, stream 14, will enter the GHR, and heat up the pipeline which contain stream 11. This concept is similar to a tube and shell heat exchanger. There are two main reactions in the process. The first one is the steam-methane reforming and the other reaction is the water-gas shift reaction which is shown in eq. (2.1.1) and eq. (2.1.2), respectively.



The exit stream of GHR, stream 15, will enter a heat exchanger where it heats up the inlet stream for GHR. Stream 16 will then enter R4, which is the isothermal reactor. The goal of this reactor is to remove as much CO as possible. It does this by only letting the water-gas shift reaction happen by using special catalysts. After R4, stream 17 now contains mainly of H_2 , CO_2 and H_2O . To remove H_2O , stream 17 enters a cooler and enters C3, which is a column for removing condensate. Stream 19 now will contain mainly of CO_2 and H_2 .

To separate these hydrogen from other components, a pressure swing adsorption has been chosen as a separator. This is due to other methods like separation with amines takes up lot of space and is heavy.

3 Simulation

3.1 Aspen Hysys Model

To simulate different cases, a model was made using the software Aspen Hysys. In this model, cases of how different concentration of nitrogen introduced in the oxygen feed into ATR would affect the process. Mainly due to hydrogen being introduced with nitrogen can form nitrous compounds like NH_3 and HCN . For the reforming part, this would not be a problem as the reactors uses nickel catalysts which degrades NH_3 into nitrogen and hydrogen. The isothermal shift reactor uses a copper catalysts which oxidizes the ammonia.

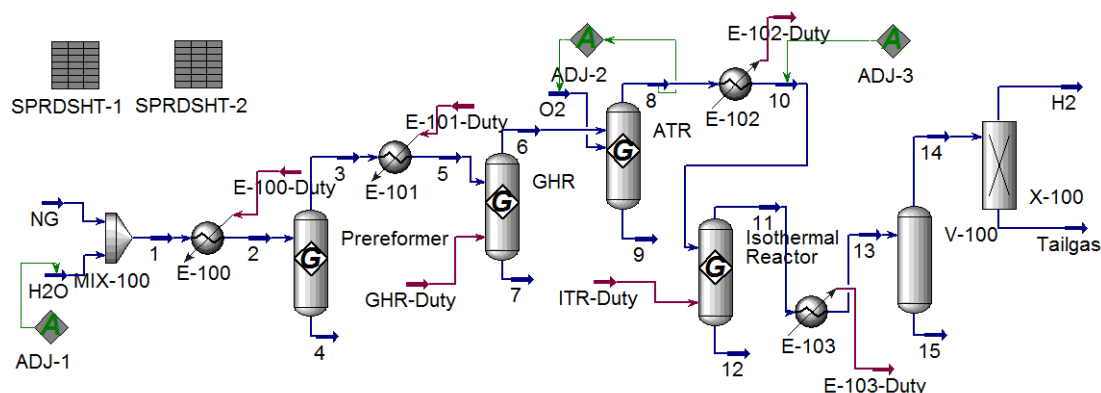


Figure 3: Aspen Hysys model

3.1.1 Case simulation

The simulation starts after the saturator process, so in other words, NG stream in fig. 3 is stream 6 in fig. 2. The composition of this stream is shown in table 1. The software has made few calculations by using adjust blocks. The first adjust block were used to control the H_2O stream such that the stream before the prereformer has a steam to carbon ratio of 2,5. The second adjustment block is to control the oxygen feed stream into the ATR such that the outlet stream has an temperature of 1050 °C. The third adjustment block were used to control the outlet temperature after the heat exchanger after ATR such that the energy gained from the heat exchanger is the same as the energy used in the GHR.

When considering different ways to create an oxygen stream, there are space and economy to consider. Air separation by membrane is the most economic way and takes up the least amount of space for separating air into a stream of oxygen. The downside of membrane separation is the purity. This nitrogen forms into ammonia, but even though it might not

damage the catalysts, it is still a inert component that needs to be removed before the last stage where the tailgas is sent to storage and hydrogen transported to land which needs to be as pure as possible. The most used methods are electrolysis, cryogenic separation and membranes.

Electrolysis splits a water molecule into hydrogen and oxygen and has no impurities compared to the other methods. This process also creates hydrogen, which is one of the goals in this project anyways. The downside of electrolysis is the high energy cost for the process. Splitting only one water molecule has a theoretical minimum of 237 kJ, which is the standard gibbs free energy of forming water. Cryogenic separation might be the best option of separating air to achieve pure oxygen (99.99% pure), but the main problem with cryogenic separation is the huge space it takes up and the weight it requires. It is also energy intensive, which makes the electrolysis a better option anyways. The last method that was considered in this project was air separation by membrane. Though it is economic viable and takes up small space compared to the other methods, the oxygen purity is low. Some studies claim that a oxygen purity of 90% is achieved^[3], but a commercial membrane separator can only achieve between 25%-40% oxygen purity, which is too low purity to apply as feed to ATR. However, the Hysys model will be used to run two different cases, one with pure oxygen stream from electrolysis and one with 90% and 40% pure oxygen stream to test out how it will affect the process. Ultimately, there can be allowed some degree of nitrogen concentration, due to the last separation step, which is the pressure swing adsorption which can separate hydrogen from CO₂, but also other components like N₂ and NH₃. This will be later on discussed after the cases.

Compound	%mol
CH ₄	78.24
C ₂ H ₆	6.10
C ₃ H ₈	6.70
n-C ₄ H ₁₀	2.48
i-C ₄ H ₁₀	1.41
C ₅ H ₁₂	3.70
H ₂ O	≈ 0
H ₂	≈ 0
CO	≈ 0
CO ₂	1.34
H ₂ S	≈ 0

Table 1: Average composition of natural gas in the north sea

When running the simulations on HYSYS, the composition on table 1 were used with a flow of 1000 kmol h⁻¹, which is a bit high, but it was a value that was arbitrary chosen to observe the numbers. Later when trying to switch up the numbers to more realistic numbers,

the HYSYS model breaks when trying to run the simulation due to inconsistency problems with having multiple adjust blocks running simultaneously. Luckily, a backup was saved with the initial value of 1000 1000 kmol h⁻¹, and the wanted observations could be made. In the case of pure oxygen stream feed in to the ATR, the product stream before the PSA separation consisted mostly of hydrogen and carbon dioxide, which is desirable. Hydrogen had a %mole of 74.76 and CO₂ had a %mole of 23.92. The rest of the compound consisted of methane, carbon monoxide, water and nitrogen. The nitrogen is due to the fact that it is inert, except when it may react to other compounds like ammonia or HCN, but in our case, even though some nitrogen might have converted into ammonia or HCN, the amount would be insignificant. Rest of the compounds can be removed by operating with reactors with higher conversion rate and better separators.

In the case of 10% mole nitrogen included in the oxygen feed in to the ATR, the results were similar as with pure oxygen. And as expected, the nitrogen were inert during the process and will have a bit larger composition than with pure oxygen, since in the case of pure oxygen, the only nitrogen coming is from the feed, which is also arbitrary in the case of north sea natural gas. The %mole of hydrogen and CO₂ were found to be 73.83 and 23.67, respectively and the other remaining compounds were the same as the case with pure oxygen. But the nitrogen had a %mole of 1.31 instead of 0.12 as with the previous case. Mole percent of 1.31 is larger than all the other compounds excluding hydrogen and CO₂ combined.

In the more extreme case, where a more common air membrane separator is used to split air into oxygen and nitrogen, the oxygen flow is 40% while the nitrogen is 60%. The results were not ideal as the hydrogen were only 60% mole and nitrogen was 15.39% mole which means that the hydrogen to nitrogen ratio in the product stream was 4:1. It can be concluded that an air membrane separation with today's technology, should not be used for the system. The data sheet of all stream information and duty required for different cases is shown in appendix A.15.

3.2 Model in Python

As Aspen HYSYS software's purpose was only to look how the numbers deviated with changing different units, the real modelling goal of this project was done in Python. The function `fsolve`, a package from `scipy.optimize`, was used to solve all the systems of nonlinear equations. The same composition from table 1 and natural gas stream data from *Norsk Petroleum* was used to estimate an average gas flow from a typical oil platform. As the platform Troll was producing average of 35 million Sm³/year^[4], a flow of 4000 Sm³/hour was used in the project, which was estimated to be 145.4 mol/hour, where it was estimated to be 0.829 kg of natural gas per 1 Sm³. The complete flowsheet with stream numbers used in the Python model is shown in fig. 4 where the stream number in the flowsheet is the same as the number used in the Python model.

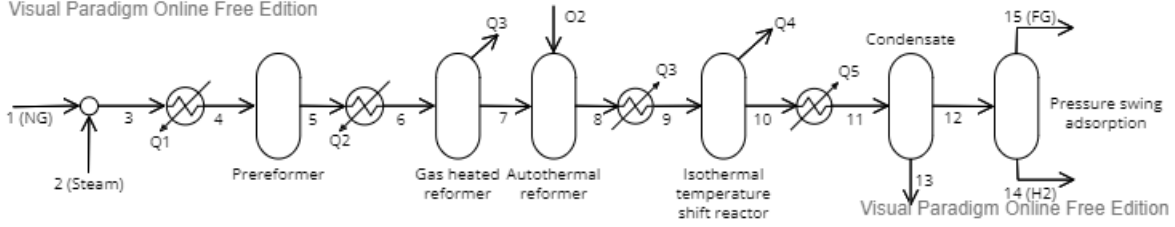


Figure 4: Flowsheet used in the Python model

3.3 Gas heated- and autothermal reformer

The model was first tested by modelling GHR by itself and the ATR by itself. The code is shown in appendix A.4. It has the inlet stream with the inlet temperature as input values and will solve to find the outlet stream and the duty required. The GHR does not actually need a duty since it is a adiabatic reactor, but to model the GHR and ATR connected together, we first calculate the heat GHR requires, and then give the same duty into a heat exchanger after the ATR which should give the outlet temperature after the ATR. While HYSYS model uses a gibbs reactor where it tries to find a composition that gives lowest gibbs energy at the given temperature, we control the reactions happening in the reactor by modelling an equilibrium reactor. The mass balance and energy balance is shown as:

$$\begin{aligned}
 n_{\text{CH}_4} &= n_{0,\text{CH}_4} - \xi_1 \\
 n_{\text{H}_2\text{O}} &= n_{0,\text{H}_2\text{O}} - \xi_1 - \xi_2 \\
 n_{\text{H}_2} &= n_{0,\text{H}_2} + 3\xi_1 + \xi_2 \\
 n_{\text{CO}} &= n_{0,\text{CO}} + \xi_1 - \xi_2 \\
 n_{\text{CO}_2} &= n_{0,\text{CO}_2} + \xi_2 \\
 nH(T) &= n_0H(T_0) + Q,
 \end{aligned} \tag{3.3.1}$$

where ξ_1 and ξ_2 are extent of reactions for the steam methane reforming reaction and water gas shift reaction, respectively. We set ξ_1 as $\xi_1 = n_{0,\text{CH}_4} - n_{\text{CH}_4}$ and $\xi_2 = n_{\text{CO}_2}$. The equilibrium is as shown:

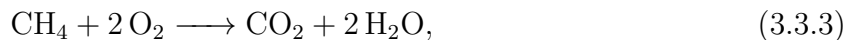
$$\begin{aligned}
 K_{\text{SMR}} &= \frac{x_{\text{CO}} \cdot x_{\text{H}_2}^3}{x_{\text{CH}_4} \cdot x_{\text{H}_2\text{O}}} \\
 K_{\text{WGSR}} &= \frac{x_{\text{CO}_2} \cdot x_{\text{H}_2}}{x_{\text{CO}} \cdot x_{\text{H}_2\text{O}}}
 \end{aligned} \tag{3.3.2}$$

To calculate the enthalpy for each stream, a function was made to easily calculate the enthalpy of each component at the given temperature, so when calculating the total enthalpy of the stream, a simple dot product between the component stream array and the enthalpy array could be used. There were made two enthalpy functions, one with only enthalpy for the five first components which can be found during the whole process; CH_4 , H_2O , H_2 , CO and CO_2 and another enthalpy function which also includes heavier hydrocarbons. The heavy enthalpy function was only used until the prereformer, as after the prereformer, all the heavier hydrocarbons has converted into CH_4 , H_2O and CO . Since there was hard to find

a source with data for every compound, an accurate data from NIST was used to find the enthalpy for the 5 first components^[5], while a more ideal gas enthalpy data was used for the heavier compounds^[6]. The code used for calculating the enthalpy is shown in appendix A.2

While one may predicted that the steam methane reforming and water gas shift reaction was well studied with equilibrium data easily available, this was not the case. As there were no problem finding the equilibrium data by themselves, it was hard to find a study where the equilibrium constant data was found together. There was one study that showed both equilibrium constant data, but the data for the water gas shift reaction were unrealistic and gave odd number^[7]. For example at a temperature of 250 °C, the water gas shift reaction equilibrium constant should have a number around 100, but the one from the study gave a number less than 0.001, which will give strange numbers for the reactors. That is why data from another study will be used for the water gas shift reaction^[8]. The code used for calculating the equilibrium constant for both reactions is shown in appendix A.3

The model for ATR is similar as GHR, but in the case for ATR, the energy balance differ. Instead of finding a duty from the enthalpy difference between the inlet and outlet temperature, we set the reactor temperature and find the amount of oxygen the reactor needs to achieve the given temperature. In addition, an chemical equation for methane combusting with oxygen is included in the model, which changes the mass balances. It was assumed that there was pressure drop, there was full combustion, meaning that methane and oxygen only reacted to methane and water. The additional reaction is given as:



and the new mass balance is given as:

$$\begin{aligned} n_{\text{CH}_4} &= n_{0,\text{CH}_4} - \xi_1 - \xi_3 \\ n_{\text{H}_2\text{O}} &= n_{0,\text{H}_2\text{O}} - \xi_1 - \xi_2 + \xi_3 \\ n_{\text{H}_2} &= n_{0,\text{H}_2} + 3\xi_1 + \xi_2 \\ n_{\text{CO}} &= n_{0,\text{CO}} + \xi_1 - \xi_2 \\ n_{\text{CO}_2} &= n_{0,\text{CO}_2} + \xi_2 + \xi_3 \\ nH(T) &= n_0H(T_0) + \Delta H_{\text{O}_2} \cdot n_{\text{O}_2}, \end{aligned} \quad (3.3.4)$$

where ξ_3 is the extent of reaction of eq. (3.3.3) the amount of oxygen moles required and ΔH_{O_2} is the change of enthalpy of oxygen. It was also assumed that it was only oxygen that was combusting and not other compounds, when it reality, other compounds like methane and hydrogen might also combust and giving more heat in the reactor. The code for the model is given in appendix A.5.

3.4 Initial complete model

After testing out the GHR and ATR models, a complete model of the entire flowsheet was written in Python. To make the code easier to understand, the code was divided up to virtual blocks with names to easily identify where all the streams and components was in a

complicated 200+ lines of code. There were total of 11 blocks, and a summary of the goal of each blocks is shown in table 2.

Block number	Goal
1	<ul style="list-style-type: none"> • Find the required amount of steam with a 2.5 steam to carbon ratio. • Component balance between stream 1, 2 and 3. • Calculating the temperature of stream 3 after mixing stream 1 and 2.
2	<ul style="list-style-type: none"> • Component balance between stream 3 and 4. • Heat duty required before heating up stream 3 to 693K.
3	<ul style="list-style-type: none"> • Prereformer. (Equilibrium) • Component balance between stream 4 and 5.
4	<ul style="list-style-type: none"> • Pre-GHR heat exchanger, calculating the required duty. • Component balance between stream 5 and 6.
5	<ul style="list-style-type: none"> • Gas heated reformer. (Equilibrium) • Component balance between stream 6 and 7. • Calculating required duty in the GHR.

Table 2: Summary of purpose and goal for each block

The system has 82 variables with 82 independent equations, which fsolve was able to solve. The code for defining and unpacking variables is shown in appendix A.1. The problem with the model was that even though the system of equations was solved, the results were not realistic. For example, there were streams that were negative, the isothermal temperature shift reactor required heat and not the other way around, and the temperature after post-ATR heat exchanger was negative. And since the code was written as a big system, it was hard to find where the error was, but the connection between GHR and ATR was the main suspect. Even though these two reactors worked fine by themselves, an error occurred when GHR required a too big duty that the post ATR heat exchanger could cool down after the ATR reactor. The code for the initial model is shown in appendix A.6 To improve this model,

6	<ul style="list-style-type: none"> • Autothermal Reformer. • Calculating required amount of oxygen to achieve the set temperature of the reactor.
7	<ul style="list-style-type: none"> • Post-ATR heat exchanger • Finding the actual outlet temperature after the ATR by cooling the outlet stream with the same duty that is required for the GHR.
8	<ul style="list-style-type: none"> • Isothermal temperature shift reactor. (Equilibrium) • Calculating the duty recovered from the reactor
9	<ul style="list-style-type: none"> • Precondensate heat exchanger • Calculating the heat recovered by cooling down the stream to 313K. • Component balance between stream 10 and 11.
10	<ul style="list-style-type: none"> • Condensate. Separating liquid water from the gas stream with a given split ratio. • Component balance between stream 11 and 12. Stream 13 is calculated outside of the nonlinear equation system.
11	<ul style="list-style-type: none"> • Pressure swing adsorption. Separating the stream 12 to product stream (stream 14) and flue gas stream (stream 15). Product stream will contain mostly hydrogen.

Table 3: Summary of purpose and goal for each block

a new model was made where a "forward passing" method used. Most of the equations were just a mass balance, so the information from the previous stream could pass directly into the next stream, given that there was no reaction, which decreases the amount of variables and required amount independent equations drastically. Rest of the blocks code will be explained in section 3.5.

3.5 Improved new model

In the new improved model, almost all component balance will be forward passing, meaning that the previous component stream data will be passed directly as a new stream data. The model will have stream 1 and stream 2 and inputs since these are given, and the output will be all 15 streams, oxygen required in the ATR and all the duty required/recovered. Block 1 in the new improved model will have a function that solves the new temperature after mixing with `fsolve`. The code for this is shown in appendix A.7. Block 2 will have only forward passing, where components from stream 3 will be stream 4 and the duty required to heating up the stream is calculated by taking the difference in enthalpy between stream 3 and 4. Block 3 is solving stream 5 by a new prereformer function by using stream 4 as input. The code for this new function is shown in appendix A.8. Block 4 is similar to block 2, but only with stream 5 and 6. The outlet temperature is set to 973K.

Block 5 and 6, which is the GHR and ATR, will be solved by the same function that was made at the start when testing the model. The code for GHR and ATR function is shown in appendix A.4 and appendix A.5, respectively. The operating temperature of GHR and ATR is set to 1073K and 1600K. In reality, the operating temperature is lower, but the reason why the operating temperature is higher will be discussed later in the results and conclusion section. Block 7 calculates the temperature after the heat exchanger after the ATR, by using the same duty that GHR requires. The code for the post ATR heat exchanger is shown in appendix A.9. Block 8 is the isothermal temperature shift reactor and calculates stream 10 and duty recovered by using stream 9 at the given operating temperature of 523K. The code for this block is shown in appendix A.10.

Block 9 is the pre-condensate part, and is similar to block 2 and 4, only with stream 10 and stream 11. Block 10 and 11 is similar to each other, as the stream is splitted into two streams based on the splitting factor that is given. The split factor in these two blocks are something to reconsider in the future as they are near ideal numbers and will differ from a realistic condensator and PSA. To observe the results easier, 4 functions has been made, which are the functions `summary()`, `error_output()`, `mass()` and `composition`. The `summary` function is print out all the streams, the duty required/recovered and the amount of oxygen required like a table. The `error_output()` function prints out the error in `fsolve` for all equations, but this is only used in the first initial model as the new improved model will only use forward passing and its own functions for each block. The function, `mass()`, prints out the mass streams, and is used to check if the mass balance is satisfied at all points. Lastly, the function `composition()` is similar to the `summary()` function, only that it prints out the composition of each component in the streams in a nice table. The code for `summary()`, `error_output()`, `mass()` and `composition` is shown in appendix A.12, appendix A.13, appendix A.14 and appendix A.15, respectively.

4 Results and discussion

The initial model that was solving the entire flowsheet at the same time with `fsolve` did solve, but gave unrealistic numbers. The temperature after the post ATR heat exchanger

was either negative kelvin or around 100K, which can be concluded to be odd. Some streams even gave negative numbers, which is physically impossible. The mass was not conserved at all points either. The main suspect of this error was indeed the connection between the GHR and ATR. If GHR has too high duty, the stream after ATR does not have enough energy to cool down the stream to a realistic number since the duty used in GHR should be the same duty that is being recovered. Realistic, GHR and ATR should be operated around 973K and 1323K, respectively, but in our model, they were set to 1073K and 1600K, respectively. The new improved model showed some promising numbers, where the product stream before the pressure swing adsorption contained almost entirely of H_2 and CO_2 . There was 2.69% CO containing in the stream, but this can be improved by finding more optimal temperatures or equilibrium constants for the isothermal temperature shift reactor. An summary table of the mole streams, mass streams and mole composition in each stream can be found in fig. 5, fig. 6 and fig. 7, respectively.

Stream	CH4 [mol/s]	H2O [mol/s]	H2 [mol/s]	CO [mol/s]	CO2 [mol/s]	T [K]	P [bar]	C2H6 [mol/s]	C3H8 [mol/s]	n-C4H10 [mol/s]	i-C4H10 [mol/s]	C5H12 [mol/s]
Stream 1:	113.76	0.00	0.00	0.00	1.95	311.00	30.00	8.87	9.74	3.61	2.05	5.38
Stream 2:	0.00	358.52	0.00	0.00	0.00	423.00	30.00	0.00	0.00	0.00	0.00	0.00
Stream 3:	113.76	358.52	0.00	0.00	1.95	381.41	30.00	8.87	9.74	3.61	2.05	5.38
Stream 4:	113.76	358.52	0.00	0.00	1.95	693.00	30.00	8.87	9.74	3.61	2.05	5.38
Stream 5:	178.58	295.52	59.49	0.33	33.28	630.52	30.00	0.00	0.00	0.00	0.00	0.00
Stream 6:	178.58	295.52	59.49	0.33	33.28	973.00	30.00	0.00	0.00	0.00	0.00	0.00
Stream 7:	22.54	122.16	544.93	139.05	50.60	1073.00	30.00	0.00	0.00	0.00	0.00	0.00
Stream 8:	0.00	217.23	494.94	184.68	27.51	1600.00	30.00	0.00	0.00	0.00	0.00	0.00
Stream 9:	0.00	217.23	494.94	184.68	27.51	512.79	30.00	0.00	0.00	0.00	0.00	0.00
Stream 10:	0.00	55.89	656.28	23.34	188.85	523.00	30.00	0.00	0.00	0.00	0.00	0.00
Stream 11:	0.00	55.89	656.28	23.34	188.85	313.00	30.00	0.00	0.00	0.00	0.00	0.00
Stream 12:	0.00	0.06	656.28	23.34	188.85	313.00	30.00	0.00	0.00	0.00	0.00	0.00
Stream 13:	0.00	55.84	0.00	0.00	0.00	313.00	30.00	0.00	0.00	0.00	0.00	0.00
Stream 14:	0.00	0.00	655.62	0.23	1.89	313.00	30.00	0.00	0.00	0.00	0.00	0.00
Stream 15:	0.00	0.06	0.66	23.11	186.96	313.00	30.00	0.00	0.00	0.00	0.00	0.00
n02 [mol/s]:	47.26											
Q1 [kJ]:	6890.45											
Q2 [kJ]:	8918.36											
Q3 [kJ]:	34472.61											
Q4 [kJ]:	-5913.26											
Q5 [kJ]:	-6345.82											

Figure 5: Mole stream table from the new improved model

We can observe that CH_4 , CO and CO_2 is increasing and H_2O decreasing after the prereformer, which is expected. We can also see that CH_4 is almost used up after the GHR and is almost non existent after the ATR, which is also good. That means that the methane has converted into H_2 , which is the ultimate goal of the process. After the isothermal temperature shift reactor, the amount of CO is reduced drastically, due to it converting into CO_2 in the water gas shift reaction. The temperature after the GHR and ATR process (post ATR heat exchanger) is calculated to be 512K, which should be higher, but this cannot be achieved without setting the operating temperature of GHR and ATR unreaistically high. Methods of fixing this will be discussed later in the section when discussing future possibilities.

The model has lot of improvements. There has not been considered any pressure drops in the process, when in reality there will be around 1 bar of pressure drop after each unit. This

```

Stream 1 [g/s]: 3316.0000
Stream 2 [g/s]: 6453.3191
Stream 3 [g/s]: 9769.3191
Stream 4 [g/s]: 9769.3191
Stream 5 [g/s]: 9769.3191
Stream 6 [g/s]: 9769.3191
Stream 7 [g/s]: 9769.3191
Stream 02 [g/s]: 1512.4298
Stream 8 [g/s]: 11281.7489
Stream 9 [g/s]: 11281.7489
Stream 10 [g/s]: 11281.7489
Stream 11 [g/s]: 11281.7489
Stream 12 [g/s]: 10276.6577
Stream 13 [g/s]: 1005.0912
Stream 14 [g/s]: 1400.8823
Stream 15 [g/s]: 8875.7753

```

Figure 6: Mass stream table from the new improved model

Stream	xCH4	xH2O	xH2	xC0	xC02	T	P	xC2H6	xC3H8	xn-C4H10	xni-C4H10	xnC5H12
Stream 1:	0.7826	0.0000	0.0000	0.0000	0.0134	311.00	30.00	0.0610	0.0670	0.0248	0.0141	0.0370
Stream 2:	0.0000	1.0000	0.0000	0.0000	0.0000	423.00	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 3:	0.2258	0.7115	0.0000	0.0000	0.0039	381.41	30.00	0.0176	0.0193	0.0072	0.0041	0.0107
Stream 4:	0.2258	0.7115	0.0000	0.0000	0.0039	693.00	30.00	0.0176	0.0193	0.0072	0.0041	0.0107
Stream 5:	0.3148	0.5210	0.1049	0.0006	0.0587	630.52	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 6:	0.3148	0.5210	0.1049	0.0006	0.0587	973.00	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 7:	0.0256	0.1389	0.6197	0.1581	0.0576	1073.00	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 8:	0.0000	0.2350	0.5354	0.1998	0.0298	1600.00	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 9:	0.0000	0.2350	0.5354	0.1998	0.0298	512.79	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 10:	0.0000	0.0605	0.7100	0.0253	0.2043	523.00	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 11:	0.0000	0.0605	0.7100	0.0253	0.2043	313.00	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 12:	0.0000	0.0001	0.7556	0.0269	0.2174	313.00	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 13:	0.0000	1.0000	0.0000	0.0000	0.0000	313.00	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 14:	0.0000	0.0000	0.9968	0.0004	0.0029	313.00	30.00	0.0000	0.0000	0.0000	0.0000	0.0000
Stream 15:	0.0000	0.0003	0.0031	0.1096	0.8870	313.00	30.00	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 7: Composition of each component in the stream table from the new improved model

means that there are not any compressors considered in the process, when in reality, a large amount of energy will be required to compress the streams at the required level. The flue gas which will be transported back to the reservoir is also at a low pressure when using a pressure swing adsorption since the pressure is being lowered to desorb the CO₂ from the particles that is being absorbed at a high pressure. The pressure for the flue gas needs to be especially higher since when being transported back, it needs to have a high pressure to be able to transport the flue gas back to the reservoir which has a high pressure. The numbers for condensate and the pressure swing adsorption has also been almost ideal splitters. This will be problematic as there are regulations which controls what components are allowed to be transported as carbon storage in a reservoir and how large concentrations of each component can be.

The equilibrium constants used in the model is also being used from different sources, and can be improved by using better constants which will improve the conversion of methane and CO. The model also assumes that the initial stream is purified at the beginning, when in reality there needs to be a part where we are adding hydrogen to convert all the sulfuric

components into H_2S which is being removed by the purifier. The hydrogen can be added by recycling some of the hydrogen product stream after PSA, which this model have not considered. The steam used in the process to achieve a 2.5 steam to carbon ratio needs to be either recycled water after the condensate or purified ocean water. This part of the process should also be included when improving the model in the future. The pressure swing adsorption unit requires also hydrogen to flush out the flue gas from the absorbed particles, which also needs to be modelled. This can be achieved by splitting up the hydrogen product stream, but to find a sweet spot where the process can achieve maximum amount of hydrogen with highest amount of purified hydrogen gas is something that can be achieved by optimizing the process, which should also be a part when improving the model.

This model has considered that the oxygen stream feeding into the ATR is pure, and this can only be achieved by either electrolysis or a cryogenic separation. When considering a cheaper option like air membrane separation, some nitrogen will be included in the process and needs to have a separation unit that can separate the hydrogen and the flue gas with nitrogen. It also needs to be checked if the concentration of nitrogen gained from the given method of achieving an oxygen stream is in bounds of the regulations when transporting flue gas back to the reservoir. Also in the ATR when achieving heat by combusting oxygen with methane, other components will also combust which will give energy to ATR, that might reduce or increase the amount of oxygen that is required to achieve the given operating temperature of the reactor. As the HYSYS model uses a gibbs reactor, it can find the heat gained from combustion quite easily, but when modelling with an equilibrium reactor like in the Python model, this can be quite challenging.

Lastly, when modelling the process in this project, there has not been considered any economic feasibility and sizing in detail. It has been concluded that this method will use less space than a regular steam methane reforming process due to the process using an ATR instead of a large furnace. But if the process is economical feasible with the startup cost and the amount of revenue and the approximated years in payback is something to consider in the future when looking if this project is feasible to build or not.

In conclusion, this project will be a good starting point when building a model that will be used when considering if the whole project is feasible or not. Economy will be a large part when considering to actually build the offshore process plant or not, but it is also worth keeping in mind that a part of this project is to achieve closer to the global net zero goal. It is still important to build an optimized process that can achieve maximum profit while at the same time ensuring an environmental profit. In the next section, future possibilities with improvements and project's potential will be discussed.

5 Future considerations

As discussed in the results and discussion, there are lot of aspects that have been only assumed or not considered at all for a more simple model. My initial plan is to continue this project on my project's thesis and my master's thesis. At least there are couple of things that are maybe quite straight forward to implement, like how achieve purified water, calculating

the correct splitter ratio, achieving oxygen stream and so on. The harder part will be when turning this model from a steady state to a dynamic system where control theory can be applied. Also the economic feasibility will play a large part in the project's thesis or in the master, as it will ultimately decide if the process plant is actually achievable or not.

Other things to research is finding new equilibrium constants, how much energy we can gain in the ATR from combusting other compounds than oxygen, calculate the GHR and ATR together with optimization to find the operating temperature without setting in manually. Carbon capture and storage will play a big part as the last part of the process plant will need to be adjusted to the laws controlling the allowed impurities in the flue gas stream down to the reservoir. Lastly, calculating the work and energy needed to keep the pressure for all streams at a desirable range.

References

- [1] Johnson Matthey. Lch process for the production of blue hydrogen, 2021. URL <https://www.gastechevent.com/media/au0mgmf5/26367-jm-lch-process-production-of-low-carbon-hydrogen-tp.pdf>.
- [2] Tuong-Van Nguyen, Brian Elmegaard, Leonardo Pierobon, Fredrik Haglind, and Peter Breuhaus. *Modelling and analysis of offshore energy systems on North Sea oil and gas platforms*. 10 2012.
- [3] Marina Micari and Kumar Varoon Agrawal. Oxygen enrichment of air: Performance guidelines for membranes based on techno-economic assessment. *Journal of Membrane Science*, 641:119883, 2022. ISSN 0376-7388. doi: <https://doi.org/10.1016/j.memsci.2021.119883>. URL <https://www.sciencedirect.com/science/article/pii/S0376738821008267>.
- [4] Norsk Petroleum. Gjenstående reserver, troll. URL <https://www.norskpetroleum.no/fakta/felt/troll/>.
- [5] Chase M.W, Jr. Nist-janaf thermochemical tables, 1998. URL <https://webbook.nist.gov/cgi/cbook.cgi?Source=1998CHA1-1951>.
- [6] Yasar Demirel. *Green Energy and Technology*. 01 2012. ISBN 978-1-4471-2372-9. doi: 10.1007/978-1-4471-2372-9.
- [7] Yunfei Yan, Jie Zhang, and Li Zhang. Properties of thermodynamic equilibrium-based methane autothermal reforming to generate hydrogen. *International Journal of Hydrogen Energy*, 38(35):15744–15750, 2013. ISSN 0360-3199. doi: <https://doi.org/10.1016/j.ijhydene.2013.06.007>. URL <https://www.sciencedirect.com/science/article/pii/S0360319913014456>.
- [8] Byron Smith, L. Muruganandam, Loganathan Murthy, and Shekhar Shantha. A review of the water gas shift reaction kinetics. *International Journal of Chemical Reactor Engineering - INT J CHEM REACT ENG*, 8, 01 2010. doi: 10.2202/1542-6580.2238.

A Python code

A.1 Defining and unpacking variables

```
from enthalpy import const, enthalpy, enthalpy_heavy, heavy_const
from Keq import K_smr, K_wgsr
import scipy.optimize as opt
import autograd.numpy as np
from autograd import grad, jacobian

# inlet/outlet = [molar flow, P, T/Q/nO2]
C = 12
H = 1.
O = 16
# molar flow component order = CH4, H2O, H2, CO, CO2
Mm = np.array([C * 1 + H * 4, H * 2 + O, H * 2, C + O, C + O * 2])
# This is to check the conservation of mass
# molar flow component order = CH4, H2O, H2, CO,
CO2, C2H6, C3H8, n-C4H10, i-C4H10, C5H12

Mm_heavy = np.array(
    [C * 1 + H * 4, H * 2 + O, H * 2, C + O, C + O * 2, C * 2 + H * 6, C * 3
     + H * 8, C * 4 + H * 10, C * 4 + H * 10, C * 5 + H * 12])

inletcomposition = np.array([78.24, 0.0, 0.0, 0.0, 1.34, 6.10, 6.7, 2.48,
1.41, 3.7]) / 100 # [Sm^3/h]
inletstream_kg = 4000 * 0.829
avgMm = np.dot(Mm_heavy, inletcomposition)
inletstream_mol = inletstream_kg / avgMm

inletstream = np.multiply(inletstream_mol, inletcomposition)
other = np.array([30.0, 311.0])

inletstream = np.append(inletstream, other) # This is the given values
```

```

guess = np.array([400, 30.0, 10000,  # n2
                  100, 400, 0.0, 0.0, 1, 30.0, 400.0,  # n3
                  100, 400, 0.0, 0.0, 1, 30.0, 4000.0,  # n4
                  200, 400, 100, 1, 10, 30.0, 700.0,  # n5
                  200, 300, 100.0, 1, 10, 30.0, 20000.0,  # n6
                  10, 300, 500.0, 100.0, 1.0, 30.0, -5000,  # n7
                  1, 200, 500.0, 100.0, 1.0, 30.0, 30,  # n8
                  1, 200, 500.0, 100.0, 1.0, 30.0, 200,  # n9
                  1, 10, 500.0, 200, 1.0, 30.0, -6000.0,  # n10
                  1, 10, 500.0, 10.0, 1.0, 30.0,  # n11
                  1, 1.0, 500.0, 10.0, 100, 30.0, 313.0,  # n12
                  30.0, 313.0,  # n13
                  1.0, 1.0, 500.0, 1.0, 1.0, 30.0, 313.0,  # n14
                  30.0, 313.0])  # n15

splitratio = np.array([[1.0, 0.001, 1.0, 1.0, 1.0], [0.01, 0.01, 0.999, 0.01, 0.01]])

```

```

# Input is given, Q and output is the initial guess values
def system(input, output, f):
    # Defining and unpacking all the variables
    n1CH4, n1H2O, n1H2, n1CO, n1CO2, \
    n1C2H6, n1C3H8, n1i_C4H10, n1n_C4H10, n1_C5, P1, T1 = input
    f1, f2 = f
    # Note: on stream 8, we set the reactor temperature depending on O2 stream
    n2H2O, P2, Q1, \
    n3CH4, n3H2O, n3H2, n3CO, n3CO2, P3, T3, \
    n4CH4, n4H2O, n4H2, n4CO, n4CO2, P4, Q2, \
    n5CH4, n5H2O, n5H2, n5CO, n5CO2, P5, T5, \
    n6CH4, n6H2O, n6H2, n6CO, n6CO2, P6, Q3, \
    n7CH4, n7H2O, n7H2, n7CO, n7CO2, P7, Q4, \
    n8CH4, n8H2O, n8H2, n8CO, n8CO2, P8, nO2, \
    n9CH4, n9H2O, n9H2, n9CO, n9CO2, P9, T9, \
    n10CH4, n10H2O, n10H2, n10CO, n10CO2, P10, Q5, \
    n11CH4, n11H2O, n11H2, n11CO, n11CO2, P11, \
    n12CH4, n12H2O, n12H2, n12CO, n12CO2, P12, T12, \

```

```

P13, T13, \
n14CH4, n14H2O, n14H2, n14CO, n14CO2, P14, T14, \
P15, T15 = output

# Defining variables
n2 = n2H2O
n5 = n5CH4 + n5H2O + n5H2 + n5CO + n5CO2
n7 = n7CH4 + n7H2O + n7H2 + n7CO + n7CO2
n8 = n8CH4 + n8H2O + n8H2 + n8CO + n8CO2
n10 = n10CH4 + n10H2O + n10H2 + n10CO + n10CO2

nCarbon = n1CH4 + n1C2H6 + n1C3H8 + n1n_C4H10 + n1i_C4H10 + n1_C5

```

A.2 Enthalpy

```

# H298, A, B, C, D, E, F, G, H
# order of compounds: CH4, H2O, H2, CO, CO2
# Temperature ranges for the constants:
# CH4 (298-1300), H2O (500-1700), H2 (298-1000), CO (298-1300), CO2 (298-1200)

const = np.array([[-74.87, -0.703029, 108.4773, -42.52157, 5.862788,
0.678565, -76.84376, 158.7163, -74.87310],
                  [-241.83, 30.09200, 6.832514, 6.793435, -2.534480,
0.082139, -250.8810, 223.3967, -241.8264],
                  [0.0, 33.066178, -11.363417, 11.432816, -2.772874,
-0.158558, -9.980797, 172.707974, 0.0],
                  [-110.53, 25.56759, 6.096130, 4.054656, -2.671301,
0.131021, -118.0089, 227.3665, -110.5271],
                  [-393.52, 24.99735, 55.18696, -33.69137, 7.948387,
-0.136638, -403.6075, 228.2431, -393.5224]
                  ])

# Order of components: CH4, H2O, H2, CO, CO2, C2H6, C3H8, n-C4H10, i-C4H10, C5+
# Order of coefficients: A, B, C, H298, Cp298/R <- remember to multiply with 8.314
heavy_const = np.array([[1.702, 9.081, -2.164, -74.520 ,4.217],

```

```

[3.470, 1.450, 0.000, -241.818, 4.038],
[3.249, 0.422, 0.000, 0.000, 3.468],
[3.376, 0.557, 0.000, -110.525, 3.507],
[5.457, 0.557, 0.000, -393.509, 4.467],
[1.131, 19.225, -5.561, -83.820, 6.369],
[1.213, 28.785, -8.824, -104.680, 9.011],
[1.935, 36.915, -11.402, -125.790, 11.298],
[1.935, 36.915, -11.402, -125.790, 11.298],
[2.464, 45.351, -14.111, -146.760, 14.731]])

```

```

def enthalpy(v, T): # kJ/mol
    T = T / 1000
    h_vec = v[:,0] + v[:,1] * T + v[:,2] / 2 * T ** 2 + v[:,3] / 3 * T ** 3 + \
            v[:,4] / 4 * T ** 4 - v[:,5] / T + v[:,6] - v[:,8]
    h_vec = enthalpy_heavy(heavy_const, T*1000)[:5]
    return h_vec

def enthalpy_heavy(v, T):
    h_vec = v[:,3] + (((v[:,0]*T + v[:,1]/2*T**2*10**(-3) +
    v[:,2]/3*T**3*10**(-6)) - (v[:,0]*298 + v[:,1]/2*298**2*10**(-3) +
    v[:,2]/3*298**3*10**(-6))))*8.314 /1000)

    return h_vec

```

A.3 Equilibrium constant

```

def K_smr(T):
    return (np.exp(-22790 / T + 8.156 * np.log(T) - 4.421 / 10 ** 3 * T
    - 4.330 * 10 ** 3 / (T ** 2) - 26.030))

def K_wgsr(T):
    return np.exp(5693.5/T + 1.077*np.log(T) + 5.44e-4*T - 1.125e-7*T**2 -
    49170/(T**2)-13.148)

```

A.4 Gas heated reformer

```
def GHR_Reactor(inlet, outlet):
    n0CH4, n0H2O, n0H2, n0CO, n0CO2, T0 = inlet
    nCH4, nH2O, nH2, nCO, nCO2, Q = outlet
    ksi1 = n0CH4 - nCH4
    ksi2 = nCO2 - n0CO2
    ntot = nCH4 + nH2O + nH2 + nCO + nCO2
    T = 973

    eq1 = K_smr(T)*((nCH4/ntot) * (nH2O/ntot )) -
    (((nCO/ntot) * (nH2/ntot) ** 3))

    eq2 = K_wgsr(T)*((nCO/ntot) * (nH2O/ntot))
    - (((nCO2/ntot) * (nH2/ntot)))

    eq3 = nH2O - n0H2O + ksi1 + ksi2

    eq4 = nH2 - n0H2 - 3 * ksi1 - ksi2

    eq5 = nCO - n0CO - ksi1 + ksi2

    #MB = np.array([eq1, eq2, eq3, eq4, eq5])

    eq6 = np.dot(inlet[:5],enthalpy(const, T0)) -
    np.dot(outlet[:5],enthalpy(const, T)) + Q

    #EB = np.array([eq6])

    balances = np.array([eq1,eq2,eq3,eq4,eq5,eq6])

    return balances
```

A.5 Autothermal reformer

```
def ATR_Reactor(inlet, outlet):
    n0CH4, n0H2O, n0H2, n0CO, n0CO2, P0, T0= inlet
    nO2, nCH4, nH2O, nH2, nCO, nCO2, P = outlet

    ch4adj = n0CH4- nO2/2
    cO2adj = n0CO2+ nO2/2

    ksi1 = ch4adj - nCH4
    ksi2 = nCO2 - cO2adj
    ntot = nCH4 + nH2O + nH2 + nCO + nCO2

    T = 1600

    eq1 = K_smr(T)*((nCH4/ntot) * (nH2O/ntot )) - (((nCO/ntot) * (nH2/ntot) ** 3))

    eq2 = K_wgsr(T)*((nCO/ntot) * (nH2O/ntot)) - (((nCO2/ntot) * (nH2/ntot)))

    eq3 = nH2O - n0H2O + ksi1 + ksi2 - nO2

    eq4 = nH2 - n0H2 - 3 * ksi1 - ksi2

    eq5 = nCO - n0CO - ksi1 + ksi2

    eq6 = np.dot(inlet[:5],enthalpy(const, T0)) -
    np.dot(outlet[1:6],enthalpy(const, T)) + nO2*22.7

    eq7 = P-P0

    balances = np.array([eq1,eq2,eq3,eq4,eq5,eq6,eq7])

    return balances
```

A.6 Initial model

Author: Yoonsik Oh

Project: SUBPRO Blue hydrogen

```
from enthalpy import const, enthalpy, enthalpy_heavy, heavy_const
from Keq import K_smr, K_wgsr
import scipy.optimize as opt
import autograd.numpy as np
from autograd import grad, jacobian

# inlet/outlet = [molar flow, P, T/Q/nO2]
C = 12
H = 1.
O = 16
# molar flow component order = CH4, H2O, H2, CO, CO2
Mm = np.array([C * 1 + H * 4, H * 2 + O, H * 2, C + O, C + O * 2]) # This
is to check the conservation of mass
# molar flow component order = CH4, H2O, H2, CO, CO2, C2H6, C3H8, n-C4H10,
i-C4H10, C5H12

Mm_heavy = np.array(
    [C * 1 + H * 4, H * 2 + O, H * 2, C + O, C + O * 2, C * 2 + H * 6, C *
    3 + H * 8, C * 4 + H * 10, C * 4 + H * 10,
    C * 5 + H * 12])

inletcomposition = np.array([78.24, 0.0, 0.0, 0.0, 1.34, 6.10, 6.7, 2.48,
1.41, 3.7]) / 100 # [Sm^3/h]
inletstream_kg = 4000 * 0.829
avgMm = np.dot(Mm_heavy, inletcomposition)
inletstream_mol = inletstream_kg / avgMm

inletstream = np.multiply(inletstream_mol, inletcomposition)
other = np.array([30.0, 311.0])

inletstream = np.append(inletstream, other) # This is the given values
```

```

guess = np.array([400, 30.0, 10000,  # n2
                  100, 400, 0.0, 0.0, 1, 30.0, 400.0,  # n3
                  100, 400, 0.0, 0.0, 1, 30.0, 4000.0,  # n4
                  200, 400, 100, 1, 10, 30.0, 700.0,  # n5
                  200, 300, 100.0, 1, 10, 30.0, 20000.0,  # n6
                  10, 300, 500.0, 100.0, 1.0, 30.0, -5000,  # n7
                  1, 200, 500.0, 100.0, 1.0, 30.0, 30,  # n8
                  1, 200, 500.0, 100.0, 1.0, 30.0, 200,  # n9
                  1, 10, 500.0, 200, 1.0, 30.0, -6000.0,  # n10
                  1, 10, 500.0, 10.0, 1.0, 30.0,  # n11
                  1, 1.0, 500.0, 10.0, 100, 30.0, 313.0,  # n12
                  30.0, 313.0,  # n13
                  1.0, 1.0, 500.0, 1.0, 1.0, 30.0, 313.0,  # n14
                  30.0, 313.0])  # n15

splitratio = np.array([[1.0, 0.001, 1.0, 1.0, 1.0], [0.01, 0.01, 0.999,
0.01, 0.01]])

```

```

# Input is given, Q and output is the initial guess values
def system(input, output, f):
    # Defining and unpacking all the variables
    n1CH4, n1H2O, n1H2, n1CO, n1CO2, \
    n1C2H6, n1C3H8, n1i_C4H10, n1n_C4H10, n1_C5, P1, T1 = input
    f1, f2 = f
    # Note: on stream 8, we set the reactor temperature depending on O2
    stream
    n2H2O, P2, Q1, \
    n3CH4, n3H2O, n3H2, n3CO, n3CO2, P3, T3, \
    n4CH4, n4H2O, n4H2, n4CO, n4CO2, P4, Q2, \
    n5CH4, n5H2O, n5H2, n5CO, n5CO2, P5, T5, \
    n6CH4, n6H2O, n6H2, n6CO, n6CO2, P6, Q3, \
    n7CH4, n7H2O, n7H2, n7CO, n7CO2, P7, Q4, \
    n8CH4, n8H2O, n8H2, n8CO, n8CO2, P8, nO2, \

```



```

n9CH4, n9H2O, n9H2, n9CO, n9CO2, P9, T9, \
n10CH4, n10H2O, n10H2, n10CO, n10CO2, P10, Q5, \
n11CH4, n11H2O, n11H2, n11CO, n11CO2, P11, \
n12CH4, n12H2O, n12H2, n12CO, n12CO2, P12, T12, \
P13, T13, \
n14CH4, n14H2O, n14H2, n14CO, n14CO2, P14, T14, \
P15, T15 = output

```

Defining variables

```

n2 = n2H2O
n5 = n5CH4 + n5H2O + n5H2 + n5CO + n5CO2
n7 = n7CH4 + n7H2O + n7H2 + n7CO + n7CO2
n8 = n8CH4 + n8H2O + n8H2 + n8CO + n8CO2
n10 = n10CH4 + n10H2O + n10H2 + n10CO + n10CO2

nCarbon = n1CH4 + n1C2H6 + n1C3H8 + n1n_C4H10 + n1i_C4H10 + n1_C5

```

BLOCK 1: mixing purified natural gas and steam and setting S/C to 2.5

stcr = 2.5 # steam to carbon ratio (molar ratio)

```
eq0 = n3H2O - stcr * nCarbon
```

```
eq1 = n3CH4 - n1CH4
```

```
eq2 = n3H2O - n2H2O - n1H2O
```

```
eq3 = n3H2 - n1H2
```

```
eq4 = n3CO - n1CO
```

```
eq5 = n3CO2 - n1CO2
```

```
eq6 = P2 - P1
```

```
eq7 = P3 - P2
```

```
n3stream = output[3:8]
```

```
n3stream = np.append(n3stream, input[5:10])
```

```
T2 = 423
```

```

# eq8 = np.dot(input[:10], cp(cp_const, T1)) * 8.314 * (T1 - 298) + n2
#       * 8.314 * single_cp(H2Ocoeff, T2) * (T2 - 298) \
#       - np.dot(n3stream, cp(cp_const, T3)) * 8.314 * (T3 - 298)

```

```

eq8 = np.dot(input[:10], enthalpy_heavy(heavy_const, T1)) + \
      np.dot([0, n2H2O, 0, 0, 0, 0, 0, 0, 0, 0],

```

```

enthalpy_heavy(heavy_const, T2)) - \
np.dot(n3stream, enthalpy_heavy(heavy_const, T3))

# BLOCK 2: Pre-prereformer heat exchanger
T4 = 693
eq9 = n4CH4 - n3CH4
eq10 = n4H2O - n3H2O
eq11 = n4H2 - n3H2
eq12 = n4CO - n3CO
eq13 = n4CO2 - n3CO2
n4stream = output[10:15]
n4stream = np.append(n4stream, input[5:10])
eq14 = np.dot(n3stream, enthalpy_heavy(heavy_const, T3)) - \
      np.dot(n4stream, enthalpy_heavy(heavy_const, T4)) + Q1
eq15 = P4 - P3

# BLOCK 3: Prereformer (Removing all heavier carbons to methane, (full
conversion))
pr_ksi1 = n1C2H6
pr_ksi2 = n1C3H8
pr_ksi3 = n1n_C4H10
pr_ksi4 = n1i_C4H10
pr_ksi5 = n1_C5
pr_ksi6 = n4CH4 - n5CH4
pr_ksi7 = n5CO2 - n4CO2

eq16 = n5H2O - n4H2O + 2 * pr_ksi1 + 3 * pr_ksi2 + 4 * pr_ksi3 + 4 *
pr_ksi4 + 5 * pr_ksi5 + pr_ksi6 + pr_ksi7
eq17 = n5H2 - n4H2 - 5 * pr_ksi1 - 7 * pr_ksi2 - 9 * pr_ksi3 - 9 *
pr_ksi4 - 11 * pr_ksi5 - 3 * pr_ksi6 - pr_ksi7
eq18 = n5CO - n4CO - 2 * pr_ksi1 - 3 * pr_ksi2 - 4 * pr_ksi3 - 4 *
pr_ksi4 - 5 * pr_ksi5 - pr_ksi6 + pr_ksi7
eq19 = K_smr(T5) * ((n5CH4 / n5) * (n5H2O / n5)) - (((n5CO / n5) *
(n5H2 / n5) ** 3))
eq20 = K_wgsr(T5) * ((n5CO / n5) * (n5H2O / n5)) - (((n5CO2 / n5) *
(n5H2 / n5)))

```

```

eq21 = np.dot(n4stream, enthalpy_heavy(heavy_const, T4)) -
np.dot(output[17:22], enthalpy(const, T5))
eq22 = P5 - P4

```

BLOCK 4: Pre-GHR Heat Exchanger

```

T6 = 973
eq23 = n6CH4 - n5CH4
eq24 = n6H2O - n5H2O
eq25 = n6H2 - n5H2
eq26 = n6CO - n5CO
eq27 = n6CO2 - n5CO2
eq28 = np.dot(output[17:22], enthalpy(const, T5)) - \
      np.dot(output[24:29], enthalpy(const, T6)) + Q2
eq29 = P6 - P5

```

BLOCK 5: GHR

```

T7 = 1073
ghr_ksi1 = n6CH4 - n7CH4
ghr_ksi2 = n7CO2 - n6CO2
eq30 = K_smr(T7) * ((n7CH4 / n7) * (n7H2O / n7)) - (((n7CO / n7) *
(n7H2 / n7) ** 3))
eq31 = K_wgsr(T7) * ((n7CO / n7) * (n7H2O / n7)) - (((n7CO2 / n7) *
(n7H2 / n7)))
eq32 = n7H2O - n6H2O + ghr_ksi1 + ghr_ksi2
eq33 = n7H2 - n6H2 - 3 * ghr_ksi1 - ghr_ksi2
eq34 = n7CO - n6CO - ghr_ksi1 + ghr_ksi2
eq35 = np.dot(output[24:29], enthalpy(const, T6)) -
np.dot(output[31:36], enthalpy(const, T7)) + Q3
eq36 = P7 - P6

```

BLOCK 6: ATR

```

atr_ksi1 = n7CH4 - n8CH4 - nO2 / 2
atr_ksi2 = n8CO2 - n7CO2 - nO2 / 2

T8 = 1323

```

```

eq37 = K_smr(T8) * ((n8CH4 / n8) * (n8H2O / n8)) - ((n8CO / n8) * (n8H2
/ n8) ** 3)
eq38 = K_wgsr(T8) * ((n8CO / n8) * (n8H2O / n8)) - ((n8CO2 / n8) *
(n8H2 / n8))
eq39 = n8H2O - n7H2O + atr_ksi1 + atr_ksi2 - nO2
eq40 = n8H2 - n7H2 - 3 * atr_ksi1 - atr_ksi2
eq41 = n8CO - n7CO - atr_ksi1 + atr_ksi2
eq42 = np.dot(output[31:36], enthalpy(const, T7)) -
np.dot(output[38:43], enthalpy(const, T8)) + nO2 * 23.7

eq43 = P8 - P7

#eq37, eq38, eq39, eq40, eq41, eq42, eq43 =
ATR_Reactor(np.array([n7CH4,n7H2O,n7H2, n7CO, n7CO2, P7, T7]),
#np.array([nO2,n8CH4, n8H2O, n8H2, n8CO, n8CO2, P8]))

# BLOCK 7: Post-ATR heat exchanger for making the model make sense by
setting the
# post temperature after ATR as the duty required for heating up GHR
eq44 = n9CH4 - n8CH4
eq45 = n9H2O - n8H2O
eq46 = n9H2 - n8H2
eq47 = n9CO - n8CO
eq48 = n9CO2 - n8CO2
eq49 = np.dot(output[38:43], enthalpy(const, T8)) - \
np.dot(output[45:50], enthalpy(const, T9)) + Q3
eq50 = P9 - P8

# BLOCK 8: Isothermal temperature shift reactor
T10 = 523 # Setting the reactor temperature and solving for the heat
required instead
itsr_ksi1 = n9H2O - n10H2O

eq51 = n10CH4 - n9CH4
eq52 = n10H2 - n9H2 - itsr_ksi1

```

```

eq53 = n10C0 - n9C0 + itsr_ksi1
eq54 = n10C02 - n9C02 - itsr_ksi1
eq55 = K_wgsr(T10) * ((n10C0 / n10) * (n10H2O / n10)) - ((n10C02 / n10)
* (n10H2 / n10))
eq56 = np.dot(output[45:50], enthalpy(const, T9)) - \
      np.dot(output[52:57], enthalpy(const, T10)) + Q4
eq57 = P10 - P9

```

BLOCK 9: Precondensate heat exchanger, same as ITSR, the temperature is set and we solve for heat gained

```

T11 = 313
eq58 = n11CH4 - n10CH4
eq59 = n11H2O - n10H2O
eq60 = n11H2 - n10H2
eq61 = n11C0 - n10C0
eq62 = n11C02 - n10C02
eq63 = np.dot(output[52:57], enthalpy(const, T10)) - \
      np.dot(output[59:64], enthalpy(const, T11)) + Q5
eq64 = P11 - P10

```

BLOCK 10: Condensate

```

eq65 = n12CH4 - f1[0] * n11CH4
eq66 = n12H2O - f1[1] * n11H2O
eq67 = n12H2 - f1[2] * n11H2
eq68 = n12C0 - f1[3] * n11C0
eq69 = n12C02 - f1[4] * n11C02
eq70 = P12 - P11
eq71 = T12 - T11

```

```

eq72 = P13 - P11
eq73 = T13 - T11

```

BLOCK 11 PSA

```

eq74 = n14CH4 - f2[0] * n12CH4
eq75 = n14H2O - f2[1] * n12H2O
eq76 = n14H2 - f2[2] * n12H2

```

```

eq77 = n14C0 - f2[3] * n12C0
eq78 = n14C02 - f2[4] * n12C02
eq79 = P14 - P12
eq80 = T14 - T12

eq81 = P15 - P12
eq82 = T15 - T12

system = np.array([eq0, eq1, eq2, eq3, eq4, eq5, eq6, eq7, eq8, eq9,
eq10, eq11, eq12, eq13, eq14, eq15,
                    eq16, eq17, eq18, eq19, eq20, eq21, eq22, eq23,
                    eq24, eq25, eq26, eq27, eq28, eq29,
                    eq30, eq31, eq32, eq33, eq34, eq35, eq36, eq37,
                    eq38, eq39, eq40, eq41, eq42, eq43,
                    eq44, eq45, eq46, eq47, eq48, eq49, eq50, eq51,
                    eq52, eq53, eq54, eq55, eq56, eq57,
                    eq58, eq59, eq60, eq61, eq62, eq63, eq64, eq65,
                    eq66, eq67, eq68, eq69, eq70, eq71,
                    eq72, eq73, eq74, eq75, eq76, eq77, eq78, eq79,
                    eq80, eq81, eq82])

return system

def function(guess):
    return system(inletstream, guess, splitratio)

def solve(f, guess, grad_function):
    sol = opt.fsolve(f, guess)
    # print(sol)
    return sol

grad_function = jacobian(function)

# sol = opt.fsolve(function, guess, fprime=grad_function)

```

```
# print("error:",function(sol))
sol = solve(function, guess, grad_function)
```

A.7 Mixing temperature

```
def mixingTemp(input,guess):
    n1CH4, n1H2O, n1H2, n1CO, n1CO2, n1C2H6, n1C3H8, n1i_C4H10, n1n_C4H10,
    n1_C5, P1, T1, n2H2O, T2 = input
    n3stream = input[:10]+np.array([0, n2H2O, 0, 0, 0, 0, 0, 0, 0, 0])
    T3 = guess[0]

    eq1 = np.dot(input[:10], enthalpy_heavy(heavy_const, T1)) + \
          np.dot(np.array([0, n2H2O, 0, 0, 0, 0, 0, 0, 0, 0]),
                  enthalpy_heavy(heavy_const, T2)) - \
          np.dot(n3stream, enthalpy_heavy(heavy_const, T3))

    return eq1

def T3function(outlet): # Using this function in fsolve
    return mixingTemp(inletstream,outlet)
```

A.8 Prereformer

```
def PR_Reactor(inlet, outlet):
    n4CH4, n4H2O, n4H2, n4CO, n4CO2, n1C2H6, n1C3H8, n1n_C4H10, n1i_C4H10,
    n1_C5,T4 = inlet
    n5CH4, n5H2O, n5H2, n5CO, n5CO2, T5 = outlet

    n4stream = inlet[:10]
    n5 = n5CH4 + n5H2O + n5H2 + n5CO + n5CO2

    pr_ksi1 = n1C2H6
    pr_ksi2 = n1C3H8
    pr_ksi3 = n1n_C4H10
    pr_ksi4 = n1i_C4H10
```

```

pr_ksi5 = n1_C5
pr_ksi6 = n4CH4 - n5CH4
pr_ksi7 = n5CO2 - n4CO2

eq1 = n5H2O - n4H2O + 2 * pr_ksi1 + 3 * pr_ksi2 + 4 * pr_ksi3 + 4 *
pr_ksi4 + 5 * pr_ksi5 + pr_ksi6 + pr_ksi7
eq2 = n5H2 - n4H2 - 5 * pr_ksi1 - 7 * pr_ksi2 - 9 * pr_ksi3 - 9 *
pr_ksi4 - 11 * pr_ksi5 - 3 * pr_ksi6 - pr_ksi7
eq3 = n5CO - n4CO - 2 * pr_ksi1 - 3 * pr_ksi2 - 4 * pr_ksi3 - 4 *
pr_ksi4 - 5 * pr_ksi5 - pr_ksi6 + pr_ksi7
eq4 = K_smr(T5) * ((n5CH4 / n5) * (n5H2O / n5)) - (((n5CO / n5) * (n5H2
/ n5) ** 3))
eq5 = K_wgsr(T5) * ((n5CO / n5) * (n5H2O / n5)) - (((n5CO2 / n5) * (n5H2
/ n5)))
eq6 = np.dot(n4stream, enthalpy_heavy(heavy_const, T4)) -
np.dot(outlet[:5], enthalpy(const, T5))

return np.array([eq1,eq2,eq3,eq4,eq5,eq6])

```

A.9 Post ATR temperature

```

def postATR_temp(input, output):
    n8CH4, n8H2O, n8H2, n8CO, n8CO2, \
    n9CH4, n9H2O, n9H2, n9CO, n9CO2, T8, Q3 = input

    T9 = output

    stream8 = np.array([n8CH4, n8H2O, n8H2, n8CO, n8CO2])
    stream9 = np.array([n9CH4, n9H2O, n9H2, n9CO, n9CO2])

    eq1 = np.dot(stream8, enthalpy(const, T8)) - \
        np.dot(stream9, enthalpy(const, T9)) + Q3

    return eq1

```


A.10 Isothermal temperature shift reactor

```
def itsr(input, output):
    n9CH4, n9H2O, n9H2, n9CO, n9CO2, T9 = input
    n10CH4, n10H2O, n10H2, n10CO, n10CO2, Q4 = output
    n10 = n10CH4 + n10H2O + n10H2 + n10CO + n10CO2
    stream9 = np.array([n9CH4, n9H2O, n9H2, n9CO, n9CO2])
    stream10 = np.array([n10CH4, n10H2O, n10H2, n10CO, n10CO2])

    T10 = 523
    # Setting the reactor temperature and solving for the heat required instead
    itsr_ksi1 = n9H2O - n10H2O

    eq1 = n10CH4 - n9CH4
    eq2 = n10H2 - n9H2 - itsr_ksi1
    eq3 = n10CO - n9CO + itsr_ksi1
    eq4 = n10CO2 - n9CO2 - itsr_ksi1
    eq5 = K_wgsr(T10) * ((n10CO / n10) * (n10H2O / n10))
    - (((n10CO2 / n10) * (n10H2 / n10)))
    eq6 = np.dot(stream9, enthalpy(const, T9)) - \
        np.dot(stream10, enthalpy(const, T10)) + Q4

    return np.array([eq1, eq2, eq3, eq4, eq5, eq6])
```

A.11 New improved model

```
# Author: Yoonsik Oh
# Project: SUBPRO Blue hydrogen

from enthalpy import const, enthalpy, enthalpy_heavy, heavy_const
from Keq import K_smr, K_wgsr
import scipy.optimize as opt
import autograd.numpy as np
from autograd import grad, jacobian
from atr import ATR_Reactor
from mixing import mixingTemp
```

```

from prereformer import PR_Reactor
from ghr import GHR_Reactor
from postATR import postATR_temp
from itsr import itsr

# inlet/outlet = [molar flow, T, P]
C = 12
H = 1.
O = 16
# molar flow component order = CH4, H2O, H2, CO, CO2
Mm = np.array([C * 1 + H * 4, H * 2 + O, H * 2, C + O, C + O * 2]) # This
is to check the conservation of mass
# molar flow component order = CH4, H2O, H2, CO, CO2 C2H6 C3H8
n-C4H10 i-C4H10 C5H12

Mm_heavy = np.array(
    [C * 1 + H * 4, H * 2 + O, H * 2, C + O, C + O * 2, C * 2 + H * 6, C *
    3 + H * 8, C * 4 + H * 10, C * 4 + H * 10,
    C * 5 + H * 12])

inletcomposition = np.array([78.24, 0.0, 0.0, 0.0, 1.34, 6.10, 6.7, 2.48,
1.41, 3.7]) / 100 # [Sm3/h]
inletstream_kg = 4000 * 0.829
avgMm = np.dot(Mm_heavy, inletcomposition)
inletstream_mol = inletstream_kg / avgMm

inletstream = np.multiply(inletstream_mol, inletcomposition)
other = np.array([30.0, 311.0])

inletstream = np.append(inletstream, other) # This is the given values

splitratio = np.array([[1.0, 0.001, 1.0, 1.0, 1.0], [0.01, 0.01, 0.999,
0.01, 0.01]])

```

```

# Input is given, Q and output is the initial guess values
def system(input, f):
    # -----
    # Defining and unpacking all the variables
    # -----
    n1CH4, n1H2O, n1H2, n1CO, n1CO2, \
    n1C2H6, n1C3H8, n1i_C4H10, n1n_C4H10, n1_C5, P1, T1 = input
    f1, f2 = f
    # -----
    # Defining variables
    # -----

    nCarbon = n1CH4 + n1C2H6 + n1C3H8 + n1n_C4H10 + n1i_C4H10 + n1_C5
    stcr = 2.5 # steam to carbon ratio (molar ratio)
    # -----
    # BLOCK 1: mixing purified natural gas and steam and setting S/C to
    2.5
    # -----

    n2H2O = stcr * nCarbon
    n3CH4 = n1CH4
    n3H2O = n2H2O + n1H2O
    n3H2 = n1H2
    n3CO = n1CO
    n3CO2 = n1CO2
    P2 = P1
    P3 = P2

    T2 = 423
    inlet1 = input
    inlet1 = np.append(inlet1, np.array([n2H2O, T2]))

    def T3function(guess):
        return mixingTemp(inlet1, guess)

```

```

sol1 = opt.fsolve(T3function, np.array([T2]))

T3 = sol1[0]

# -----
# BLOCK 2: Pre-prereformer heat exchanger
# -----

T4 = 693
n4CH4 = n3CH4
n4H2O = n3H2O
n4H2 = n3H2
n4CO = n3CO
n4CO2 = n3CO2

n4stream = np.array([n4CH4, n4H2O, n4H2, n4CO, n4CO2])
n4stream = np.append(n4stream, input[5:10])
n3stream = n4stream
Q1 = np.dot(n4stream, enthalpy_heavy(heavy_const, T4)) -
np.dot(n3stream, enthalpy_heavy(heavy_const, T3))
P4 = P3

# -----
# BLOCK 3: Pre-reformer (Removing all heavier carbons to methane,
# (full conversion))
# -----

stream4 = np.array([n4CH4, n4H2O, n4H2, n4CO, n4CO2])
stream4 = np.append(stream4, input[5:10])
stream4 = np.append(stream4, T4)
guess3 = np.array([100, 500, 100, 1, 100, 700])

def PR_function(guess):
    return PR_Reactor(stream4, guess)

sol3 = opt.fsolve(PR_function, guess3)

n5CH4, n5H2O, n5H2, n5CO, n5CO2, T5 = sol3

```

```

stream5 = np.array([n5CH4, n5H2O, n5H2, n5CO, n5CO2])
P5 = P4

# -----
# BLOCK 4: Pre-GHR Heat Exchanger
# -----

T6 = 973
n6CH4 = n5CH4
n6H2O = n5H2O
n6H2 = n5H2
n6CO = n5CO
n6CO2 = n5CO2

stream6 = np.array([n6CH4, n6H2O, n6H2, n6CO, n6CO2])
Q2 = np.dot(stream6, enthalpy(const, T6)) - np.dot(stream5,
enthalpy(const, T5))
P6 = P5

# -----
# BLOCK 5: GHR
# -----

guess5 = np.array([100, 500, 500, 100, 100, 28000.0])
T7 = 1073
input5 = np.append(stream6, np.array([T6]))

def GHR_function(guess):
    return GHR_Reactor(input5, guess)

sol5 = opt.fsolve(GHR_function, guess5)

n7CH4, n7H2O, n7H2, n7CO, n7CO2, Q3 = sol5
P7 = P6

# -----

```

```

# BLOCK 6: ATR
# -----
stream7 = np.array([n7CH4, n7H2O, n7H2, n7CO, n7CO2])
guess6 = np.array([100,100, 500, 500, 100, 100, 30])
inlet6 = np.append(stream7, np.array([30, T7]))
T8 = 1600

def ATR_function(guess):
    return ATR_Reactor(inlet6, guess)

sol6 = opt.fsolve(ATR_function, guess6)

nO2, n8CH4, n8H2O, n8H2, n8CO, n8CO2, P8 = sol6
stream8 = np.array([n8CH4, n8H2O, n8H2, n8CO, n8CO2])

# -----
# BLOCK 7: Post-ATR heat exchanger for making the model make sense by
# setting the
# post temperature after ATR as the duty required for heating up GHR
# -----
n9CH4 = n8CH4
n9H2O = n8H2O
n9H2 = n8H2
n9CO = n8CO
n9CO2 = n8CO2
stream9 = np.array([n9CH4, n9H2O, n9H2, n9CO, n9CO2])
inlet7 = np.append(stream8, stream9)
inlet7 = np.append(inlet7, np.array([T8, -Q3]))

def postAtr_function(guess):
    return postATR_temp(inlet7, guess)

guess7 = np.array([400])

sol7 = opt.fsolve(postAtr_function, guess7)

```

```

T9 = sol7[0]

P9 = P8

# -----
# BLOCK 8: Isothermal temperature shift reactor
# -----
T10 = 523 # Setting the reactor temperature and solving for the heat
required instead

input8 = np.append(stream9, T9)
guess8 = np.array([1, 500, 500, 100, 100, -100000])

def itsr_function(guess):
    return itsr(input8, guess)

sol8 = opt.fsolve(itstr_function, guess8)

n10CH4, n10H2O, n10H2, n10CO, n10CO2, Q4 = sol8

P10 = P9

# -----
# BLOCK 9: Precondensate heat exchanger, same as ITSR, the temperature
is set
# and we solve for heat gained
# -----
T11 = 313
n11CH4 = n10CH4
n11H2O = n10H2O
n11H2 = n10H2
n11CO = n10CO
n11CO2 = n10CO2
stream10 = np.array([n10CH4, n10H2O, n10H2, n10CO, n10CO2])
stream11 = np.array([n11CH4, n11H2O, n11H2, n11CO, n11CO2])

```

```
Q5 = np.dot(stream11, enthalpy(const, T11)) - \
      np.dot(stream10, enthalpy(const, T10))
```

```
P11 = P10
```

```
# -----
```

```
# BLOCK 10: Condensate
```

```
# -----
```

```
n12CH4 = f1[0] * n11CH4
```

```
n12H2O = f1[1] * n11H2O
```

```
n12H2 = f1[2] * n11H2
```

```
n12CO = f1[3] * n11CO
```

```
n12CO2 = f1[4] * n11CO2
```

```
P12 = P11
```

```
T12 = T11
```

```
n13CH4 = (1 - f1[0]) * n11CH4
```

```
n13H2O = (1 - f1[1]) * n11H2O
```

```
n13H2 = (1 - f1[2]) * n11H2
```

```
n13CO = (1 - f1[3]) * n11CO
```

```
n13CO2 = (1 - f1[4]) * n11CO2
```

```
P13 = P11
```

```
T13 = T11
```

```
# -----
```

```
# BLOCK 11 PSA
```

```
# -----
```

```
n14CH4 = f2[0] * n12CH4
```

```
n14H2O = f2[1] * n12H2O
```

```
n14H2 = f2[2] * n12H2
```

```
n14CO = f2[3] * n12CO
```

```
n14CO2 = f2[4] * n12CO2
```

```
P14 = P12
```

```
T14 = T12
```



```

n15CH4 = (1 - f2[0]) * n12CH4
n15H2O = (1 - f2[1]) * n12H2O
n15H2 = (1 - f2[2]) * n12H2
n15CO = (1 - f2[3]) * n12CO
n15CO2 = (1 - f2[4]) * n12CO2
P15 = P12
T15 = T12

# -----
# Return the variables
# -----

system = np.array([n1CH4, n1H2O, n1H2, n1CO, n1CO2, T1, P1,
                   0.0, n2H2O, 0.0, 0.0, 0.0, T2, P2,
                   n3CH4, n3H2O, n3H2, n3CO, n3CO2, T3, P3,
                   n4CH4, n4H2O, n4H2, n4CO, n4CO2, T4, P4,
                   n5CH4, n5H2O, n5H2, n5CO, n5CO2, T5, P5,
                   n6CH4, n6H2O, n6H2, n6CO, n6CO2, T6, P6,
                   n7CH4, n7H2O, n7H2, n7CO, n7CO2, T7, P7,
                   n8CH4, n8H2O, n8H2, n8CO, n8CO2, T8, P8,
                   n9CH4, n9H2O, n9H2, n9CO, n9CO2, T9, P9,
                   n10CH4, n10H2O, n10H2, n10CO, n10CO2, T10, P10,
                   n11CH4, n11H2O, n11H2, n11CO, n11CO2, T11, P11,
                   n12CH4, n12H2O, n12H2, n12CO, n12CO2, T12, P12,
                   n13CH4, n13H2O, n13H2, n13CO, n13CO2, T13, P13,
                   n14CH4, n14H2O, n14H2, n14CO, n14CO2, T14, P14,
                   n15CH4, n15H2O, n15H2, n15CO, n15CO2, T15, P15,
                   nO2, Q1, Q2, Q3, Q4, Q5])

return system

```

```
data = system(inletstream,splitratio)
```

A.12 Stream summary print function

```

def mass(data):
    print('{:<15}{:.4f}'.format("Stream 1 [g/s]: ", Mm_heavy @
    inletstream[:10]))

```

```

print('{:<15}{:.4f}'.format("Stream 2 [g/s]: ", Mm @ data[7:12]))
print('{:<15}{:.4f}'.format("Stream 3 [g/s]: ", Mm @ data[14:19] +
(Mm_heavy[5:] @ inletstream[5:10])))
print('{:<15}{:.4f}'.format("Stream 4 [g/s]: ", Mm @ data[21:26] +
(Mm_heavy[5:] @ inletstream[5:10])))
print('{:<15}{:.4f}'.format("Stream 5 [g/s]: ", Mm @ data[28:33]))
print('{:<15}{:.4f}'.format("Stream 6 [g/s]: ", Mm @ data[35:40]))
print('{:<15}{:.4f}'.format("Stream 7 [g/s]: ", Mm @ data[42:47]))
print('{:<15}{:.4f}'.format("Stream 02 [g/s]: ", data[105] * 32))
print('{:<15}{:.4f}'.format("Stream 8 [g/s]: ", Mm @ data[49:54]))
print('{:<15}{:.4f}'.format("Stream 9 [g/s]: ", Mm @ data[56:61]))
print('{:<15}{:.4f}'.format("Stream 10 [g/s]: ", Mm @ data[63:68]))
print('{:<15}{:.4f}'.format("Stream 11 [g/s]: ", Mm @ data[70:75]))
print('{:<15}{:.4f}'.format("Stream 12 [g/s]: ", Mm @ data[77:82]))
print('{:<15}{:.4f}'.format("Stream 13 [g/s]: ", Mm @ data[84:89]))
print('{:<15}{:.4f}'.format("Stream 14 [g/s]: ", Mm @ data[91:96]))
print('{:<15}{:.4f}'.format("Stream 15 [g/s]: ", Mm @ data[98:103]))

```

A.13 Error output

```

def error_output():
    sol = opt.fsolve(function, guess, fprime=grad_function, xtol=1e-8)
    # print(sol)
    errors = function(sol)
    print(sum(abs(errors)))
    for i in range(len(errors)):
        print("eq", i, ":", errors[i])

```

A.14 Mass balance check function

```

def mass(data):
    print('{:<15}{:.4f}'.format("Stream 1 [g/s]: ", Mm_heavy @
inletstream[:10]))
    print('{:<15}{:.4f}'.format("Stream 2 [g/s]: ", Mm @ data[7:12]))
    print('{:<15}{:.4f}'.format("Stream 3 [g/s]: ", Mm @ data[14:19] +
(Mm_heavy[5:] @ inletstream[5:10])))
    print('{:<15}{:.4f}'.format("Stream 4 [g/s]: ", Mm @ data[21:26] +

```

```

(Mm_heavy[5:] @ inletstream[5:10]))
print('{:<15}{:.4f}'.format("Stream 5 [g/s]: ", Mm @ data[28:33]))
print('{:<15}{:.4f}'.format("Stream 6 [g/s]: ", Mm @ data[35:40]))
print('{:<15}{:.4f}'.format("Stream 7 [g/s]: ", Mm @ data[42:47]))
print('{:<15}{:.4f}'.format("Stream 02 [g/s]: ", data[105] * 32))
print('{:<15}{:.4f}'.format("Stream 8 [g/s]: ", Mm @ data[49:54]))
print('{:<15}{:.4f}'.format("Stream 9 [g/s]: ", Mm @ data[56:61]))
print('{:<15}{:.4f}'.format("Stream 10 [g/s]: ", Mm @ data[63:68]))
print('{:<15}{:.4f}'.format("Stream 11 [g/s]: ", Mm @ data[70:75]))
print('{:<15}{:.4f}'.format("Stream 12 [g/s]: ", Mm @ data[77:82]))
print('{:<15}{:.4f}'.format("Stream 13 [g/s]: ", Mm @ data[84:89]))
print('{:<15}{:.4f}'.format("Stream 14 [g/s]: ", Mm @ data[91:96]))
print('{:<15}{:.4f}'.format("Stream 15 [g/s]: ", Mm @ data[98:103]))

```

A.15 Composition print function

```

def composition(data):
    n1 = np.sum(data[:5]+inletstream[5:10])
    n2 = np.sum(data[7:12])
    n3 = np.sum(data[14:19]+inletstream[5:10])
    n4 = np.sum(data[21:26]+inletstream[5:10])
    n5 = np.sum(data[28:33])
    n6 = np.sum(data[35:40])
    n7 = np.sum(data[42:47])
    n8 = np.sum(data[49:54])
    n9 = np.sum(data[56:61])
    n10 = np.sum(data[63:68])
    n11 = np.sum(data[70:75])
    n12 = np.sum(data[77:82])
    n13 = np.sum(data[84:89])
    n14 = np.sum(data[91:96])
    n15 = np.sum(data[98:103])

    print("-----")
    print("-----")
    print("-----")

```



```

        inletstream[7]/n4, inletstream[8]/n4,
        inletstream[9]/n4))

print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 5:", data[28]/n5, data[29]/n5, data[30]/n5,
    data[31]/n5, data[32]/n5, data[33], data[34],
    0, 0, 0, 0, 0))

print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 6:", data[35]/n6, data[36]/n6, data[37]/n6,
    data[38]/n6, data[39]/n6, data[40], data[41],
    0, 0, 0, 0, 0))

print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 7:", data[42]/n7, data[43]/n7, data[44]/n7,
    data[45]/n7, data[46]/n7, data[47], data[48],
    0, 0, 0, 0, 0))

print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 8:", data[49]/n8, data[50]/n8, data[51]/n8,
    data[52]/n8, data[53]/n8, data[54], data[55],
    0, 0, 0, 0, 0))

print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 9:", data[56]/n9, data[57]/n9, data[58]/n9,
    data[59]/n9, data[60]/n9, data[61], data[62],
    0, 0, 0, 0, 0))

print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 10:", data[63]/n10, data[64]/n10,

```

```

        data[65]/n10, data[66]/n10, data[67]/n10, data[68], data[69],
        0, 0, 0, 0, 0))

print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 11:", data[70]/n11, data[71]/n11,
    data[72]/n11, data[73]/n11, data[74]/n11, data[75], data[76],
    0, 0, 0, 0, 0))

print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 12:", data[77]/n12, data[78]/n12,
    data[79]/n12, data[80]/n12, data[81]/n12, data[82], data[83],
    0, 0, 0, 0, 0))


print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 13:", data[84]/n13, data[85]/n13,
    data[86]/n13, data[87]/n13, data[88]/n13, data[89], data[90],
    0, 0, 0, 0, 0))


print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 14:", data[91]/n14, data[92]/n14,
    data[93]/n14, data[94]/n14, data[95]/n14, data[96], data[97],
    0, 0, 0, 0, 0))


print(
    '{:<12}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.2f}{:<1
    .2f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}{:<10.4f}'
    .format("Stream 15:", data[98]/n15, data[99]/n15,
    data[100]/n15, data[101]/n15, data[102]/n15, data[103],
    data[104],
    0, 0, 0, 0, 0))


```

B Hysys simulation with pure oxygen


1				Case Name: case_28.06.22_TemporarilyDone.hsc				
2		NORWEGIAN UNIVERSITY OF Bedford, MA USA			Unit Set: NewUser			
3					Date/Time: Fri Jul 15 14:32:16 2022			
4								
5								
6								
7	Workbook: Case (Main)							
8								
9								
10	Material Streams				Fluid Pkg:	All		
11	Name	H2O	NG	1	2	3		
12	Vapour Fraction	0.0000	1.0000 *	0.3181	1.0000	1.0000		
13	Temperature (C)	150.0 *	38.00 *	117.9	420.0 *	382.4		
14	Pressure (kPa)	3000 *	3000 *	3000	3000 *	3000		
15	Molar Flow (kgmole/h)	2368 *	1000 *	3368	3368	3466		
16	Mass Flow (kg/h)	4.265e+004	1.687e+004	5.952e+004	5.952e+004	5.952e+004		
17	Liquid Volume Flow (m3/h)	42.74	54.89	97.63	97.63	101.3		
18	Heat Flow (kJ/h)	-6.541e+008	-7.604e+007	-7.302e+008	-5.981e+008	-5.981e+008		
19	Name	4	5	6	7	8		
20	Vapour Fraction	0.0000	1.0000	1.0000	0.0000	1.0000		
21	Temperature (C)	382.4	480.0 *	700.0 *	700.0	1050		
22	Pressure (kPa)	3000	3000 *	3000	3000	3000		
23	Molar Flow (kgmole/h)	0.0000	3466	4096	0.0000	5428		
24	Mass Flow (kg/h)	0.0000	5.952e+004	5.952e+004	0.0000	7.410e+004		
25	Liquid Volume Flow (m3/h)	0.0000	101.3	122.9	0.0000	147.9		
26	Heat Flow (kJ/h)	0.0000	-5.830e+008	-4.826e+008	0.0000	-4.796e+008		
27	Name	9	O2	10	11	12		
28	Vapour Fraction	0.0000	1.0000	1.0000	1.0000	0.0000		
29	Temperature (C)	1050	250.0 *	546.8 *	250.0 *	250.0		
30	Pressure (kPa)	3000	3000 *	3000	3000	3000		
31	Molar Flow (kgmole/h)	0.0000	455.6 *	5428	5428	0.0000		
32	Mass Flow (kg/h)	0.0000	1.458e+004	7.410e+004	7.410e+004	0.0000		
33	Liquid Volume Flow (m3/h)	0.0000	12.82	147.9	168.2	0.0000		
34	Heat Flow (kJ/h)	0.0000	3.069e+006	-5.800e+008	-6.624e+008	0.0000		
35	Name	13	14	15	H2	Tailgas		
36	Vapour Fraction	0.7723	1.0000	0.0000	1.0000	0.9971		
37	Temperature (C)	40.00 *	40.00	40.00	40.00 *	59.06		
38	Pressure (kPa)	3000 *	3000	3000	3000 *	3000		
39	Molar Flow (kgmole/h)	5428	4192	1236	3134	1058		
40	Mass Flow (kg/h)	7.410e+004	5.175e+004	2.234e+004	6318	4.543e+004		
41	Liquid Volume Flow (m3/h)	168.2	145.8	22.42	90.44	55.37		
42	Heat Flow (kJ/h)	-7.529e+008	-4.003e+008	-3.526e+008	1.342e+006	-4.017e+008		
43								
44								
45								
46								
47								
48								
49								
50								
51								
52								
53								
54								
55								
56								
57								
58								
59								
60								
61								
62								
63	Aspen Technology Inc.		Aspen HYSYS Version 11		Page 1 of 4			
Licensed to: NORWEGIAN UNIVERSITY OF								
* Specified by user.								

1	<div></div> <div>NORWEGIAN UNIVERSITY OF Bedford, MA USA</div>			Case Name: case_28.06.22_TemporarilyDone.hsc		
2						
3				Unit Set: NewUser		
4				Date/Time: Fri Jul 15 14:32:16 2022		
5						
6	Workbook: Case (Main) (continued)					
7						
8						
9						
10	Compositions				Fluid Pkg: All	
11	Name	H2O	NG	1	2	3
12	Comp Mole Frac (Methane)	0.0000 *	0.9470 *	0.2812	0.2812	0.2857
13	Comp Mole Frac (Ethane)	0.0000 *	0.0420 *	0.0125	0.0125	0.0000
14	Comp Mole Frac (Propane)	0.0000 *	0.0020 *	0.0006	0.0006	0.0000
15	Comp Mole Frac (n-Butane)	0.0000 *	0.0002 *	0.0001	0.0001	0.0000
16	Comp Mole Frac (CO2)	0.0000 *	0.0030 *	0.0009	0.0009	0.0152
17	Comp Mole Frac (Oxygen)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000
18	Comp Mole Frac (Nitrogen)	0.0000 *	0.0050 *	0.0015	0.0015	0.0014
19	Comp Mole Frac (H2S)	0.0000 *	0.0000 *	0.0000	0.0000	0.0000
20	Comp Mole Frac (H2O)	1.0000 *	0.0000 *	0.7030	0.7030	0.6544
21	Comp Mole Frac (H2O2)	0.0000 *	0.0000 *	0.0000	0.0000	0.0000
22	Comp Mole Frac (CO)	0.0000 *	0.0000 *	0.0000	0.0000	0.0001
23	Comp Mole Frac (Hydrogen)	0.0000 *	0.0002 *	0.0001	0.0001	0.0432
24	Comp Mole Frac (i-Butane)	0.0000 *	0.0002 *	0.0001	0.0001	0.0000
25	Comp Mole Frac (n-Pentane)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000
26	Comp Mole Frac (i-Pentane)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000
27	Comp Mole Frac (n-Hexane)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000
28	Comp Mole Frac (HCN)	0.0000 *	0.0000 *	0.0000	0.0000	0.0000
29	Comp Mole Frac (Ammonia)	0.0000 *	0.0000 *	0.0000	0.0000	0.0002
30	Name	4	5	6	7	8
31	Comp Mole Frac (Methane)	0.2857	0.2857	0.1650	0.1650	0.0018
32	Comp Mole Frac (Ethane)	0.0000	0.0000	0.0000	0.0000	0.0000
33	Comp Mole Frac (Propane)	0.0000	0.0000	0.0000	0.0000	0.0000
34	Comp Mole Frac (n-Butane)	0.0000	0.0000	0.0000	0.0000	0.0000
35	Comp Mole Frac (CO2)	0.0152	0.0152	0.0616	0.0616	0.0564
36	Comp Mole Frac (Oxygen)	0.0000	0.0000	0.0000	0.0000	0.0000
37	Comp Mole Frac (Nitrogen)	0.0014	0.0014	0.0012	0.0012	0.0009
38	Comp Mole Frac (H2S)	0.0000	0.0000	0.0000	0.0000	0.0000
39	Comp Mole Frac (H2O)	0.6544	0.6544	0.4282	0.4282	0.3583
40	Comp Mole Frac (H2O2)	0.0000	0.0000	0.0000	0.0000	0.0000
41	Comp Mole Frac (CO)	0.0001	0.0001	0.0281	0.0281	0.1339
42	Comp Mole Frac (Hydrogen)	0.0432	0.0432	0.3158	0.3158	0.4486
43	Comp Mole Frac (i-Butane)	0.0000	0.0000	0.0000	0.0000	0.0000
44	Comp Mole Frac (n-Pentane)	0.0000	0.0000	0.0000	0.0000	0.0000
45	Comp Mole Frac (i-Pentane)	0.0000	0.0000	0.0000	0.0000	0.0000
46	Comp Mole Frac (n-Hexane)	0.0000	0.0000	0.0000	0.0000	0.0000
47	Comp Mole Frac (HCN)	0.0000	0.0000	0.0000	0.0000	0.0000
48	Comp Mole Frac (Ammonia)	0.0002	0.0002	0.0001	0.0001	0.0000
49						
50						
51						
52						
53						
54						
55						
56						
57						
58						
59						
60						
61						
62						
63	Aspen Technology Inc.		Aspen HYSYS Version 11		Page 2 of 4	

1	<div></div> <div>NORWEGIAN UNIVERSITY OF Bedford, MA USA</div>			Case Name: case_28.06.22_TemporarilyDone.hsc		
2				Unit Set: NewUser		
3				Date/Time: Fri Jul 15 14:32:16 2022		
4						
5						
6						
7	Workbook: Case (Main) (continued)					
8						
9						
10	Compositions (continued)				Fluid Pkg:	All
11	Name	9	O2	10	11	12
12	Comp Mole Frac (Methane)	0.0018	0.0000 *	0.0018	0.0018	0.0018
13	Comp Mole Frac (Ethane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
14	Comp Mole Frac (Propane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
15	Comp Mole Frac (n-Butane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
16	Comp Mole Frac (CO2)	0.0564	0.0000 *	0.0564	0.1853	0.1853
17	Comp Mole Frac (Oxygen)	0.0000	1.0000 *	0.0000	0.0000	0.0000
18	Comp Mole Frac (Nitrogen)	0.0009	0.0000 *	0.0009	0.0009	0.0009
19	Comp Mole Frac (H2S)	0.0000	0.0000 *	0.0000	0.0000	0.0000
20	Comp Mole Frac (H2O)	0.3583	0.0000 *	0.3583	0.2295	0.2295
21	Comp Mole Frac (H2O2)	0.0000	0.0000 *	0.0000	0.0000	0.0000
22	Comp Mole Frac (CO)	0.1339	0.0000 *	0.1339	0.0051	0.0051
23	Comp Mole Frac (Hydrogen)	0.4486	0.0000 *	0.4486	0.5774	0.5774
24	Comp Mole Frac (i-Butane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
25	Comp Mole Frac (n-Pentane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
26	Comp Mole Frac (i-Pentane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
27	Comp Mole Frac (n-Hexane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
28	Comp Mole Frac (HCN)	0.0000	0.0000 *	0.0000	0.0000	0.0000
29	Comp Mole Frac (Ammonia)	0.0000	0.0000 *	0.0000	0.0000	0.0000
30	Name	13	14	15	H2	Tailgas
31	Comp Mole Frac (Methane)	0.0018	0.0023	0.0000	0.0000 *	0.0092
32	Comp Mole Frac (Ethane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
33	Comp Mole Frac (Propane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
34	Comp Mole Frac (n-Butane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
35	Comp Mole Frac (CO2)	0.1853	0.2392	0.0023	0.0000 *	0.9479
36	Comp Mole Frac (Oxygen)	0.0000	0.0000	0.0000	0.0000 *	0.0000
37	Comp Mole Frac (Nitrogen)	0.0009	0.0012	0.0000	0.0000 *	0.0046
38	Comp Mole Frac (H2S)	0.0000	0.0000	0.0000	0.0000 *	0.0000
39	Comp Mole Frac (H2O)	0.2295	0.0030	0.9975	0.0000 *	0.0119
40	Comp Mole Frac (H2O2)	0.0000	0.0000	0.0000	0.0000 *	0.0000
41	Comp Mole Frac (CO)	0.0051	0.0066	0.0000	0.0000 *	0.0263
42	Comp Mole Frac (Hydrogen)	0.5774	0.7476	0.0000	1.0000 *	0.0000
43	Comp Mole Frac (i-Butane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
44	Comp Mole Frac (n-Pentane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
45	Comp Mole Frac (i-Pentane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
46	Comp Mole Frac (n-Hexane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
47	Comp Mole Frac (HCN)	0.0000	0.0000	0.0000	0.0000 *	0.0000
48	Comp Mole Frac (Ammonia)	0.0000	0.0000	0.0001	0.0000 *	0.0000
49						
50	Energy Streams				Fluid Pkg:	All
51	Name	E-100-Duty	E-101-Duty	GHR-Duty	E-102-Duty	ITR-Duty
52	Heat Flow (kJ/h)	1.321e+008	1.509e+007	1.004e+008	1.004e+008	-8.249e+007
53	Name	E-103-Duty				
54	Heat Flow (kJ/h)	9.048e+007				
55						
56	Unit Ops					
57	Operation Name	Operation Type	Feeds	Products	Ignored	Calc Level
58	MIX-100	Mixer	H2O	1	No	500.0
59			NG			
60	E-100	Heater	1	2	No	500.0
61			E-100-Duty			
62	E-101	Heater	3	5	No	500.0
63	Aspen Technology Inc.		Aspen HYSYS Version 11		Page 3 of 4	

1	 NORWEGIAN UNIVERSITY OF Bedford, MA USA			Case Name: case_28.06.22_TemporarilyDone.hsc								
2				Unit Set: NewUser								
3				Date/Time: Fri Jul 15 14:32:16 2022								
4												
5												
6	Workbook: Case (Main) (continued)											
7												
8												
9	Unit Ops (continued)											
10												
11	Operation Name	Operation Type	Feeds	Products	Ignored	Calc Level						
12	E-101	Heater	E-101-Duty		No	500.0 *						
13	ADJ-1	Adjust			No	3500 *						
14	ADJ-2	Adjust			No	3500 *						
15	ADJ-3	Adjust			No	3500 *						
16	SPRDSHT-1	Spreadsheet			No	500.0 *						
17	SPRDSHT-2	Spreadsheet			No	500.0 *						
18	Prereformer	Gibbs Reactor	2	4	No	500.0 *						
19				3								
20	GHR	Gibbs Reactor	5	7	No	500.0 *						
21			GHR-Duty	6								
22				GHR-Duty								
23	ATR	Gibbs Reactor	6	9	No	500.0 *						
24			O2	8								
25	Isothermal Reactor	Gibbs Reactor	10	12	No	500.0 *						
26			ITR-Duty	11								
27				ITR-Duty								
28	E-102	Cooler	8	10	No	500.0 *						
29				E-102-Duty								
30	E-103	Cooler	11	13	No	500.0 *						
31				E-103-Duty								
32	V-100	Separator	13	15	No	500.0 *						
33				14								
34	X-100	Component Splitter	14	H2	No	500.0 *						
35				Tailgas								
36												
37												
38												
39												
40												
41												
42												
43												
44												
45												
46												
47												
48												
49												
50												
51												
52												
53												
54												
55												
56												
57												
58												
59												
60												
61												
62												
63							Aspen Technology Inc.		Aspen HYSYS Version 11		Page 4 of 4	

C Hysys simulation with 90% oxygen


1	<div></div> <div>NORWEGIAN UNIVERSITY OF Bedford, MA USA</div>			Case Name: case_28.06.22_TemporarilyDone.hsc		
2				Unit Set: NewUser		
3				Date/Time: Fri Jul 15 14:32:16 2022		
4						
5						
6						
7	Workbook: Case (Main)					
8						
9						
10	Material Streams				Fluid Pkg:	All
11	Name	H2O	NG	1	2	3
12	Vapour Fraction	0.0000	1.0000 *	0.3181	1.0000	1.0000
13	Temperature (C)	150.0 *	38.00 *	117.9	420.0 *	382.4
14	Pressure (kPa)	3000 *	3000 *	3000	3000 *	3000
15	Molar Flow (kgmole/h)	2368 *	1000 *	3368	3368	3466
16	Mass Flow (kg/h)	4.265e+004	1.687e+004	5.952e+004	5.952e+004	5.952e+004
17	Liquid Volume Flow (m3/h)	42.74	54.89	97.63	97.63	101.3
18	Heat Flow (kJ/h)	-6.541e+008	-7.604e+007	-7.302e+008	-5.981e+008	-5.981e+008
19	Name	4	5	6	7	8
20	Vapour Fraction	0.0000	1.0000	1.0000	0.0000	1.0000
21	Temperature (C)	382.4	480.0 *	700.0 *	700.0	1050
22	Pressure (kPa)	3000	3000 *	3000	3000	3000
23	Molar Flow (kgmole/h)	0.0000	3466	4096	0.0000	5428
24	Mass Flow (kg/h)	0.0000	5.952e+004	5.952e+004	0.0000	7.410e+004
25	Liquid Volume Flow (m3/h)	0.0000	101.3	122.9	0.0000	147.9
26	Heat Flow (kJ/h)	0.0000	-5.830e+008	-4.826e+008	0.0000	-4.796e+008
27	Name	9	O2	10	11	12
28	Vapour Fraction	0.0000	1.0000	1.0000	1.0000	0.0000
29	Temperature (C)	1050	250.0 *	546.8 *	250.0 *	250.0
30	Pressure (kPa)	3000	3000 *	3000	3000	3000
31	Molar Flow (kgmole/h)	0.0000	455.6 *	5428	5428	0.0000
32	Mass Flow (kg/h)	0.0000	1.458e+004	7.410e+004	7.410e+004	0.0000
33	Liquid Volume Flow (m3/h)	0.0000	12.82	147.9	168.2	0.0000
34	Heat Flow (kJ/h)	0.0000	3.069e+006	-5.800e+008	-6.624e+008	0.0000
35	Name	13	14	15	H2	Tailgas
36	Vapour Fraction	0.7723	1.0000	0.0000	1.0000	0.9971
37	Temperature (C)	40.00 *	40.00	40.00	40.00 *	59.06
38	Pressure (kPa)	3000 *	3000	3000	3000 *	3000
39	Molar Flow (kgmole/h)	5428	4192	1236	3134	1058
40	Mass Flow (kg/h)	7.410e+004	5.175e+004	2.234e+004	6318	4.543e+004
41	Liquid Volume Flow (m3/h)	168.2	145.8	22.42	90.44	55.37
42	Heat Flow (kJ/h)	-7.529e+008	-4.003e+008	-3.526e+008	1.342e+006	-4.017e+008
43						
44						
45						
46						
47						
48						
49						
50						
51						
52						
53						
54						
55						
56						
57						
58						
59						
60						
61						
62						
63	Aspen Technology Inc.		Aspen HYSYS Version 11		Page 1 of 4	


Licensed to: NORWEGIAN UNIVERSITY OF


* Specified by user.

Licensed to: NORWEGIAN UNIVERSITY OF


* Specified by user.


1	<div></div> <div>NORWEGIAN UNIVERSITY OF Bedford, MA USA</div>			Case Name: case_28.06.22_TemporarilyDone.hsc		
2				Unit Set: NewUser		
3				Date/Time: Wed Jul 27 14:08:55 2022		
4						
5						
6	Workbook: Case (Main) (continued)					
7						
8						
9						
10	Compositions				Fluid Pkg: All	
11	Name	H2O	NG	1	2	3
12	Comp Mole Frac (Methane)	0.0000 *	0.9470 *	0.2812	0.2812	0.2857
13	Comp Mole Frac (Ethane)	0.0000 *	0.0420 *	0.0125	0.0125	0.0000
14	Comp Mole Frac (Propane)	0.0000 *	0.0020 *	0.0006	0.0006	0.0000
15	Comp Mole Frac (n-Butane)	0.0000 *	0.0002 *	0.0001	0.0001	0.0000
16	Comp Mole Frac (CO2)	0.0000 *	0.0030 *	0.0009	0.0009	0.0152
17	Comp Mole Frac (Oxygen)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000
18	Comp Mole Frac (Nitrogen)	0.0000 *	0.0050 *	0.0015	0.0015	0.0014
19	Comp Mole Frac (H2S)	0.0000 *	0.0000 *	0.0000	0.0000	0.0000
20	Comp Mole Frac (H2O)	1.0000 *	0.0000 *	0.7030	0.7030	0.6544
21	Comp Mole Frac (H2O2)	0.0000 *	0.0000 *	0.0000	0.0000	0.0000
22	Comp Mole Frac (CO)	0.0000 *	0.0000 *	0.0000	0.0000	0.0001
23	Comp Mole Frac (Hydrogen)	0.0000 *	0.0002 *	0.0001	0.0001	0.0432
24	Comp Mole Frac (i-Butane)	0.0000 *	0.0002 *	0.0001	0.0001	0.0000
25	Comp Mole Frac (n-Pentane)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000
26	Comp Mole Frac (i-Pentane)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000
27	Comp Mole Frac (n-Hexane)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000
28	Comp Mole Frac (HCN)	0.0000 *	0.0000 *	0.0000	0.0000	0.0000
29	Comp Mole Frac (Ammonia)	0.0000 *	0.0000 *	0.0000	0.0000	0.0002
30	Name	4	5	6	7	8
31	Comp Mole Frac (Methane)	0.2857	0.2857	0.1650	0.1650	0.0017
32	Comp Mole Frac (Ethane)	0.0000	0.0000	0.0000	0.0000	0.0000
33	Comp Mole Frac (Propane)	0.0000	0.0000	0.0000	0.0000	0.0000
34	Comp Mole Frac (n-Butane)	0.0000	0.0000	0.0000	0.0000	0.0000
35	Comp Mole Frac (CO2)	0.0152	0.0152	0.0616	0.0616	0.0561
36	Comp Mole Frac (Oxygen)	0.0000	0.0000	0.0000	0.0000	0.0000
37	Comp Mole Frac (Nitrogen)	0.0014	0.0014	0.0012	0.0012	0.0102
38	Comp Mole Frac (H2S)	0.0000	0.0000	0.0000	0.0000	0.0000
39	Comp Mole Frac (H2O)	0.6544	0.6544	0.4282	0.4282	0.3558
40	Comp Mole Frac (H2O2)	0.0000	0.0000	0.0000	0.0000	0.0000
41	Comp Mole Frac (CO)	0.0001	0.0001	0.0281	0.0281	0.1326
42	Comp Mole Frac (Hydrogen)	0.0432	0.0432	0.3158	0.3158	0.4436
43	Comp Mole Frac (i-Butane)	0.0000	0.0000	0.0000	0.0000	0.0000
44	Comp Mole Frac (n-Pentane)	0.0000	0.0000	0.0000	0.0000	0.0000
45	Comp Mole Frac (i-Pentane)	0.0000	0.0000	0.0000	0.0000	0.0000
46	Comp Mole Frac (n-Hexane)	0.0000	0.0000	0.0000	0.0000	0.0000
47	Comp Mole Frac (HCN)	0.0000	0.0000	0.0000	0.0000	0.0000
48	Comp Mole Frac (Ammonia)	0.0002	0.0002	0.0001	0.0001	0.0001
49						
50						
51						
52						
53						
54						
55						
56						
57						
58						
59						
60						
61						
62						
63	Aspen Technology Inc.		Aspen HYSYS Version 11		Page 2 of 4	


1	<div></div> <div>NORWEGIAN UNIVERSITY OF Bedford, MA USA</div>			Case Name: case_28.06.22_TemporarilyDone.hsc		
2				Unit Set: NewUser		
3				Date/Time: Wed Jul 27 14:08:55 2022		
4						
5						
6						
7	Workbook: Case (Main) (continued)					
8						
9						
10	Compositions (continued)				Fluid Pkg:	All
11	Name	9	O2	10	11	12
12	Comp Mole Frac (Methane)	0.0017	0.0000 *	0.0017	0.0017	0.0017
13	Comp Mole Frac (Ethane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
14	Comp Mole Frac (Propane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
15	Comp Mole Frac (n-Butane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
16	Comp Mole Frac (CO2)	0.0561	0.0000 *	0.0561	0.1836	0.1836
17	Comp Mole Frac (Oxygen)	0.0000	0.9000 *	0.0000	0.0000	0.0000
18	Comp Mole Frac (Nitrogen)	0.0102	0.1000 *	0.0102	0.0102	0.0102
19	Comp Mole Frac (H2S)	0.0000	0.0000 *	0.0000	0.0000	0.0000
20	Comp Mole Frac (H2O)	0.3558	0.0000 *	0.3558	0.2283	0.2283
21	Comp Mole Frac (H2O2)	0.0000	0.0000 *	0.0000	0.0000	0.0000
22	Comp Mole Frac (CO)	0.1326	0.0000 *	0.1326	0.0051	0.0051
23	Comp Mole Frac (Hydrogen)	0.4436	0.0000 *	0.4436	0.5711	0.5711
24	Comp Mole Frac (i-Butane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
25	Comp Mole Frac (n-Pentane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
26	Comp Mole Frac (i-Pentane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
27	Comp Mole Frac (n-Hexane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
28	Comp Mole Frac (HCN)	0.0000	0.0000 *	0.0000	0.0000	0.0000
29	Comp Mole Frac (Ammonia)	0.0001	0.0000 *	0.0001	0.0001	0.0001
30	Name	13	14	15	H2	Tailgas
31	Comp Mole Frac (Methane)	0.0017	0.0022	0.0000	0.0000 *	0.0085
32	Comp Mole Frac (Ethane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
33	Comp Mole Frac (Propane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
34	Comp Mole Frac (n-Butane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
35	Comp Mole Frac (CO2)	0.1836	0.2367	0.0023	0.0000 *	0.9047
36	Comp Mole Frac (Oxygen)	0.0000	0.0000	0.0000	0.0000 *	0.0000
37	Comp Mole Frac (Nitrogen)	0.0102	0.0131	0.0000	0.0000 *	0.0502
38	Comp Mole Frac (H2S)	0.0000	0.0000	0.0000	0.0000 *	0.0000
39	Comp Mole Frac (H2O)	0.2283	0.0030	0.9973	0.0000 *	0.0115
40	Comp Mole Frac (H2O2)	0.0000	0.0000	0.0000	0.0000 *	0.0000
41	Comp Mole Frac (CO)	0.0051	0.0065	0.0000	0.0000 *	0.0250
42	Comp Mole Frac (Hydrogen)	0.5711	0.7383	0.0000	1.0000 *	0.0000
43	Comp Mole Frac (i-Butane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
44	Comp Mole Frac (n-Pentane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
45	Comp Mole Frac (i-Pentane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
46	Comp Mole Frac (n-Hexane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
47	Comp Mole Frac (HCN)	0.0000	0.0000	0.0000	0.0000 *	0.0000
48	Comp Mole Frac (Ammonia)	0.0001	0.0000	0.0004	0.0000 *	0.0001
49						
50	Energy Streams				Fluid Pkg:	All
51	Name	E-100-Duty	E-101-Duty	GHR-Duty	E-102-Duty	ITR-Duty
52	Heat Flow (kJ/h)	1.321e+008	1.509e+007	1.004e+008	1.004e+008	-8.384e+007
53	Name	E-103-Duty				
54	Heat Flow (kJ/h)	9.103e+007				
55						
56	Unit Ops					
57	Operation Name	Operation Type	Feeds	Products	Ignored	Calc Level
58	MIX-100	Mixer	H2O	1	No	500.0
59			NG			
60	E-100	Heater	1	2	No	500.0
61			E-100-Duty			
62	E-101	Heater	3	5	No	500.0
63	Aspen Technology Inc.		Aspen HYSYS Version 11		Page 3 of 4	

1	 NORWEGIAN UNIVERSITY OF Bedford, MA USA			Case Name: case_28.06.22_TemporarilyDone.hsc								
2				Unit Set: NewUser								
3				Date/Time: Wed Jul 27 14:08:55 2022								
4												
5												
6	Workbook: Case (Main) (continued)											
7												
8												
9	Unit Ops (continued)											
10												
11	Operation Name	Operation Type	Feeds	Products	Ignored	Calc Level						
12	E-101	Heater	E-101-Duty		No	500.0 *						
13	ADJ-1	Adjust			No	3500 *						
14	ADJ-2	Adjust			No	3500 *						
15	ADJ-3	Adjust			No	3500 *						
16	SPRDSHT-1	Spreadsheet			No	500.0 *						
17	SPRDSHT-2	Spreadsheet			No	500.0 *						
18	Prereformer	Gibbs Reactor	2	4	No	500.0 *						
19				3								
20	GHR	Gibbs Reactor	5	7	No	500.0 *						
21			GHR-Duty	6								
22				GHR-Duty								
23	ATR	Gibbs Reactor	6	9	No	500.0 *						
24			O2	8								
25	Isothermal Reactor	Gibbs Reactor	10	12	No	500.0 *						
26			ITR-Duty	11								
27				ITR-Duty								
28	E-102	Cooler	8	10	No	500.0 *						
29				E-102-Duty								
30	E-103	Cooler	11	13	No	500.0 *						
31				E-103-Duty								
32	V-100	Separator	13	15	No	500.0 *						
33				14								
34	X-100	Component Splitter	14	H2	No	500.0 *						
35				Tailgas								
36												
37												
38												
39												
40												
41												
42												
43												
44												
45												
46												
47												
48												
49												
50												
51												
52												
53												
54												
55												
56												
57												
58												
59												
60												
61												
62												
63							Aspen Technology Inc.		Aspen HYSYS Version 11		Page 4 of 4	

D Hysys simulation with 40% oxygen

1	<div></div> <div>NORWEGIAN UNIVERSITY OF Bedford, MA USA</div>			Case Name: case_28.06.22_TemporarilyDone.hsc		
2				Unit Set: NewUser		
3				Date/Time: Fri Jul 15 14:32:16 2022		
4						
5						
6						
7	Workbook: Case (Main)					
8						
9						
10	Material Streams				Fluid Pkg: All	
11	Name	H2O	NG	1	2	3
12	Vapour Fraction	0.0000	1.0000 *	0.3181	1.0000	1.0000
13	Temperature (C)	150.0 *	38.00 *	117.9	420.0 *	382.4
14	Pressure (kPa)	3000 *	3000 *	3000	3000 *	3000
15	Molar Flow (kgmole/h)	2368 *	1000 *	3368	3368	3466
16	Mass Flow (kg/h)	4.265e+004	1.687e+004	5.952e+004	5.952e+004	5.952e+004
17	Liquid Volume Flow (m3/h)	42.74	54.89	97.63	97.63	101.3
18	Heat Flow (kJ/h)	-6.541e+008	-7.604e+007	-7.302e+008	-5.981e+008	-5.981e+008
19	Name	4	5	6	7	8
20	Vapour Fraction	0.0000	1.0000	1.0000	0.0000	1.0000
21	Temperature (C)	382.4	480.0 *	700.0 *	700.0	1050
22	Pressure (kPa)	3000	3000 *	3000	3000	3000
23	Molar Flow (kgmole/h)	0.0000	3466	4096	0.0000	5428
24	Mass Flow (kg/h)	0.0000	5.952e+004	5.952e+004	0.0000	7.410e+004
25	Liquid Volume Flow (m3/h)	0.0000	101.3	122.9	0.0000	147.9
26	Heat Flow (kJ/h)	0.0000	-5.830e+008	-4.826e+008	0.0000	-4.796e+008
27	Name	9	O2	10	11	12
28	Vapour Fraction	0.0000	1.0000	1.0000	1.0000	0.0000
29	Temperature (C)	1050	250.0 *	546.8 *	250.0 *	250.0
30	Pressure (kPa)	3000	3000 *	3000	3000	3000
31	Molar Flow (kgmole/h)	0.0000	455.6 *	5428	5428	0.0000
32	Mass Flow (kg/h)	0.0000	1.458e+004	7.410e+004	7.410e+004	0.0000
33	Liquid Volume Flow (m3/h)	0.0000	12.82	147.9	168.2	0.0000
34	Heat Flow (kJ/h)	0.0000	3.069e+006	-5.800e+008	-6.624e+008	0.0000
35	Name	13	14	15	H2	Tailgas
36	Vapour Fraction	0.7723	1.0000	0.0000	1.0000	0.9971
37	Temperature (C)	40.00 *	40.00	40.00	40.00 *	59.06
38	Pressure (kPa)	3000 *	3000	3000	3000 *	3000
39	Molar Flow (kgmole/h)	5428	4192	1236	3134	1058
40	Mass Flow (kg/h)	7.410e+004	5.175e+004	2.234e+004	6318	4.543e+004
41	Liquid Volume Flow (m3/h)	168.2	145.8	22.42	90.44	55.37
42	Heat Flow (kJ/h)	-7.529e+008	-4.003e+008	-3.526e+008	1.342e+006	-4.017e+008
43						
44						
45						
46						
47						
48						
49						
50						
51						
52						
53						
54						
55						
56						
57						
58						
59						
60						
61						
62						
63	Aspen Technology Inc.		Aspen HYSYS Version 11		Page 1 of 4	
Licensed to: NORWEGIAN UNIVERSITY OF		* Specified by user.				

1	<div></div> <div>NORWEGIAN UNIVERSITY OF Bedford, MA USA</div>			Case Name: case_27.07.22_0.4O2.hsc			
2							
3				Unit Set: NewUser			
4				Date/Time: Wed Jul 27 14:11:02 2022			
5							
6							
7	Workbook: Case (Main) (continued)						
8							
9							
10	Compositions					Fluid Pkg: All	
11	Name	H2O	NG	1	2	3	
12	Comp Mole Frac (Methane)	0.0000 *	0.9470 *	0.2812	0.2812	0.2857	
13	Comp Mole Frac (Ethane)	0.0000 *	0.0420 *	0.0125	0.0125	0.0000	
14	Comp Mole Frac (Propane)	0.0000 *	0.0020 *	0.0006	0.0006	0.0000	
15	Comp Mole Frac (n-Butane)	0.0000 *	0.0002 *	0.0001	0.0001	0.0000	
16	Comp Mole Frac (CO2)	0.0000 *	0.0030 *	0.0009	0.0009	0.0152	
17	Comp Mole Frac (Oxygen)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000	
18	Comp Mole Frac (Nitrogen)	0.0000 *	0.0050 *	0.0015	0.0015	0.0014	
19	Comp Mole Frac (H2S)	0.0000 *	0.0000 *	0.0000	0.0000	0.0000	
20	Comp Mole Frac (H2O)	1.0000 *	0.0000 *	0.7030	0.7030	0.6544	
21	Comp Mole Frac (H2O2)	0.0000 *	0.0000 *	0.0000	0.0000	0.0000	
22	Comp Mole Frac (CO)	0.0000 *	0.0000 *	0.0000	0.0000	0.0001	
23	Comp Mole Frac (Hydrogen)	0.0000 *	0.0002 *	0.0001	0.0001	0.0432	
24	Comp Mole Frac (i-Butane)	0.0000 *	0.0002 *	0.0001	0.0001	0.0000	
25	Comp Mole Frac (n-Pentane)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000	
26	Comp Mole Frac (i-Pentane)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000	
27	Comp Mole Frac (n-Hexane)	0.0000 *	0.0001 *	0.0000	0.0000	0.0000	
28	Comp Mole Frac (HCN)	0.0000 *	0.0000 *	0.0000	0.0000	0.0000	
29	Comp Mole Frac (Ammonia)	0.0000 *	0.0000 *	0.0000	0.0000	0.0002	
30	Name	4	5	6	7	8	
31	Comp Mole Frac (Methane)	0.2857	0.2857	0.1650	0.1650	0.0011	
32	Comp Mole Frac (Ethane)	0.0000	0.0000	0.0000	0.0000	0.0000	
33	Comp Mole Frac (Propane)	0.0000	0.0000	0.0000	0.0000	0.0000	
34	Comp Mole Frac (n-Butane)	0.0000	0.0000	0.0000	0.0000	0.0000	
35	Comp Mole Frac (CO2)	0.0152	0.0152	0.0616	0.0616	0.0518	
36	Comp Mole Frac (Oxygen)	0.0000	0.0000	0.0000	0.0000	0.0000	
37	Comp Mole Frac (Nitrogen)	0.0014	0.0014	0.0012	0.0012	0.1213	
38	Comp Mole Frac (H2S)	0.0000	0.0000	0.0000	0.0000	0.0000	
39	Comp Mole Frac (H2O)	0.6544	0.6544	0.4282	0.4282	0.3254	
40	Comp Mole Frac (H2O2)	0.0000	0.0000	0.0000	0.0000	0.0000	
41	Comp Mole Frac (CO)	0.0001	0.0001	0.0281	0.0281	0.1160	
42	Comp Mole Frac (Hydrogen)	0.0432	0.0432	0.3158	0.3158	0.3842	
43	Comp Mole Frac (i-Butane)	0.0000	0.0000	0.0000	0.0000	0.0000	
44	Comp Mole Frac (n-Pentane)	0.0000	0.0000	0.0000	0.0000	0.0000	
45	Comp Mole Frac (i-Pentane)	0.0000	0.0000	0.0000	0.0000	0.0000	
46	Comp Mole Frac (n-Hexane)	0.0000	0.0000	0.0000	0.0000	0.0000	
47	Comp Mole Frac (HCN)	0.0000	0.0000	0.0000	0.0000	0.0000	
48	Comp Mole Frac (Ammonia)	0.0002	0.0002	0.0001	0.0001	0.0003	
49							
50							
51							
52							
53							
54							
55							
56							
57							
58							
59							
60							
61							
62							
63	Aspen Technology Inc.		Aspen HYSYS Version 11			Page 2 of 4	

1	<div></div> <div>NORWEGIAN UNIVERSITY OF Bedford, MA USA</div>			Case Name: case_27.07.22_0.4O2.hsc		
2				Unit Set: NewUser		
3				Date/Time: Wed Jul 27 14:11:02 2022		
4						
5						
6	Workbook: Case (Main) (continued)					
7						
8						
9						
10	Compositions (continued)				Fluid Pkg:	All
11	Name	9	O2	10	11	12
12	Comp Mole Frac (Methane)	0.0011	0.0000 *	0.0011	0.0011	0.0011
13	Comp Mole Frac (Ethane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
14	Comp Mole Frac (Propane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
15	Comp Mole Frac (n-Butane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
16	Comp Mole Frac (CO2)	0.0518	0.0000 *	0.0518	0.1636	0.1636
17	Comp Mole Frac (Oxygen)	0.0000	0.4000 *	0.0000	0.0000	0.0000
18	Comp Mole Frac (Nitrogen)	0.1213	0.6000 *	0.1213	0.1213	0.1213
19	Comp Mole Frac (H2S)	0.0000	0.0000 *	0.0000	0.0000	0.0000
20	Comp Mole Frac (H2O)	0.3254	0.0000 *	0.3254	0.2136	0.2136
21	Comp Mole Frac (H2O2)	0.0000	0.0000 *	0.0000	0.0000	0.0000
22	Comp Mole Frac (CO)	0.1160	0.0000 *	0.1160	0.0042	0.0042
23	Comp Mole Frac (Hydrogen)	0.3842	0.0000 *	0.3842	0.4959	0.4959
24	Comp Mole Frac (i-Butane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
25	Comp Mole Frac (n-Pentane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
26	Comp Mole Frac (i-Pentane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
27	Comp Mole Frac (n-Hexane)	0.0000	0.0000 *	0.0000	0.0000	0.0000
28	Comp Mole Frac (HCN)	0.0000	0.0000 *	0.0000	0.0000	0.0000
29	Comp Mole Frac (Ammonia)	0.0003	0.0000 *	0.0003	0.0003	0.0003
30	Name	13	14	15	H2	Tailgas
31	Comp Mole Frac (Methane)	0.0011	0.0014	0.0000	0.0000 *	0.0037
32	Comp Mole Frac (Ethane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
33	Comp Mole Frac (Propane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
34	Comp Mole Frac (n-Butane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
35	Comp Mole Frac (CO2)	0.1636	0.2071	0.0020	0.0000 *	0.5585
36	Comp Mole Frac (Oxygen)	0.0000	0.0000	0.0000	0.0000 *	0.0000
37	Comp Mole Frac (Nitrogen)	0.1213	0.1539	0.0000	0.0000 *	0.4150
38	Comp Mole Frac (H2S)	0.0000	0.0000	0.0000	0.0000 *	0.0000
39	Comp Mole Frac (H2O)	0.2136	0.0031	0.9968	0.0000 *	0.0083
40	Comp Mole Frac (H2O2)	0.0000	0.0000	0.0000	0.0000 *	0.0000
41	Comp Mole Frac (CO)	0.0042	0.0053	0.0000	0.0000 *	0.0143
42	Comp Mole Frac (Hydrogen)	0.4959	0.6293	0.0000	1.0000 *	0.0000
43	Comp Mole Frac (i-Butane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
44	Comp Mole Frac (n-Pentane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
45	Comp Mole Frac (i-Pentane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
46	Comp Mole Frac (n-Hexane)	0.0000	0.0000	0.0000	0.0000 *	0.0000
47	Comp Mole Frac (HCN)	0.0000	0.0000	0.0000	0.0000 *	0.0000
48	Comp Mole Frac (Ammonia)	0.0003	0.0001	0.0011	0.0000 *	0.0002
49						
50	Energy Streams				Fluid Pkg:	All
51	Name	E-100-Duty	E-101-Duty	GHR-Duty	E-102-Duty	ITR-Duty
52	Heat Flow (kJ/h)	1.321e+008	1.509e+007	1.004e+008	1.004e+008	-1.022e+008
53	Name	E-103-Duty				
54	Heat Flow (kJ/h)	9.852e+007				
55						
56	Unit Ops					
57	Operation Name	Operation Type	Feeds	Products	Ignored	Calc Level
58	MIX-100	Mixer	H2O	1	No	500.0
59			NG			
60	E-100	Heater	1	2	No	500.0
61			E-100-Duty			
62	E-101	Heater	3	5	No	500.0
63	Aspen Technology Inc.		Aspen HYSYS Version 11		Page 3 of 4	

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58


59

60

61

62

63



NORWEGIAN UNIVERSITY OF
Bedford, MA
USA

Case Name:case_27.07.22_0.4O2.hsc

Unit Set:NewUser

Date/Time:Wed Jul 27 14:11:02 2022

Workbook: Case (Main) (continued)

Unit Ops (continued)

Operation Name	Operation Type	Feeds	Products	Ignored	Calc Level
E-101	Heater	E-101-Duty		No	500.0 *
ADJ-1	Adjust			No	3500 *
ADJ-2	Adjust			No	3500 *
ADJ-3	Adjust			No	3500 *
SPRDSHT-1	Spreadsheet			No	500.0 *
SPRDSHT-2	Spreadsheet			No	500.0 *
Prereformer	Gibbs Reactor	2	4	No	500.0 *
			3		
GHR	Gibbs Reactor	5	7	No	500.0 *
		GHR-Duty	6		
			GHR-Duty		
ATR	Gibbs Reactor	6	9	No	500.0 *
		O2	8		
Isothermal Reactor	Gibbs Reactor	10	12	No	500.0 *
		ITR-Duty	11		
			ITR-Duty		
E-102	Cooler	8	10	No	500.0 *
			E-102-Duty		
E-103	Cooler	11	13	No	500.0 *
			E-103-Duty		
V-100	Separator	13	15	No	500.0 *
			14		
X-100	Component Splitter	14	H2	No	500.0 *
			Tailgas		

Aspen Technology Inc. Aspen HYSYS Version 11 Page 4 of 4